9.3

IBM MQ İçin Uygulamalar Geliştirilmesi

Not

Bu bilgileri ve desteklediği ürünü kullanmadan önce, “Özel notlar” sayfa 1241 bölümündeki bilgileri okuyun.

Bu basım, yeni basımlarda tersi belirtilmedikçe, IBM® MQ sürüm 9 yayın düzeyi 3 ve sonraki tüm yayınlar ve değişiklikler için geçerlidir.

IBM'e bilgi gönderdiğinizde, IBM ' e bu bilgileri size hiçbir sorumluluk yüklemeden uygun gördüğü yöntemlerle kullanması ya da dağıtması için münhasır olmayan bir hak verirsiniz.

İçindekiler

IBM MQ için uygulama geliştirilmesi

İleti göndermek ve almak için uygulamalar geliştirebilir ve kuyruk yöneticilerinizi ve ilgili kaynakları yönetebilirsiniz. IBM MQ , birçok farklı dilde ve çerçevede yazılmış uygulamaları destekler.

IBM MQiçin uygulama geliştirmede yeni misiniz?

IBM MQiçin uygulama geliştirme hakkında bilgi edinmek üzere IBM Developer adresini ziyaret edin:

- IBM MQ Developer Essentials *(temel bilgileri öğrenin, bir gösterim çalıştırın, bir uygulamayı kodlayın, daha gelişmiş öğretici programlar edinin)*
- IBM MQ Geliştiriciler İçin Yüklemeler *(ücretsiz geliştirici sürümleri ve deneme sürümleri dahil)*

Aşağıdaki bölümlerde açıklanan kavramları biliyorsanız, uygulamalarınızı geliştirmeyi daha kolay bulabilirsiniz:

- “Uygulama geliştirme kavramları” sayfa 6
- “IBM MQ uygulamaları için tasarımla ilgili önemli noktalar” sayfa 47

Nesne odaklı diller ve çerçeveler için destek

IBM MQ , aşağıdaki dillerde ve çerçevede geliştirilen uygulamalar için temel destek sağlar:

- JMS
- Java
- C++
- .NET

Ayrıca bkz. “Nesne yönelimli uygulamalar” sayfa 15.

.NET birçok dilde geliştirilen uygulamaları destekler. .NET ile ilgili IBM MQ sınıflarının IBM MQ kuyruklarına erişmesini göstermek için, MQ ürün belgeleri aşağıdaki dillere ilişkin bilgileri içerir:

- C# örnek kod ve örnek uygulamalar
- C++ örnek uygulamaları
- Visual Basic örnek uygulamalar

Bkz. “IBM MQ .NET programlarının yazılması ve konuşlandırılması” sayfa 556.

IBM MQ , IBM MQ 9.1.1 ortamındaki Windows uygulamaları için .NET Core ürününü ve IBM MQ 9.1.2ortamındaki Linux® ortamlarındaki uygulamalar için destekler. Daha fazla bilgi için bkz “kurmaIBM MQ classes for .NET” sayfa 534.

ALW IBM MQ , OASIS AMQP 1.0 iletişim kuralını uygulayan AMQP istemcilerini de destekler.

MQ Light, Apache Qpid Proton ve Apache Qpid JMS API ' leri gibi Apache Qpid istemcileri bu iletişim kuralını temel alır.

MQ Light API ' leri IBM MQ Lightadresinde bulunur.

Apache Qpid istemcileri QPid Protonadresinden edinilebilir.

Aşağıdaki dil bağ tanımları olduğu gibi sağlanır:

- a Go binding (Git bağlama)
- Node.js uygulamalarıyla çalışan bir JavaScript API uygulaması

Programlı REST API ' leri için destek

IBM MQ , ileti göndermek ve almak için aşağıdaki programlı REST API ' leri için destek sağlar:

- IBM MQ messaging REST API
- z/OS IBM z/OS Connect EE
- IBM Integration Bus
- IBM DataPower Ağ Geçidi

IBM Developer 'ın IBM MQ alanındaki “IBM MQ ile REST uygulamaları geliştirilmesi” sayfa 674başlıklı konuya ve ayrıca IBM MQ ileti alışverişi REST API 'sini kullanmaya başlayın eğitmenine bakın. Bu öğretici program, IBM MQ messaging REST APIile birlikte kullanılmak üzere, "olduğu gibi" esasıyla sağlanan aşağıdaki dillerdeki örnekleri içerir:

- MQ ileti alışverişi REST API 'sini kullanan örnek
- Node.js örneği, HTTPS modülünü kullanma
- Promise modülü ile Node.js örneği

Yordamsal programlama dilleri için destek

IBM MQ , aşağıdaki yordam programlama dillerinde geliştirilen uygulamalar için destek sağlar:

- C
- Windows Visual Basic (yalnızcaWindows sistemleri)
- COBOL
- z/OS Çevirici (yalnızcaIBM MQ for z/OS)
- z/OS PL/I (yalnızcaIBM MQ for z/OS)
- IBM i RPG (yalnızcaIBM MQ for IBM i)

Bu diller, ileti kuyruklama hizmetlerine erişmek için ileti kuyruğu arabirimini (MQI) kullanır. Bkz. “IBM MQ ile MQI uygulamaları geliştirilmesi” sayfa 688. Nesne yönelimli diller ve çerçeveler tarafından kullanılan IBM MQ Nesne Modelinin, MQI kullanılarak yordam dillerinde kullanılamayan ek işlevler sağladığını unutmayın.

Uygulama adının belirtilmesi

ALW

IBM MQ 9.1.2' den önce, Java ya da JMS istemci uygulamalarında bir uygulama adı belirtebilirsiniz. IBM MQ 9.1.2' den ek programlama dillerinde uygulama adını da belirtebilirsiniz. Daha fazla bilgi için bkz “Desteklenen programlama dillerinde uygulama adının belirtilmesi” sayfa 50.

İlgili görevler

“MQ Telemetry için uygulama geliştirilmesi” sayfa 1188

“IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi” sayfa 1210
IBM MQ için Microsoft Windows Communication Foundation (WCF) özel kanalı, WCF istemcileri ve hizmetleri arasında ileti gönderir ve alır.

İlgili başvurular

“Managed File Transfer için uygulama geliştirilmesi” sayfa 1167
Managed File Transferile çalıştırılacak programları belirtin, Managed File Transferile Apache Ant kullanın, kullanıcı çıkışlarıyla Managed File Transfer 'u özelleştirin ve Managed File Transfer ' u aracı komut kuyruğuna iletileri koyarak denetleyin.

Uygulama geliştirme kavramları

IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

IBM MQiçin yazabileceğiniz uygulama tipleri hakkında bilgi için bkz. “IBM MQ için uygulama geliştirilmesi” sayfa 5 ve “Uygulamalarınızın gerçekleştirebileceği işlemler” sayfa 7.

İlgili kavramlar

“IBM MQ uygulamaları için tasarımla ilgili önemli noktalar” sayfa 47
Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğine karar verdiğinizde, IBM MQtarafından sunulan özellikleri nasıl kullanacağınıza karar vermeniz gerekir.

Uygulamalarınızın gerçekleştirebileceği işlemler

İş süreçlerinizi desteklemek için gereksinim duyduğunuz iletileri göndermek ve almak üzere uygulamalar geliştirebilirsiniz. Kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetecek uygulamalar da geliştirebilirsiniz.

Uygulamalarınızın IBM MQ for Multiplatforms üzerinde gerçekleştirebileceği işlemler

Multi

Çoklu platformlarüzerinde, aşağıdaki eylemleri gerçekleştiren uygulamalar yazabilirsiniz:

- Aynı işletim sistemleri altında çalışan diğer uygulamalara ileti gönderin. Uygulamalar aynı sistemde ya da başka bir sistemde olabilir.
- Diğer IBM MQ platformlarında çalışan uygulamalara ileti gönderin.
- Aşağıdaki sistemler için CICS içinden ileti kuyruklama özelliğini kullanın:
  - AIX TXSeries - AIX
  - IBM i IBM i
  - Windows Windows
- Aşağıdaki sistemler için Encina içinden ileti kuyruklama özelliğini kullanın:
  - AIX AIX
  - Windows Windows
- Aşağıdaki sistemler için Tuxedo içinden ileti kuyruklama özelliğini kullanın:
  - AIX AIX
  - AT & T
  - Windows Windows
- IBM MQ iş birimlerinde dış kaynak yöneticileri tarafından yapılan güncellemeleri eşgüdümleyen bir hareket yöneticisi olarak IBM MQ kullanın. Aşağıdaki dış kaynak yöneticileri desteklenir ve X/OPEN XA arabirimiyle uyumludur
  - Db2
  - Informix
  - Oracle
  - Sybase
- Birden çok iletiyi, kesinleştirilebilecek ya da geriletilebilecek tek bir iş birimi olarak işleyin.
- Tam bir IBM MQ ortamından çalıştırın ya da IBM MQ istemci ortamından çalıştırın.

Uygulamalarınızın IBM MQ for z/OS üzerinde gerçekleştirebileceği işlemler

z/OS

z/OSüzerinde, aşağıdaki eylemleri gerçekleştiren uygulamalar yazabilirsiniz:

- CICS ya da IMSiçinde ileti kuyruklama özelliğini kullanın.
- Toplu iş, CICSve IMS uygulamaları arasında her işlev için en uygun ortamı seçerek ileti gönderin.
- Diğer IBM MQ platformlarında çalışan uygulamalara ileti gönderin.
- Birden çok iletiyi, kesinleştirilebilecek ya da geriletilebilecek tek bir iş birimi olarak işleyin.
- IMS köprüsü aracılığıyla IMS uygulamalarına ileti gönderin ve bu uygulamalarla etkileşim kurun.
- RRS tarafından koordine edilen iş birimlerine katılın.

z/OS içindeki her ortamın kendi özellikleri, avantajları ve dezavantajları vardır. IBM MQ for z/OS ' in avantajı, uygulamaların herhangi bir ortama bağlı olmamasıdır, ancak her ortamın avantajlarından yararlanmak için dağıtılabilir. Örneğin, TSO ya da CICSkullanarak son kullanıcı arabirimleri geliştirebilir, z/OS kümesinde işleme yoğun modüller çalıştırabilir ve veritabanı uygulamalarını IMS ya da CICSiçinde çalıştırabilirsiniz. Her durumda, uygulamanın çeşitli kısımları iletileri ve kuyrukları kullanarak iletişim kurabilir.

IBM MQ uygulamalarının tasarımcıları, bu ortamların gerektirdiği farklılıkları ve sınırlamaları bilmelidirler. Örneğin:

- IBM MQ , kuyruk yöneticileri arasında iletişim kurulmasına izin veren olanaklar sağlar (bu, *dağıtılmış kuyruğa alma*olarak bilinir).
- Değişiklikleri kesinleştirme ve yedekleme yöntemleri, toplu iş ve CICS ortamları arasında farklılık gösterir.
- IBM MQ for z/OS , çevrimiçi ileti işleme programları (MPPs), etkileşimli hızlı yol programları (IFP) ve toplu ileti işleme programları (BMP) için IMS ortamında destek sağlar. If you are writing batch DL/I programs, follow the guidance given in topics such as “z/OS toplu iş uygulamaları oluşturma” sayfa 980 and “z/OS toplu işte dikkat edilmesi gereken noktalar” sayfa 702 for z/OS batch programs.
- Tek bir z/OS sisteminde birden çok IBM MQ for z/OS yönetim ortamı bulunabilse de, bir CICS bölgesi aynı anda yalnızca bir kuyruk yöneticisine bağlanabilir. Ancak, aynı kuyruk yöneticisine birden çok CICS bölgesi bağlanabilir. IMS ve z/OS toplu iş ortamlarında, programlar birden çok kuyruk yöneticisine bağlanabilir.
- IBM MQ for z/OS , yerel kuyrukların bir grup kuyruk yöneticisi tarafından paylaşılmasına olanak sağlayarak daha yüksek verim ve kullanılabilirlik sağlar. Bu tür kuyruklara *paylaşılan kuyruk*adı verilir ve kuyruk yöneticileri, aynı paylaşılan kuyruklardaki iletileri işleyebilen bir *kuyruk paylaşım grubu*oluşturur. Toplu iş uygulamaları, belirli bir kuyruk yöneticisi adı yerine kuyruk paylaşım grubu adını belirterek, bir kuyruk paylaşım grubu içindeki birden çok kuyruk yöneticisinden birine bağlanabilir. Bu, *grup toplu bağlantısı*ya da daha basit *grup bağlantısı*olarak bilinir. Bkz. Paylaşılan kuyruklar ve kuyruk paylaşım grupları.

z/OS Desteklenen ortamlar ve bunların sınırlamaları arasındaki farklar “IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852içinde daha ayrıntılı olarak açıklanmıştır.

İlgili kavramlar

“Uygulama geliştirme kavramları” sayfa 6
IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

“IBM MQ uygulamaları için tasarımla ilgili önemli noktalar” sayfa 47
Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğine karar verdiğinizde, IBM MQtarafından sunulan özellikleri nasıl kullanacağınıza karar vermeniz gerekir.

“Kuyruğa alma için yordam uygulaması yazılması” sayfa 692
Kuyruğa alma uygulamaları yazma, bir kuyruk yöneticisine bağlanma ve bağlantı kesme, yayınlama/abone olma ve nesneleri açma ve kapatma hakkında bilgi edinmek için bu bilgileri kullanın.

“İstemci yordamsal uygulamaları yazılıyor” sayfa 875
IBM MQ üzerinde yordam dili kullanarak istemci uygulamaları yazmak için bilmeniz gerekenleri.

“IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma” sayfa 78

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , IBM MQile verilen Java ileti alışverişi sağlayıcılarıdır. JMS ve Jakarta Messaging belirtimlerinde tanımlanan arabirimleri gerçekleştirmenin yanı sıra, bu ileti alışverişi sağlayıcıları Java ileti sistemi API 'sine iki uzantı kümesi ekler.

“kullanmaIBM MQ classes for Java” sayfa 334
Bir Java ortamında IBM MQ kullanın. IBM MQ classes for Java , bir Java uygulamasının IBM MQ istemcisi olarak IBM MQ ' e bağlanmasına ya da IBM MQ kuyruk yöneticisine doğrudan bağlanmasına izin verir.

“.NET uygulamalarının geliştirilmesi” sayfa 533
IBM MQ classes for .NET , .NET uygulamalarının bir IBM MQ MQI client olarak IBM MQ ' e bağlanmasına ya da bir IBM MQ sunucusuna doğrudan bağlanmasına izin verir.

“C++ uygulamaları geliştirilmesi” sayfa 505
IBM MQ , IBM MQ nesnelerine eşdeğer C++ sınıflarını ve dizi veri tiplerine eşdeğer bazı ek sınıfları sağlar. MQI aracılığıyla kullanılamayan bazı özellikler sağlar.

“Yordamsal uygulama oluşturma” sayfa 957
Birkaç yordam dilinden birinde bir IBM MQ uygulaması yazabilir ve uygulamayı birkaç farklı platformda çalıştırabilirsiniz.

İlgili görevler

“IBM MQ örnek yordam programlarının kullanılması” sayfa 1013
Bu örnek programlar yordamsal dillerde yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir. Farklı platformlarda IBM MQ programları.

“IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi” sayfa 1210
IBM MQ için Microsoft Windows Communication Foundation (WCF) özel kanalı, WCF istemcileri ve hizmetleri arasında ileti gönderir ve alır.

güvenlik

Multi

Uygulamalar, uygulama adları ve uygulama örnekleri

Uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, uygulamalarla, uygulama adlarıyla ve uygulama örnekleriyle ilgili temel kavramları tanıyın.

Uygulamalar

Multi

Bir kuyruk yöneticisine yönelik bağlantılar, aynı *uygulama adını*sağlarsa, aynı *uygulamadan* olduğu varsayılır. Uygulama adı, DISPLAY CONN (*) TYPE CONN komutunun APPLTAG özniteliği olarak görüntülenir.

Notlar:

1. IBM MQ client Daha önceki IBM MQ 9.1.2sürümünü kullanan uygulamalar için uygulama adı, IBM MQ clienttarafından otomatik olarak ayarlanır. Değeri, uygulama programlama diline ve uygulamanın çalıştığı platforma bağlıdır. Daha fazla bilgi için bkz. PutApplAdı .
2. IBM MQ 9.1.2 ' te ya da daha sonraki bir yerde IBM MQ client kullanan IBM MQ client uygulamaları için uygulama adı belirli bir değere ayarlanabilir. Çoğu durumda bu, uygulama kodunda değişiklik yapılmasını ya da uygulamanın yeniden derlenmesi gerekmesini gerektirmez. Daha fazla bilgi için bkz. “Desteklenen programlama dillerinde uygulama adının kullanılması” sayfa 51 .

Uygulama örnekleri

Multi

Bağlantılar *uygulama yönetim ortamlarına*alt bölümlere ayrılır. Bir uygulamanın eşgörünümü, o uygulama için bir 'yürütme birimi' sağlayan, yakından ilişkili bağlantılar kümesıdır. Genellikle bu, bir dizi iş parçacığı ve ilişkili IBM MQ bağlantısına sahip olabilen tek bir işletim sistemi işlemidir.

IBM MQ for Multiplatforms üzerinde bir uygulama yönetim ortamı belirli bir Bağlantı Etiketiile ilişkilendirilir. Kuyruk yöneticisi, yeni bağlantıları var olan bir uygulama yönetim ortamıyla otomatik olarak ilişkilendirir; ancak, ilişkili olduklarını görebilir.

Notlar:

- İstemci bağlantıları kullanılıyorsa, bu işlemler çalışan bir ya da daha çok kanal üzerinden kuyruk yöneticisine bağlanabilir.
- JMS uygulamalarında, bir uygulama eşgörünümü belirli bir JMS bağlantısıyla ve ilişkili tüm JMS oturumlarıyla eşlenir.

Uygulama eşgörünümleri, tek tip küme otomatik uygulama dengelemesikullanılırken IBM MQ for Multiplatforms için özellikle önemlidir. IBM MQ for Multiplatforms altyapılarında, DISPLAY APSTATUS komutunu kullanarak bağlı uygulama yönetim ortamlarını görüntüleyebilirsiniz.

Bazı durumlarda, kuyruk yöneticisi uygulama eşgörünümü ilişkilendirmesine yönelik bağlantıyı doğru olarak gerçekleştiremiyor; özellikle:

- Aynı işlemden paylaşılan bir etkileşimle farklı uygulama adları kullanılarak birden çok bağlantı kurulursa.
- Daha eski düzeydeki istemci kitaplıkları kullanıyorsa. Örneğin, IBM MQ 9.1.2 ve daha önceki sürümlerde IBM MQ JMS istemci kuruluşları.

Bu durumlarda, uygulamalar kendilerini yeniden bağlanabilir olarak tanımlamazsa, buna izin verilir, ancak uygulama örneği gruplamalarından bazıları yanlış olabilir. Bağlantılardan herhangi biri MQCNO_RECONNECT olarak bildirilirse, bu durum uygulama dengelemeyi önemli ölçüde olumsuz etkiler; bu nedenle MQCONN çağrısı MQCNO_RECONNECT_INCOMPATIBLE ile reddedilir.

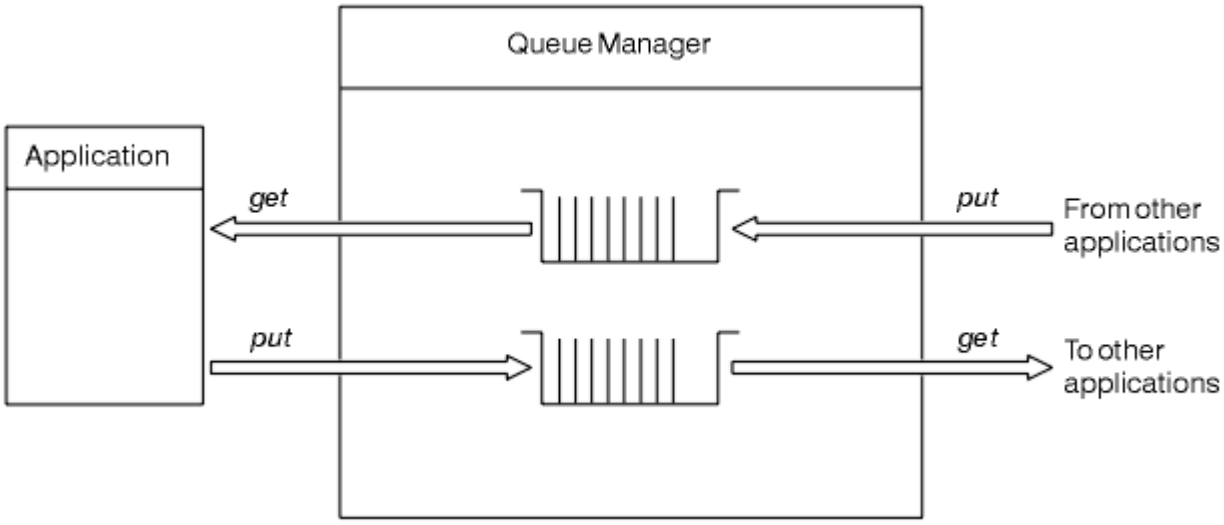
İlgili kavramlar

“Desteklenen programlama dillerinde uygulama adının belirtilmesi” sayfa 50
IBM MQ 9.2.0' den önce, Java ya da JMS istemci uygulamalarında bir uygulama adı belirtebilirsiniz. IBM MQ 9.2.0 ' den bu özellik, IBM MQ for Multiplatformsüzerindeki diğer programlama dillerine genişletilir.

MQI kullanan uygulama programları

IBM MQ uygulama programlarının başarıyla çalışabilmeleri için bazı nesneler gerekir.

Şekil 1 sayfa 10 içinde, kuyruktan ileti kaldıran, bunları işleyen ve daha sonra bazı sonuçları aynı kuyruk yöneticisindeki başka bir kuyruğa gönderen bir uygulama gösterilir.



Şekil 1. Kuyruklar, iletiler ve uygulamalar

Uygulamalar iletileri yerel ya da uzak kuyruklara yerleştirebilirken (MQPUT kullanılarak), iletileri yalnızca doğrudan yerel kuyruklardan alabilirler (MQGET kullanılarak).

Bu uygulamanın çalışabilmesi için aşağıdaki koşulların yerine getirilmesi gerekir:

- Kuyruk yöneticisi var olmalı ve çalışıyor olmalıdır.
- İletilerin kaldırılacağı ilk uygulama kuyruğu tanımlanmalıdır.
- Uygulamanın iletileri koyduğu ikinci kuyruk da tanımlanmalıdır.
- Uygulamanın kuyruk yöneticisine bağlanabilmesi gerekir. Bunu yapmak için IBM MQile bağlantılandırılmalıdır. Bkz. “Yordamsal uygulama oluşturma” sayfa 957.
- İletileri ilk kuyruğa koyan uygulamaların da bir kuyruk yöneticisine bağlanması gerekir. Bunlar uzaksa, iletim kuyrukları ve kanallarıyla da kurulmalıdır. Sistemin bu bölümü Şekil 1 sayfa 10içinde gösterilmez.

Birden çok IBM MQ kuyruk yöneticisine bağlanmak için istemci bağlantılarının kullanılması

İstemci bağlantılı uygulamalar birden çok kuyruk yöneticisine bağlanacak şekilde yapılandırılabilir (yük dengeleme ya da hizmet kullanılabilirliği nedeniyle).

IBM MQ istemcisinde bunu başarmak için birincil mekanizmalar, istemci kanal tanımlama çizelgelerinin kullanılması, İstemci kanal tanımlama çizelgelerinin yapılandırılmasıbaşlıklı konuya ya da bağlantı listelerine bakın.

Dış yük dengeleme ürünlerini kullanarak ya da anasistem adlarını ya da IP adreslerini yeniden yönlendirebilen bir ' sınırlı kod öbeğinde IBM MQ bağlantı kodunu kaydırarak benzer davranışlar da gerçekleştirilebilir.

Bu tekniklerin her biri bazı kısıtlamalarla birlikte gelir ve belirli uygulama gereksinimlerine az ya da çok uygun olabilir. Aşağıdaki bölümler, kapsamlı olmasa da, göz önünde bulundurmanız gereken belirli yönleri ve bu farklı yaklaşımların bu yönler üzerindeki etkisini açıklamaktadır.

IBM MQ tek tip kümeler, bkz. Tek tip kümeler hakkında, birden çok hedef sağlamak için CCDT ' nin temel mekanizmasını temel alan birden çok kuyruk yöneticisi arasında uygulamaların yatay ölçeklenmesini sağlamak için güçlü bir mekanizma sağlar. Tek tip kümeler, altta yatan IBM MQ iletişim kurallarının farkında olmayan bir dış yük dengeleyici kullanarak mümkün olan yeteneklerin ötesinde yetenekler sağlayabilir ve aşağıda tartışılan bazı sorunları önleyebilir, bu nedenle, uygun olan diğer tekniklere göre tek tip bir küme kullanmayı düşünebilirsiniz.

Uyarı: Yük dengeleme teknolojilerini kullanarak kuyruk yöneticilerine bağlanan IBM MQ Kaynak Bağdaştırıcılarından birini kullananlar da içinde olmak üzere IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingkullanan uygulamalarla birlikte kullanmanız gerekir. Sorunlarla karşılaşırsanız, yük dengelemeyi kullanmaya çalışmadan bu sorunları yeniden yaratın.

Bu tür bağlantıların en iyi sorunlu ve en kötü ihtimalle güvenilmez olduğu anlamına gelen birden çok sorun vardır:

- Herhangi bir yük dengeleme biçimi kullanılarak kuyruk yöneticisine birden çok bağlantı yapan herhangi bir uygulama bağlanırken özel özen gereklidir. Bu, genel kullanımda birden çok IBM MQ bağlantısı yaratırken JMS/Jakarta Messaging için IBM MQ Sınıflarını kullanan tüm uygulamaları içerir. Dış yük dengeleyici ya da özel kod sınırlı kod öbeği kullanılıyorsa, bu işlem bağlantıları aynı uygulama yönetim ortamından aynı kuyruk yöneticisine her zaman yönlendirmelidir.
- XA hareket yönetiminin ya da JTA ' nın (Java Transaction API) kullanılması, aynı kuyruk yöneticisine tutarlı olarak bağlanma yeteneğine dayanır-pratikte bu, herhangi bir yük dengeleme biçimiyle hiçbir zaman pratik olmaz.
- -Tek tip küme yönetimi, istemcilere müdahale olmadan belirli kuyruk yöneticilerine yeniden bağlanmaları için talimat verilmesine dayanır. Tek Tip Kümeler kullanımıyla dış yük dengelemeyi birleştirmeyi denemeniz önerilmez

Dış yük dengeleme teknolojileri yerine birden çok kuyruk yöneticisindeki uygulamaların yatay ölçeklenmesini sağlamak için IBM MQ tek tip küme işlevselliğini kullanmalısınız. Tek tip kümeleri nasıl oluşturduğunuz ve kullandığınız da dahil olmak üzere tek tip kümelerle ilgili bilgi için Tek tip kümeyi yapılandırmabaşlıklı konuya ve aşağıdaki konulara bakın.

Bu bilgilerde kullanılan terimler

CCDT-çoklu QMGR
Farklı CLNTCONN girişlerinin farklı kuyruk yöneticilerine çözüldüğü, kuyruk yöneticisi adı istemcisi bağlantısı (QMNAME CLNTCONN) özniteliği olan, aynı gruba sahip birden çok istemci bağlantısı (CLNTCONN) kanalı içeren bir CCDT dosyası anlamına gelir.

Bu, aynı çok eşgörünümlü kuyruk yöneticisi için farklı IP adresleri ya da anasistem adları olan birden çok CLNTCONN girişi içeren bir CCDT dosyasından farklıdır; bu, bir kod sınırlı kod öbeğiyle birleştirmeyi seçebileceğiniz bir yaklaşımdır.

Bir CCDT çoklu kuyruk yöneticisi yaklaşımını seçerseniz, girişlere öncelik mi vereceğinizi yoksa iş yükü yönetimini (WLM) rasgele mi yapacağınızı seçmeniz gerekir:

Önceliklendirilmiş
Son iyi bağlantıyı hatırlamak için CLNTWGHT (1) ve AFFINITY (PREFERRED) öznitelikleriyle birden çok alfabetik sıralı giriş kullanın.

Rasgele hale getirilmiş
CLNTWGHT (1) ve AFFINITY (NONE) özniteliklerini kullanın. WLM ağırlığını, CLNTWGHT ' yi ayarlayarak farklı ölçeklenen IBM MQ sunucularında ayarlayabilirsiniz.

Not: Kanallar arasındaki CLNTWGHT ' de büyük farklılıkları önleyin.

Yük dengeleyici
Birden çok IBM MQ kuyruk yöneticisinin TCP/IP dinleyicilerinin kapı izlemesiyle yapılandırılan Sanal IP adresi (VIP) olan bir ağ aracı anlamına gelir. VIP ' in ağ aracında nasıl yapılandırıldığı, kullandığınız ağ aygıtına bağlıdır.

Aşağıdaki seçenekler yalnızca ileti gönderen ya da zamanuyumlu istek ve yanıt ileti sistemi başlatan uygulamalarla ilgilidir. Bu iletilere ve isteklere hizmet eden uygulamalara ilişkin dikkat edilmesi gereken noktalar; örneğin, dinleyiciler tamamen ayrıdır ve "İleti dinleyicisinin kuyruğa bağlanması" konusunda ayrıntılı olarak ele alınmıştır.

Tek bir kuyruk yöneticisine bağlanan var olan uygulamalar için gereken kod değişikliği ölçeği

CONNAME listesi, CCDT çoklu QMGR ve Yük dengeleyici
MQCONN ("QMNAME") MQCONN 'a ("*QMNAME")

Kuyruk yöneticisi adı, Java Platform, Enterprise Edition (Java EE) uygulamaları için Java Naming and Directory Interface (JNDI) yapılandırmasında olabilir. Tersi durumda, tek karakterlik bir kod değişikliği gerekir.

Kod parçası
Var olan JMS ya da MQI bağlantısı mantığını bir kod koçanı ile değiştirin.

Farklı WLM stratejileri için destek

CONNAME listesi
Yalnızca önceliklendirilmiş.

Bunun kod üzerinde olumsuz bir etkisi olabilir.

CCDT çoklu QMGR
Önceliklendirilmiş ya da rastgele.

Bunun kod üzerinde herhangi bir etkisi olması olası değildir.

Yük dengeleyici
Tüm iletiler için her bağlantı da içinde olmak üzere, herhangi biri.

Bunun kod üzerinde olumlu bir etkisi olabilir.

Kod parçası
Tüm iletiler için her ileti de dahil olmak üzere herhangi biri.

Bunun kod üzerinde olumlu bir etkisi olabilir.

Birincil kuyruk yöneticisi kullanılamıyorsa performans ek yükü

CONNAME listesi
Her zaman listede ilk sıradadır.

Bunun kod üzerinde olumsuz bir etkisi olabilir.

CCDT çoklu QMGR
Son iyi bağlantıyı hatırlar.

Bunun kod üzerinde olumlu bir etkisi olabilir.

Yük dengeleyici
Kapı izleme, hatalı kuyruk yöneticilerini önler.

Bunun kod üzerinde olumlu bir etkisi olabilir.

Kod parçası
Son iyi bağlantıyı anımsayabilir ve akıllıca yeniden deneyebilir.

Bunun kod üzerinde olumlu bir etkisi olabilir.

XA hareket desteği

CONNAME listesi, CCDT çoklu QMGR ve Yük dengeleyici
Hareket yöneticisinin, aynı kuyruk yöneticisi kaynağına yeniden bağlanan kurtarma bilgilerini saklaması gerekir.

Farklı kuyruk yöneticilerine çözülen bir MQCONN çağrısı genellikle bunu geçersiz kılar. Örneğin, Java EE ' de tek bir bağlantı üreticisi XA kullanırken tek bir kuyruk yöneticisine çözülmelidir.

Bunun kod üzerinde olumsuz bir etkisi olabilir.

Kod parçası
Kod parçası, bir hareket yöneticisine ilişkin XA gereksinimlerini (örneğin, birden çok bağlantı üreticileri) karşılayabilir.

Bunun kod üzerinde olumlu bir etkisi olabilir.

Uygulamalardan altyapı değişikliklerini gizlemek için yönetici esnekliği

CONNAME listesi
Yalnızca DNS.

Bunun kod üzerinde olumsuz bir etkisi olabilir.

CCDT çoklu QMGR
DNS ve paylaşılan kütük sistemi ya da paylaşılan kütük sistemi ya da CCDT kütük gönderme.

Bunun kod üzerinde herhangi bir etkisi olması olası değildir.

Yük dengeleyici
Dinamik sanal IP adresi (VIP).

Bunun kod üzerinde olumlu bir etkisi olabilir.

Kod parçası
DNS ya da tek kuyruk yöneticisi CCDT girişleri.

Bunun kod üzerinde herhangi bir etkisi olması olası değildir.

Planlanan bakım kapsamında kesintiden kaçınma

Göz önünde bulundurmanız ve planlamanız gereken başka bir durum daha vardır; bu, bir kuyruk yöneticisinin planlı bakımı sırasında son kullanıcıların görebileceği hatalar ve zamanaşımları gibi uygulamalarda kesintiyi nasıl önleyebileceğidir. Kesintiyi önlemek için en iyi yaklaşım, durdurulmadan önce bir kuyruk yöneticisinden tüm işleri kaldırmaktır.

Bir istek ve yanıt senaryosu düşünün. Tüm devam eden isteklerin tamamlanmasını ve yanıtların uygulama tarafından işlenmesini istiyorsunuz, ancak sisteme ek bir iş gönderilmesini istemezsiniz. Kuyruk

yöneticisinin susturulması bu gereksinimi karşılamaz; iyi kodlanmış uygulamalar, hareket eden isteklere ilişkin yanıt iletilerini almadan önce RC2161 MQRC_Q_MGR_QUIESCING kural dışı durumu alır.

Hem PUT (ENABLED), hem de GET (ENABLED) yanıt kuyruklarını bırakırken, işi sunmak için kullanılan istek kuyruklarında PUT (DISABLED) ayarını ayarlayabilirsiniz. Bu şekilde, istek, iletim ve yanıt kuyruklarının derinliğini izleyebilirsiniz. Hepsi stabilize olduktan sonra, yani, uçuş sırasında istekler tamamlandı ya da zaman doldu, kuyruk yöneticisini durdurabilirsiniz.

Ancak, istekte bulunan uygulamalarda, ileti gönderme girişimi sırasında RC2051 MQRC_PUT_INENGELLENEN hata dönüş koduyla sonuçlanan bir PUT (DISABLED) istek kuyruğunu işlemek için iyi kodlama gerekir.

IBM MQbağlantısı yaratılırken ya da istek kuyruğu açılırken kural dışı durumun oluşmadığını unutmayın. Bu kural dışı durum yalnızca, MQPUT çağrısı kullanılarak gerçekten ileti gönderme girişiminde bulunulduğunda oluşur.

İstek ve yanıt senaryoları için bu hata işleme mantığını içeren bir kod parçası oluşturulması ve uygulama ekiplerinizin gelecekte bu tür bir kod parçasını kullanmalarını istemesi, tutarlı davranışla uygulamalar geliştirmenize yardımcı olabilir.

V 9.3.0 Esnek ve ölçeklenebilir istemci uygulamaları geliştirilmesi

Hataya dayanıklılık ve ölçeklenebilirlik için, bağlantı seçeneklerini tek tek kümelere destekleyen istemci uygulamalarının devreye alınması, uygulamanın eşgörünümlerinin kuyruk yöneticileri arasında yeniden dengelenmesini sağlar.

Tek tip kümelere genel bakış için bkz. Tek tip kümeler hakkında .

İdeal olarak, bu yeniden dengeleme uygulama için görünmez, ancak yalnızca belirli uygulama tipleri bu tür bir devreye alma için uygundur ve uygulama tasarımında dikkate alınması gerekebilir.

Bu konular iki ana kategoriye ayrılır:

- Yeniden bağlanabilir uygulamalar için var olabilecek, ancak tek tip bir kümede konuşlandırıldığında daha olası olan nadir *hata yolları* . Örneğin, yeniden bağlanmayı takiben, herhangi bir çalışma birimi geriletir ve imleçleri ilk durumuna getirir. Bunlar, yürürlükteki ortamında yeniden bağlanabilir uygulamanız için nadir görülen bir olay olabilir ve bu nedenle, uygulama kodu tarafından mümkün olduğu kadar düzgün bir şekilde işlenmeyebilir. Uygulama mantığının incelenmesi, bu tür durumlarda uygun işlemenin sağlandığından emin olmak için, ortaya çıkan beklenmedik sorunların önlenmesine yardımcı olur.
- Belirli bir kuyruk yöneticisine ilişkin *etkiler* . Bir uygulamanın her zaman aynı ya da belirli bir kuyruk yöneticisine yeniden bağlanması gerektiğini biliyorsanız, uygulama o kuyruk yöneticisine yeniden bağlanacak şekilde yapılandırılmalı ya da o kuyruk yöneticisiyle bağlantısı etkinleştirilmemelidir. Ancak bu yakınlıklar, yanıt iletisi bekleme gibi geçici olabilir. Uygulama kodundan bu yakınlıkları hesaba katmak için dengeleme algoritmasının etkilenmesi aşağıdaki bölümde ele alınmıştır. Bu seçeneklere ilişkin daha fazla ayrıntı ve uygulama kodu yerine yapılandırma aracılığıyla benzer bir yaklaşımın nasıl gerçekleştirileceğine ilişkin bilgi için Tek biçimli kümelerde uygulama yeniden dengelemesibaşlıklı konuya bakın.

MQI 'da yeniden bağlantı seçeneklerini etkileme

MQCNO_RECONNECT ile ilgili ek bilgi için Reconnection options (Yeniden uzlaştırma seçenekleri) konusuna bakın.

Bir uygulamanın her zaman aynı ya da belirli bir kuyruk yöneticisine geri bağlanması gerektiğini biliyorsanız, uygulamanın MQCNO_RECONNECT_Q_MGR ya da MQCNO_RECONNECT_DISABLED olarak yapılandırılması gerekir.

MQI 'da dengeleme algoritmasını etkileme

Ancak, yeniden dengeleme davranışını belirli uygulama tiplerinin gereksinimlerine uyacak şekilde denetlemek ya da etkilemek isteyebilirsiniz; örneğin, uçuş işlemlerindeki kesintileri en aza indirmek ya da istek sahibi uygulamalarının taşınmadan önce yanıtlarını almalarını sağlamak gibi.

Belirli varsayılan istenen davranışlar, Tek biçimli kümelerde uygulama yeniden dengelemesikonusunda kabul edilir ve tartışılır. client.ini dosyası aracılığıyla yapılandırma ya da konuşlandırma sırasında belirli uygulamaların davranışını da etkileyebilirsiniz.

Diğer durumlarda, dengeleme davranışının ve gereksinimlerinin uygulama mantığının bir parçası olması daha mantıklı olabilir. Bu durumlarda, MQCONNX çağrısında kuyruk yöneticisine bağlanırken, MQBNO (dengeleme seçenekleri) adlı bir yapıda uygulamanın aynı ilgili özellikleri IBM MQ ' e sağlanabilir.

Bir MQBNO yapısı sağlarsanız, uygulamanın farklı bir kuyruk yöneticisine nasıl ve ne zaman yeniden bağlanmasının istenmesi gerektiğine ilişkin bir karar almak için IBM MQ ' in gerektirdiği tüm bilgileri sağlamalıdır.

Aşağıdakileri sağlamalısınız:

- Uygulamanın **Type**
- Yönetim ortamının durumundan bağımsız olarak yeniden dengelendiği **Timeout**
- Herhangi bir özel **BalanceOptions**

Bunun kural dışı durumu, gerektiğinde zamanaşımını MQBNO_TIMEOUT_DEFAULT olarak bırakabilmektir. Bu durumda, zamanaşımı client.ini dosyasında, uygulamasında ya da genel kıtalarda herhangi bir değere çözülür (sağlandıysa) ve başarısız olursa, 10 saniyelik temel varsayılan değere ayarlanır.

Bu yapının biçimiyle ilgili ayrıntılar için MQBNO kısmına bakın.

.NET uygulamaları için, daha fazla bilgi için Influencing application re-balancing in .NET başlıklı konuya bakın.

Nesne yönelimli uygulamalar

IBM MQ , JMS, Java, C + + ve .NETiçin destek sağlar. Bu diller ve çerçeveler, IBM MQ çağrıları ve yapılarıyla aynı işlevselliği sağlayan sınıfları sağlayan IBM MQ Nesne Modeli 'ni kullanır.

IBM MQ Nesne Modeli 'ni kullanan bazı diller ve çerçeveler, ileti kuyruğu arabirimiyle (MQI) yordam dilleri kullandığınızda kullanılamayan ek işlevler sağlar.

Bu model tarafından sağlanan sınıfların, yöntemlerin ve özelliklerin ayrıntıları için bkz. “IBM MQ Nesne Modeli” sayfa 16.

JMS

IBM MQ , Jakarta Messaging 3.0 ve Java Message Service 2.0 belirtimlerini gerçekleştiren sınıfları sağlar. IBM MQ classes for JMSile ilgili ayrıntılar için bkz. IBM MQ classes for JMS ' yi kullanma. IBM MQ classes for Java ile IBM MQ classes for JMSarasındaki farklar hakkında bilgi için, hangisini kullanacağınıza karar vermenize yardımcı olması için bkz. “JMS/Jakarta Messaging ve Java uygulamalarının geliştirilmesi” sayfa 77.

IBM MQ Message Service Client (XMS) for C/C++ ve IBM MQ Message Service Client (XMS) for .NET , Java Message Service (JMS) ile aynı arabirim kümesine sahip XMS adlı bir uygulama programlama arabirimi (API) sağlar. API. Daha fazla bilgi için bkz “XMS .NET uygulamalarının geliştirilmesi” sayfa 591.

Java

Javaiçindeki IBM MQ Nesne Modeli 'ni kullanarak programların kodlanması hakkında bilgi için bkz. IBM MQ classes for Java ' yi kullanma .

Stabilized IBM , IBM MQ classes for Java üzerinde başka geliştirme yapmayacaktır ve bunlar IBM MQ 8.0içinde gönderilen düzeyde işlevsel olarak sabitlenecektir. Hangisini kullanacağınıza karar vermenize yardımcı olacak IBM MQ classes for Java ile IBM MQ classes for JMS arasındaki farklar hakkında bilgi için bkz. “JMS/Jakarta Messaging ve Java uygulamalarının geliştirilmesi” sayfa 77.

C++

IBM MQ , IBM MQ nesnelerine eşdeğer C++ sınıflarını ve dizi veri tiplerine eşdeğer bazı ek sınıfları sağlar. MQI aracılığıyla kullanılamayan bazı özellikler sağlar. IBM MQ Object Model in C + +. Message Service Clients for C/C++ and .NET , Java Message Service (JMS) ile aynı arabirim kümesine sahip XMS

adlı bir uygulama programlama arabirimi (API) sağlayan programların kodlanmasına ilişkin bilgi için bkz. C++ kullanarak . API.

.NET
IBM MQ .NET sınıflarını kullanarak .NET programlarının kodlanması hakkında bilgi için bkz. .NET uygulamaları geliştirilmesi . Message Service Clients for C/C++ and .NET , Java Message Service (JMS) ile aynı arabirim kümesine sahip XMS adlı bir uygulama programlama arabirimi (API) sağlar. API.

İlgili kavramlar
"IBM MQ ile MQI uygulamaları geliştirilmesi" sayfa 688
IBM MQ , C, Visual Basic, COBOL, Assembler, RPG, pTALve PL/I için destek sağlar. Bu yordamsal diller, ileti kuyruklama hizmetlerine erişmek için ileti kuyruğu arabirimini (MQI) kullanır.

Teknik genel bakış
"Uygulama geliştirme kavramları" sayfa 6
IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

İlgili başvurular
Uygulama başvurusu geliştirilmesi

IBM MQ Nesne Modeli

IBM MQ Nesne Modeli, sınıflardan, yöntemlerden ve özelliklerden oluşur.

IBM MQ Nesne Modeli şunlardan oluşur:

- Kuyruk yöneticileri, kuyruklar ve iletiler gibi bilinen IBM MQ kavramlarını gösteren *sınıflar* .
- MQI çağrılarına karşılık gelen her sınıfta *Yöntemler* .
- IBM MQ nesnelerinin özniteliklerine karşılık gelen her sınıfta *Özellikler* .

IBM MQ Nesne Modeli 'ni kullanarak bir IBM MQ uygulaması yaratırken, uygulamada bu sınıfların eşgörünümlerini yaratırsınız. Nesne yönelimli programlamada bir sınıfın eşgörünümüne *nesne*adı verilir. Bir nesne yaratıldığında, nesne özelliklerinin değerlerini inceleyerek ya da ayarlayarak (bir MQINQ ya da MQSET çağrısı yayınlamanın eşdeğeri) ve nesne için yöntem çağrıları yaparak (diğer MQI çağrılarını yayınlamanın eşdeğeri) nesneyle etkileşimde bulunabilirsiniz.

Sınıflar

IBM MQ Nesne Modeli aşağıdaki temel sınıf kümesini sağlar.

Modelin gerçek uygulaması, desteklenen farklı nesne yönelimli ortamlar arasında biraz değişiklik gösterir.

MQQueueManager
MQQueueManager sınıfının bir nesnesi, bir kuyruk yöneticisine yönelik bağlantıyı gösterir. Connect (), Disconnect (), Commit () ve Backout () yöntemleri vardır (MQCONN ya da MQCONNX, MQDISC, MQCMIT ve MQBACK ile eşdeğerdir). Bir kuyruk yöneticisinin özniteliklerine karşılık gelen özelliklere sahiptir. Bir kuyruk yöneticisi özniteliği özelliğine erişilmesi, önceden bağlanmamışsa, kuyruk yöneticisine örtük olarak bağlanır. Bir MQQueueManager nesnesinin yok edilmesi, kuyruk yöneticisiyle örtük olarak bağlantıyı keser.

MQQueue
MQQueue sınıfındaki bir nesne bir kuyruğu gösterir. Kuyruğa koyma () ve kuyruktan alma (alma) iletileri (MQPUT ve MQGET ile eşdeğeri) için yöntemleri vardır. Bir kuyruğun özniteliklerine karşılık gelen özelliklere sahiptir. Bir kuyruk özniteliği özelliğine erişilmesi ya da Put () ya da Get () yöntemi çağrısı verilmesi, kuyruğu örtük olarak açar (MQOPEN ' in eşdeğeri). Bir MQQueue nesnesinin yok edilmesi, kuyruğu örtük olarak kapatır (MQCLOSE eşdeğeri).

MQTopic
MQTopic sınıfının bir nesnesi bir konuyu temsil eder. Konuyla ilgili (MQPUT ve MQGET eşdeğeri) iletileri koymak () (yayınlamak) ve almak (almak ya da abone olmak) için yöntemleri vardır. Bir konunun özniteliklerine karşılık gelen özelliklere sahiptir. Bir MQTopic nesnesine aynı anda değil, yalnızca yayın ya da abonelik için erişilebilir. İleti almak için kullanıldığında, MQTopic nesnesi

yönetilmeyen ya da yönetilen bir abonelikle oluşturulabilir ve sürekli ya da sürekli olmayan bir abone olarak oluşturulabilir-bu farklı senaryolar için birden çok aşırı yüklenmiş oluşturucu sağlanır.

MQMessage
MQMessage sınıfının nesnesi, kuyruğa konacak ya da kuyruktan alınan bir iletiyi temsil eder. Bir arabellek içerir ve hem uygulama verilerini hem de MQMD ' yi kapsar. MQMD alanlarıyla ilgili özellikler ve arabelleğe ve arabellekten farklı tiplerde (örneğin, dizgiler, uzun tamsayılar, kısa tamsayılar, tek byte 'lar) kullanıcı verilerini yazmanızı ve okumanızı sağlayan yöntemler vardır.

MQPutMessageSeçenekleri
MQPutMessageSeçenekleri sınıfının bir nesnesi, MQPMO yapısını temsil eder. MQPMO alanlarıyla ilgili özellikler içeriyor.

MQGetMessageSeçenekleri
MQGetMessageSeçenekleri sınıfının bir nesnesi, MQGMO yapısını temsil eder. MQGMO alanlarıyla ilgili özellikler içeriyor.

MQProcess
MQProcess sınıfındaki bir nesne bir süreç tanımlamasını (tetikleyici ile kullanılır) gösterir. Bir süreç tanımlamasının özniteliklerini gösteren özellikler içerir.

Multi **MQDistributionList**
MQDistributionList sınıfının bir nesnesi bir dağıtım listesini temsil eder (tek bir MQPUT ile birden çok ileti göndermek için kullanılır). MQDistributionListÖğe nesnelerinin bir listesini içerir.

Multi **MQDistributionListÖğesi**
MQDistributionListÖğe sınıfının bir nesnesi, tek bir dağıtım listesi hedefini temsil eder. MQOR, MQRR ve MQPMR yapılarını içerir ve bu yapıların alanlarına karşılık gelen özelliklere sahiptir.

nesne başvuruları

MQI kullanan bir IBM MQ programında IBM MQ , programa bağlantı tanıtıcılarını ve nesne tanıtıcılarını döndürür.

Bu tanıtıcının sonraki IBM MQ çağrılarında parametre olarak iletilmesi gerekir. IBM MQ Nesne Modeli ile bu tanıtıcıları uygulama programından gizlenir. Bunun yerine, bir sınıftan nesne yaratılması, uygulama programına bir nesne başvurusunun döndürülmesine neden olur. Nesne için yöntem çağrıları ve özellik erişimleri yapılırken kullanılan bu nesne başvurusudur.

Dönüş kodları

Bir yöntem çağrısı verilmesi ya da bir özellik değeri ayarlanması, dönüş kodlarının ayarlanmasıyla sonuçlanır.

Bu dönüş kodları, bir tamamlanma kodu ve bir neden kodudur ve nesnenin özellikleridir. Tamamlanma kodu ve neden kodu değerleri, nesne yönelimli ortama özgü bazı ek değerlerle MQI için tanımlananlarla aynıdır.

IBM MQ ileti

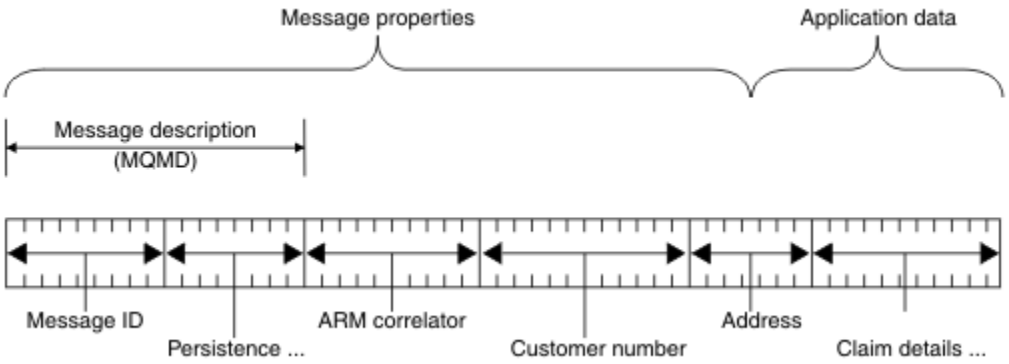
IBM MQ iletisi, ileti özellikleri ve uygulama verilerinden oluşur. İleti kuyruklama iletisi tanımlayıcısı (MQMD), bir ileti gönderen ve alan uygulamalar arasında dolaşırken uygulama verileriyle birlikte gönderilen denetim bilgilerini içerir.

İletinin kısımları

IBM MQ iletileri iki bölümden oluşur:

- İleti Özellikleri
- Uygulama Verileri

Şekil 2 sayfa 18 bir iletiyi gösterir ve iletinin mantıksal olarak ileti özelliklerine ve uygulama verilerine nasıl bölündüğünü gösterir.



Şekil 2. İletinin gösterimi

IBM MQ iletisinde taşınan uygulama verileri, üzerinde veri dönüştürme gerçekleştirilmedikçe kuyruk yöneticisi tarafından değiştirilmez. Ayrıca IBM MQ , bu verilerin içeriğine herhangi bir kısıtlama getirmez. Her iletideki verilerin uzunluğu, hem kuyruk yöneticisinin hem de kuyruk yöneticisinin **MaxMsgLength** özniteliğinin değerini aşamaz.

ALW AIX, Linux, and Windowsüzerinde, kuyruk yöneticisinin ve kuyruğun *MaxMsgLength* özniteliği varsayılan olarak 4 MB (4 194 304 bayt) değerine ayarlanır ve gerekirse en çok 100 MB (104 857 600 bayt) değerini değiştirebilirsiniz.

IBM i IBM iüzerinde, kuyruk yöneticisinin ve kuyruğun *MaxMsgLength* özniteliği varsayılan olarak 4 MB (4 194 304 bayt) değerine ayarlanır ve gerekirse en çok 100 MB (104 857 600 bayt) değerini değiştirebilirsiniz. IBM iişletim sistemlerinde 15 MB ' den büyük IBM MQ iletilerini kullanmayı planlıyorsanız, bkz. “IBM i üzerinde yordamsal uygulamanızı oluşturma” sayfa 963.

z/OS z/OSişletim sistemlerinde, kuyruk yöneticisinin **MaxMsgLength** özniteliği 100 MB olarak sabitlenir ve kuyruğun **MaxMsgLength** özniteliği varsayılan olarak 4 MB (4 194 304 bayt) değerine ayarlanır; bu, gerekirse en çok 100 MB olabilir.

İletilerinizi bazı durumlarda **MaxMsgLength** özniteliğinin değerinden biraz daha kısa yapın. Daha fazla bilgi için bkz “İletinizdeki veriler” sayfa 727.

MQPUT ya da MQPUT1 MQI çağrılarını kullandığınızda bir ileti yaratırsınız. Bu çağrılara giriş olarak, denetim bilgilerini (iletinin önceliği ve yanıt kuyruğunun adı gibi) ve verilerinizi girersiniz ve çağrı iletiyi kuyruğa koyar. Bu çağrılarla ilgili daha fazla bilgi için bkz. MQPUT ve MQPUT1 .

İleti tanımlayıcı

*İleti tanımlayıcıyı*tanımlayan MQMD yapısını kullanarak ileti denetimi bilgilerine erişebilirsiniz.

MQMD yapısının tam açıklaması için bkz. MQMD-Message descriptor.

İletinin kökeni hakkında bilgi içeren MQMD içindeki alanların nasıl kullanılacağına ilişkin bir açıklama için bkz. “İleti bağlamı” sayfa 45.

İleti tanımlayıcının farklı sürümleri var. İleti tanımlayıcısının (ya da MQMDE) Sürüm 2 'de iletileri gruplamaya ve bölümlere ayırmaya ilişkin ek bilgi (bkz. “İleti grupları” sayfa 42). Bu, Sürüm 1 ileti tanımlayıcısıyla aynıdır, ancak ek alanları vardır. Bu alanlar MQMDE-İleti tanımlayıcı uzantısındaaçıklanmıştır.

İleti tipleri

IBM MQtarafından tanımlanan dört tip ileti vardır.

Bu dört ileti şunlardır:

- Veri Paketi

- İstek iletileri
- Yanıt iletileri
- Rapor iletileri
  - Rapor iletisi tipleri
  - Rapor iletisi seçenekleri

Uygulamalar, kendi aralarında bilgi aktarmak için ilk üç ileti tipini kullanabilir. Dördüncü tür, rapor, uygulamaların ve kuyruk yöneticilerinin hata oluşumu gibi olaylarla ilgili bilgileri raporlamak için kullanmaları içindir.

Her ileti tipi bir MQMT_ * değeriyle tanıtılır. Kendi ileti tiplerinizi de tanımlayabilirsiniz. Kullanabileceğiniz değer aralığı için bkz. MsgType.

Veri paketleri

İletiyi alan (yani, iletiyi kuyruktan alan) uygulamadan yanıt almanız gerekmediğinde bir *veri paketi* kullanın.

Veri paketlerini kullanabilen bir uygulama örneği, bir havaalanı salonunda uçuş bilgilerini görüntüleyen bir uygulamadır. Bir ileti, uçuş bilgilerinin tüm ekranına ilişkin verileri içerebilir. Bu tür bir uygulama, bir iletinin teslim edilmemesi büyük olasılıkla önemli olmadığından, ileti için bir alındı bildirimi isteme olasılığına sahip değildir. Uygulama kısa bir süre sonra bir güncelleme iletisi gönderir.

İstek iletileri

İletiyi alan uygulamadan yanıt almak istediğinizde bir *istek iletisi* kullanın.

İstek iletilerini kullanabilen bir uygulama örneği, bir çek hesabının bakiyesini görüntüleyen uygulamadır. İstek iletisi hesabın numarasını içerebilir ve yanıt iletisi hesap bakiyesini içerir.

Yanıt iletinizi istek iletinizle ilişkilendirmek istiyorsanız, iki seçenek vardır:

- İstek iletisini işleyen uygulamayı, istek iletisiyle ilgili yanıt iletisine bilgi koymasını sağlamaktan sorumlu kılın.
- Yanıt iletisinin *MsgId* ve *CorrelId* alanlarının içeriğini belirtmek için istek iletinizin ileti tanımlayıcısındaki rapor alanını kullanın:
  - Özgün iletinin *MsgId* ya da `CorrelId` iletisinin yanıt iletisinin *CorrelId* alanına kopyalanmasını isteyebilirsiniz (varsayılan işlem *MsgId*öğesini kopyalamaktır).
  - Yanıt iletisi için yeni bir `MsgId` oluşturulmasını ya da özgün iletinin *MsgId* değerinin, yanıt iletisinin *MsgId* alanına kopyalanmasını isteyebilirsiniz (varsayılan işlem yeni bir ileti tanıtıcısı oluşturulmasıdır).

Yanıt iletileri

Başka bir iletiyi yanıtlarken *yanıt iletisi* kullanın.

Bir yanıt iletisi oluşturduğunuzda, yanıtladığınız iletinin ileti tanımlayıcısında ayarlanan seçeneklere saygı gösterin. Rapor seçenekleri, ileti tanıtıcısı (`MsgId`) ve ilinti tanıtıcısı (`CorrelId`) alanlarının içeriğini belirtir. Bu alanlar, yanıtı alan uygulamanın yanıtı özgün isteğiyle ilişkilendirmesini sağlar.

Rapor iletileri

Rapor iletileri , bir ileti işlenirken hata oluşması gibi olaylarla ilgili bilgi verir.

Bu öğeler aşağıdaki şekilde oluşturulabilir:

- Bir kuyruk yöneticisi,
- Bir ileti kanalı aracısı (örneğin, iletiyi teslim edemezse) ya da
- Bir uygulama (örneğin, iletideki verileri kullanamazsa).

Rapor iletileri herhangi bir zamanda oluşturulabilir ve uygulamanız bunları beklemediğinde bir kuyruğa gelebilir.

Rapor iletisi tipleri

Bir iletiyi kuyruğa koyduğunuzda, aşağıdakileri almayı seçebilirsiniz:

- *Kural dışı durum raporu iletisi*. Bu, kural dışı durum işareti ayarlanmış bir iletiye yanıt olarak gönderilir. İleti kanalı aracısı (MCA) ya da uygulama tarafından oluşturulur.
- Bir *süre bitimi rapor iletisi*. Bu, bir uygulamanın süre sonu eşiğine ulaşan bir iletiyi almayı denediğini gösterir; ileti atılacak olarak işaretlenir. Bu tip bir rapor, kuyruk yöneticisi tarafından oluşturulur.
- *geliş onayı (COA) rapor iletisi*. Bu, iletinin hedef kuyruğuna ulaştığını gösterir. Kuyruk yöneticisi tarafından oluşturulur.
- *Teslim onayı (COD) rapor iletisi*. Bu, iletinin alan bir uygulama tarafından alındığını gösterir. Kuyruk yöneticisi tarafından oluşturulur.
- *pozitif eylem bildirimi (PAN) rapor iletisi*. Bu, bir isteğin başarıyla hizmet verildiğini (yani, iletide istenen işlemin başarıyla gerçekleştirildiğini) gösterir. Bu tip bir rapor uygulama tarafından oluşturulur.
- *Negatif eylem bildirimi (NAN) rapor iletisi*. Bu, bir isteğin başarıyla hizmet verilmediğini (yani, iletide istenen işlemin başarıyla gerçekleştirilmediğini) gösterir. Bu tip bir rapor uygulama tarafından oluşturulur.

Not: Her rapor iletisi tipi aşağıdakilerden birini içerir:

- Tüm özgün ileti
- Özgün iletideki verilerin ilk 100 baytı
- Özgün iletiden veri yok

Kuyruğa bir ileti koyduğunuzda birden çok rapor iletisi tipi isteyebilirsiniz. Teslim doğrulama raporu iletisini ve kural dışı durum raporu ileti seçeneklerini seçerseniz, ileti teslim edilemezse, bir kural dışı durum raporu iletisi alırsınız. Ancak, yalnızca teslim onayı rapor iletisi seçeneğini belirlerseniz ve ileti teslim edilemezse, bir kural dışı durum raporu iletisi almazsanız.

Belirli bir ileti oluşturmaya ilişkin ölçütler karşılandığında, istekte bulunacağınız rapor iletileri yalnızca sizin aldığınız iletilerdir.

Rapor iletisi seçenekleri

Bir kural dışı durum ortaya çıktıktan sonra bir iletiyi *atabilirsiniz* . At seçeneğini belirlerseniz ve bir kural dışı durum raporu iletisi istediyseniz, rapor iletisi *ReplyToQ* ve *ReplyToQMgr*' e gider ve özgün ileti atılır.

Not: Bunun bir yararı, gitmeyen iletiler kuyruğuna giden iletilerin sayısını azaltabilmendir. Ancak, yalnızca veri paketi iletileri göndermediği sürece, uygulamanızın döndürülen iletilerle ilgilenmesi gerektiği anlamına gelir. Bir kural dışı durum raporu iletisi oluşturulduğunda, özgün iletinin kalıcılığını devralır.

Bir rapor iletisi teslim edilemezse (örneğin, kuyruk doluysa), rapor iletisi teslim edilmeyen iletiler kuyruğuna yerleştirilir.

Bir rapor iletisi almak istiyorsanız, *ReplyToQ* alanında yanıt kuyruğunun adını belirtin; tersi durumda, özgün iletinizin MQPUT ya da MQPUT1 MQRC_MISSING_REPLY_TO_Q ile başarısız olur.

İleti için oluşturulan rapor iletilerinin *MsgId* ve *CorrelId* alanlarının içeriğini belirtmek için bir iletinin ileti tanımlayıcısındaki (MQMD) diğer rapor seçeneklerini kullanabilirsiniz:

- Özgün iletinin *MsgId* ya da *CorrelId* öğesinin rapor iletisinin *CorrelId* alanına kopyalanmasını isteyebilirsiniz. Varsayılan işlem, ileti tanıtıcısını kopyalamaktır. MQRO_COPY_MSG_ID_TO_CORRELID, bir ileti gönderenin yanıt ya da rapor iletisini özgün iletiyle ilintilendirmesini sağladığından, bunu kullanın. Yanıt ya da rapor iletisinin ilinti tanıtıcısı, özgün iletinin ileti tanıtıcısıyla aynı.

- Rapor iletisi için yeni bir *MsgId* oluşturulmasını ya da özgün iletinin *MsgId* öğesinin rapor iletisinin *MsgId* alanına kopyalanmasını isteyebilirsiniz. Varsayılan işlem, yeni bir ileti tanıtıcısı oluşturulmasını belirtir. Sistemdeki her iletinin farklı bir ileti tanıtıcısına sahip olmasını ve sistemdeki diğer tüm iletilerden belirsiz bir şekilde ayırt edilebilmesini sağladığından MQRO_NEW_MSG_ID ' yi kullanın.
- Özelleştirilmiş uygulamaların MQRO_PASS_MSG_ID ya da MQRO_PASS_CORREL_ID kullanması gerekebilir. Ancak, kuyruktaki iletileri okuyan uygulamayı, örneğin, kuyruk aynı ileti tanıtıcısına sahip birden çok ileti içerdiğinde doğru çalıştığından emin olmak için tasarlamanız gerekir.

  Sunucu uygulamaları, istek iletisindeki bu işaretlerin ayarlarını denetleyip yanıt ya da rapor iletisindeki *MsgId* ve *CorrelId* alanlarını uygun şekilde ayarlamalıdır.

  İstekte bulunan uygulama ile sunucu uygulaması arasında aracı görevi yapan uygulamaların bu işaretlerin ayarlarını denetlemesi gerekmez. Bunun nedeni, bu uygulamaların genellikle iletiyi *MsgId*, *CorrelId*ve *Report* alanlarını değiştirmeden sunucu uygulamasına iletmeleri gerekmesidir. Bu, sunucu uygulamasının yanıt iletisinin *CorrelId* alanındaki özgün iletiden *MsgId* dosyasını kopyalamasını sağlar.

Bir ileti hakkında rapor oluştururken, sunucu uygulamaları bu seçeneklerden herhangi birinin ayarlı olup olmadığını görmek için test etmelidir.

Rapor iletilerinin nasıl kullanılacağı hakkında daha fazla bilgi için bkz. Rapor.

Raporun niteliğini belirtmek için kuyruk yöneticileri bir dizi geribildirim kodu kullanır. Bu kodları, bir rapor iletisinin ileti tanımlayıcısının *Feedback* alanına koyarlar. Kuyruk yöneticileri, *Feedback* alanında MQI neden kodlarını da döndürebilir. IBM MQ , uygulamaların kullanması için bir dizi geribildirim kodunu tanımlar.

Geribildirim ve neden kodlarıyla ilgili daha fazla bilgi için bkz. Geribildirim.

Geribildirim kodu kullanabilen bir program örneği, bir kuyruğa hizmet veren diğer programların iş yüklerini izleyen bir programdır. Bir kuyruğa hizmet eden programın birden fazla eşgörünümü varsa ve kuyruğa gelen ileti sayısı artık bunu haklı göstermezse, bu tür bir program, programın etkinliğini sonlandırması gerektiğini belirtmek için hizmet veren programlardan birine bir rapor iletisi (MQFB_QUIT geribildirim koduyla) gönderebilir. (Bir izleme programı, bir kuyruğa kaç program hizmet vermekte olduğunu öğrenmek için MQINQ çağrısına başvurabilir.)

Multi *Raporlar ve bölümlenmiş iletiler*

IBM MQ for z/OSüzerinde desteklenmez.

Bir ileti bölümlenirse ve raporların oluşturulmasını isterseniz, ileti bölümlenmeseydi yapacağınız rapordan daha fazla rapor alabilirsiniz.

Bölümlenmiş iletilerin açıklaması için bkz. “İleti bölümlemesi” sayfa 760.

IBM MQ tarafından oluşturulan raporlar için

İletilerinizi kesimlere böler ya da kuyruk yöneticisinin bunu yapmasına izin verirseniz, iletinin tamamı için tek bir rapor almayı bekleyebileceğiniz tek bir vaka vardır. Bu, yalnızca COD raporlarını istediğinizde ve alma uygulamasında MQGMO_COMPLETE_MSG belirttiğinizde olur.

Diğer durumlarda başvurunuz, genellikle her bölüm için bir olmak üzere çeşitli raporlarla ilgilenmek üzere hazırlanmalıdır.

Not: İletilerinizi kesimlere ayırırsanız ve özgün ileti verilerinin yalnızca ilk 100 baytı döndürülmesi gerekiyorsa, rapor seçeneklerinin ayarını değiştirerek, 100 ya da daha fazla göreli konumu olan kesimler için veri olmayan raporlar isteyin. Bunu yapmazsanız ve her bir kesimin 100 baytlık veri istemesi için ayarı bırakırsanız ve rapor iletilerini MQGMO_COMPLETE_MSG belirterek tek bir MQGET ile alırsınız, raporlar her uygun göreli konumda 100 baytlık okuma verisi içeren büyük bir iletide toplanır. Bu durumda büyük bir arabellek gerekir ya da MQGMO_ACCEPT_TRUNCATED_MSG belirtmeniz gerekir.

Uygulamalar tarafından oluşturulan raporlar için

Uygulamanız raporlar oluşturursa, özgün ileti verilerinin başlangıcında bulunan IBM MQ üstbilgilerini her zaman rapor ileti verilerine kopyalayın.

Daha sonra rapor ileti verilerine hiçbir şey, 100 bayt ya da tüm özgün ileti verilerini (ya da genellikle ekleyeceğiniz başka bir miktar) ekleyin.

Kopyalanması gereken IBM MQ üstbilgilerini, MQMD ' den başlayarak ve var olan üstbilgilerden devam ederek art arda Biçim adlarına bakarak tanıyabilirsiniz. Aşağıdaki `Format` adları bu IBM MQ üstbilgilerini gösterir:

- MQMDE
- MQDLH
- MQXQH
- MQIIH
- MQH*

MQH*, MQH karakterleriyle başlayan herhangi bir ad anlamına gelir.

`Format` adı, MQDLH ve MQXQH için belirli konumlarda, ancak diğer IBM MQ üstbilgileri için aynı konumda oluşur. Üstbilginin uzunluğu, MQMDE, MQIMSve tüm MQH* üstbilgileri için aynı konumda bulunan bir alanda bulunur.

Sürüm 1 MQMD kullanıyorsanız ve bir bölüm ya da gruptaki bir ileti ya da bölümlemeye izin verilen bir ileti hakkında raporlama yapıyorsanız, rapor verileri bir MQMDE ile başlamalıdır. *`OriginalLength`* alanını, bulacağınız IBM MQ üstbilgilerinin uzunlukları dışında, özgün ileti verilerinin uzunluğuna ayarlayın.

Raporlar alınıyor

COA ya da COD raporları isterseniz, bunların MQGMO_COMPLETE_MSG ile sizin için yeniden birleştirilmesini isteyebilirsiniz.

MQGMO_COMPLETE_MSG içeren bir MQGET, kuyrukta tam bir özgün iletiyi temsil etmek için yeterli sayıda rapor iletisi (örneğin, COA ve aynı *`GroupId`*ile) olduğunda karşılanır. Bu, rapor iletilerinin kendisi tüm orijinal verileri içermese bile doğrudur; her bir rapor iletisindeki *`OriginalLength`* alanı, verilerin kendisi mevcut olmasa da, bu rapor iletisiyle temsil edilen özgün verilerin uzunluğunu verir.

Kuyrukta birkaç farklı rapor tipi varsa (örneğin, COA ve COD) bu tekniği kullanabilirsiniz; çünkü MQGMO_COMPLETE_MSG içeren bir MQGET, iletileri yalnızca aynı *`Feedback`* koduna sahipse bildirir. Ancak, genel olarak bunlar farklı *`Feedback`* kodlarına sahip olduğundan, kural dışı durum raporları için genellikle bu tekniği kullanamazsınız.

Bu tekniği, iletinin tamamının ulaştığı konusunda olumlu bir belirti elde etmek için kullanabilirsiniz. Ancak, çoğu durumda, bazı kesimlerin gelme olasılığı için, bazılarının bir kural dışı durum oluşturması (ya da buna izin verdiyseniz, süre bitimi) için yemek yapmanız gerekir. Bu durumda MQGMO_COMPLETE_MSG kullanamazsınız; genel olarak, farklı kesimler için farklı *`Feedback`* kodları alabilir ve bir bölüm için birden fazla rapor edinebilirsiniz. Ancak, MQGMO_ALL_SEGMENTS_KULLANILABILIR öğesini kullanabilirsiniz.

Buna izin vermek için raporları geldiklerinde almanız ve özgün iletiye ne olduğuna ilişkin uygulamanızda bir resim oluşturmanız gerekebilir. Özgün iletinin *`GroupId`* ile raporları ilintilendirmek için rapor iletisindeki *`GroupId`* alanını ve her bir rapor iletisinin tipini tanımlamak için *`Feedback`* alanını kullanabilirsiniz. Bunu yapma şekliniz, uygulama gereksinimlerinize bağlıdır.

Bir yaklaşım aşağıdaki gibidir:

- COD raporlarını ve kural dışı durum raporlarını isteyin.
- Belirli bir süre sonra, MQGMO_COMPLETE_MSG kullanılarak eksiksiz bir COD raporları kümesinin alınıp alınmadığını denetleyin. Bu durumda, uygulamanız iletinin tamamının işlendiğini bilir.
- Yoksa ve bu iletiyle ilgili kural dışı durum raporları varsa, sorunu bölümlenmemiş iletiler için olduğu gibi işleyip, artık bölümleri bir noktada temizlediğinizden emin olun.

- Herhangi bir türde rapor olmayan bölümler varsa, orijinal bölümler (ya da raporlar) bir kanalın yeniden bağlanmasını bekliyor olabilir ya da ağ bir noktada aşırı yüklenmiş olabilir. Herhangi bir özel durum raporu alınmamışsa (ya da sahip olduğunuz raporların yalnızca geçici olabileceğini düşünüyorsanız), uygulamanızın biraz daha beklemesine izin vermeye karar verebilirsiniz.

  Daha önce olduğu gibi, bu, bölümlenmemiş iletilerle ilgilenirken sahip olduğunuz hususlara benzer, ancak artık kesimleri temizleme olasılığını da göz önünde bulundurmanız gerekir.

Özgün ileti kritik değilse (örneğin, bir sorgu ya da daha sonra yinelenebilecek bir iletiyse), artık bölümlerin kaldırıldığından emin olmak için bir süre sonu belirleyin.

Arka düzey kuyruk yöneticileri

Bir rapor, bölümlemeyi destekleyen bir kuyruk yöneticisi tarafından oluşturulduğunda, ancak bölümlemeyi desteklemeyen bir kuyruk yöneticisinde alındığında, rapor verilerine her zaman sıfır, 100 bayta ya da iletideki özgün verilerin tümüne ek olarak MQMDE yapısı (rapor tarafından gösterilen *Offset* ve *OriginalLength* öğelerini tanımlar) dahil edilir.

Ancak, bir iletinin bir bölümü bölümlemeyi desteklemeyen bir kuyruk yöneticisinden geçerse, orada bir rapor oluşturulursa, özgün iletideki MQMDE yapısı yalnızca veri olarak işlenir. Bu nedenle, özgün verilerin sıfır baytı istendiyse, rapor verilerine dahil edilmez. MQMDE olmadan, rapor iletisi yararlı olmayabilir.

İletinin bir arka düzey kuyruk yöneticisinden geçmesi olasılığı varsa, raporlarda en az 100 baytlık veri isteyin.

İleti denetim bilgilerinin ve ileti verilerinin biçimi

Kuyruk yöneticisi yalnızca bir ileti içindeki denetim bilgilerinin biçimiyle ilgilenirken, iletiyi işleyen uygulamalar hem denetim bilgilerinin hem de verilerin biçimiyle ilgilenir.

İleti denetim bilgilerinin biçimi

İleti tanımlayıcısının karakter dizgisi alanlarındaki denetim bilgileri, kuyruk yöneticisi tarafından kullanılan karakter kümesinde olmalıdır.

Kuyruk yöneticisi nesnesinin **CodedCharSetId** özniteliği bu karakter kümesini tanımlar. Uygulamalar bir kuyruk yöneticisinden diğerine ileti geçirdiğinde, iletileri ileten ileti kanalı aracıları hangi veri dönüştürmesinin gerçekleştirileceğini belirlemek için bu özniteliğin değerini kullandığından, denetim bilgileri bu karakter kümesinde olmalıdır.

İleti verilerinin biçimi

Aşağıdakilerden herhangi birini belirtebilirsiniz:

- Uygulama verilerinin biçimi
- Karakter verilerinin karakter kümesi
- Sayısal verilerin biçimi

Bunu yapmak için şu alanları kullanın:

Format
Bu, bir iletinin alıcısına iletideki uygulama verilerinin biçimini gösterir.

Kuyruk yöneticisi bir ileti yarattığında, bazı durumlarda iletinin biçimini tanımlamak için *Format* alanını kullanır. Örneğin, bir kuyruk yöneticisi bir iletiyi teslim edemediğinde, iletiyi teslim edilmeyen (teslim edilmeyen) bir kuyruğa koyar. İletiye bir üstbilgi ekler (daha fazla denetim bilgisi içerir) ve bunu göstermek için *Format* alanını değiştirir.

Kuyruk yöneticisinde, adları MQile başlayan *yerleşik biçimler* vardır; örneğin, MQFMT_STRING. Bunlar gereksinimlerinizi karşılamıyorsa, kendi biçimlerinizi (*kullanıcı tanımlı biçimler*) tanımlayabilirsiniz, ancak bunlar için MQ ile başlayan adları kullanmamalısınız.

Kendi biçimlerinizi yarattığınızda ve kullandığınızda, iletiyi MQGMO_CONVERT kullanarak alan bir programı desteklemek için bir veri dönüştürme çıkışı yazmanız gerekir.

CodedCharSetId
Bu, iletideki karakter verilerinin karakter kümesini tanımlar. Bu karakter kümesini kuyruk yöneticisine ayarlamak istiyorsanız, bu alanı değişmez MQCCSI_Q_MGR ya da MQCSI_INHERIT değerine ayarlayabilirsiniz.

Bir kuyruktan ileti aldığınızda, *CodedCharSetId* alanının değerini uygulamanızın beklediği değerle karşılaştırın. İki değer farklıysa, iletideki herhangi bir karakter verisini dönüştürmeniz ya da varsa bir veri dönüştürme iletisi çıkışı kullanmanız gerekebilir.

Encoding
Bu, ikili tamsayıları, paketlenmiş ondalık tamsayıları ve kayan noktalı sayıları içeren sayısal ileti verilerinin biçimini açıklar. Genellikle kuyruk yöneticisinin çalıştığı makineye göre kodlanır.

Bir iletiyi kuyruğa koyduğunuzda, genellikle *Encoding* alanında değişmez MQENC_NATIVE değerini belirtirsiniz. Bu, ileti verilerinizin kodlanmasının, uygulamanızın çalıştığı makineyle aynı olduğu anlamına gelir.

Bir kuyruktan ileti aldığınızda, ileti tanımlayıcıdaki *Encoding* alanının değerini makinenizdeki değişmez MQENC_NATIVE değeriyle karşılaştırın. İki değer farklıysa, iletideki sayısal verileri dönüştürmeniz ya da varsa bir veri dönüştürme iletisi çıkışı kullanmanız gerekebilir.

Uygulama verilerini dönüştürme

Uygulama verilerinin karakter kümesine ve farklı platformların ilgili olduğu başka bir uygulamanın gerektirdiği kodlamaya dönüştürülmesi gerekebilir.

Gönderen kuyruk yöneticisinde ya da alan kuyruk yöneticisinde dönüştürülebilir. Yerleşik biçimlerin kitaplığı gereksinimlerinizi karşılamıyorsa, kendi biçiminizi tanımlayabilirsiniz. Dönüştürme tipi, ileti tanımlayıcısı MQMD ' nin biçim alanında belirtilen ileti biçimine bağlıdır.

Not: MQFMT_NONE belirtilmiş iletiler dönüştürülmez.

Gönderen kuyruk yöneticisinde dönüştürme

Uygulama verilerini dönüştürmek için gönderen ileti kanalı aracısına (MCA) gereksiniminiz varsa, CONVERT kanal özniteliğini YES değerine ayarlayın.

Dönüştürme, gönderen kuyruk yöneticisinde belirli yerleşik biçimler için ve uygun bir kullanıcı çıkışı sağlanırsa, kullanıcı tanımlı biçimler için gerçekleştirilir.

Yerleşik biçimler

Bunlar arasında aşağıdakiler yer alır:

- Tüm karakterler olan iletiler (MQFMT_STRING biçim adını kullanarak)
- IBM MQ tanımlı iletiler; örneğin, Programlanabilir Komut Biçimleri

  IBM MQ , denetim iletileri ve olayları için Programlanabilir Komut Biçimi iletilerini kullanır (bu durumda kullanılan biçim adı MQFMT_ADMIN 'dir). Kendi iletileriniz için aynı biçimi (MQFMT_PCF biçim adını kullanarak) kullanabilir ve yerleşik veri dönüştürme olanağından yararlanabilirsiniz.

Kuyruk yöneticisi yerleşik biçimlerinin tümünün adları MQFMT ile başlar. Bunlar Biçim' de listelenir ve açıklanır.

Uygulama tanımlı biçimler

Kullanıcı tanımlı biçimler için, uygulama verileri dönüştürme işlemi bir veri dönüştürme çıkış programı tarafından gerçekleştirilmelidir (daha fazla bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941). İstemci-sunucu ortamında, çıkış sunucuya yüklenir ve dönüştürme burada gerçekleşir.

Alıcı kuyruk yöneticisinde dönüştürme

Uygulama ileti verileri, alıcı kuyruk yöneticisi tarafından hem yerleşik hem de kullanıcı tanımlı biçimler için dönüştürülebilir.

MQGMO_CONVERT seçeneğini belirtirseniz, dönüştürme bir MQGET çağrısının işlenmesi sırasında gerçekleştirilir. Ayrıntılar için bkz. Seçenekler

Kodlanmış karakter takımları

IBM MQ ürünleri, temel işletim sistemi tarafından sağlanan kodlanmış karakter kümelerini destekler.

Bir kuyruk yöneticisi yarattığınızda, kullanılan kuyruk yöneticisi kodlanmış karakter takımı tanıtıcısı (CCSID) temeldeki ortama bağlıdır. Bu karma kod sayfasıysa, IBM MQ karma kod sayfasının SBCS bölümünü kuyruk yöneticisi CCSID 'si olarak kullanır.

Genel veri dönüştürme için, temel işletim sistemi DBCS kod sayfalarını destekliyorsa, IBM MQ bunu kullanabilir.

Desteklediği kodlanmış karakter kümelerinin ayrıntıları için işletim sisteminize ilişkin belgelere bakın.

Birden çok platforma yayılan uygulamalar yazarken uygulama verilerini dönüştürme, biçim adlarını ve kullanıcı çıkışlarını göz önünde bulundurmanız gerekir. Veri dönüştürme çıkışlarının çağrılması ve yazılması hakkında bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941 .

İleti öncelikleri

İletinin önceliğini sayısal bir değere ayarlayabilir ya da iletinin kuyruğun varsayılan önceliğini almasını seçebilirsiniz.

İletiyi kuyruğa koyduğunuzda, iletinin önceliğini (MQMD yapısının *Priority* alanında) belirlersiniz. Öncelik için sayısal bir değer belirleyebilir ya da iletinin kuyruğun varsayılan önceliğini almasına izin verebilirsiniz.

Kuyruğun **MsgDeliverySequence** özniteliği, kuyruktaki iletilerin FIFO (ilk giren ilk çıkar) sırasında mı, yoksa FIFO (ilk giren ilk çıkar) sırasında mı saklanacağını belirler. Bu öznitelik MQMDS_PRIORITY değerine ayarlanırsa, iletiler ileti tanımlayıcılarının *Priority* alanında belirtilen öncelikle kuyruğa alınır; ancak MQMDS_FIFO olarak ayarlanırsa, iletiler kuyruğun varsayılan önceliğiyle kuyruğa alınır. Eşit önceliğe sahip iletiler, varış sırasına göre kuyrukta saklanır.

Bir kuyruğun **DefPriority** özniteliği, o kuyruğa konan iletiler için varsayılan öncelik değerini ayarlar. Bu değer, kuyruk yaratıldığında belirlenir, ancak daha sonra değiştirilebilir. Diğer ad kuyrukları ve uzak kuyrukların yerel tanımlamaları, çözümlendikleri temel kuyruklardan farklı varsayılan önceliklere sahip olabilir. Çözüm yolunda birden fazla kuyruk tanımlaması varsa (bkz. “Ad çözümlemesi” sayfa 714), varsayılan öncelik, açık komutta belirtilen kuyruğun **DefPriority** özniteliğinin değerinden (koyma işlemi sırasında) alınır.

Kuyruk yöneticisinin **MaxPriority** özniteliğinin değeri, o kuyruk yöneticisi tarafından işlenen bir iletiye atayabileceğiniz öncelik üst sınırıdır. Bu özniteliğin değerini değiştiremezsiniz. IBM MQiçinde öznitelik 9 değerine sahiptir; 0 (en düşük) ile 9 (en yüksek) arasında önceliklere sahip iletiler oluşturabilirsiniz.

İleti Özellikleri

Bir uygulamanın işlenecek iletileri seçmesini ya da MQMD ya da MQRFH2 üstbilgilerine erişmeden bir iletiyle ilgili bilgileri almasını sağlamak için ileti özelliklerini kullanın. IBM MQ ve JMS uygulamaları arasındaki iletişimi de kolaylaştırır.

İleti özelliği, bir metin adı ve belirli bir tipte bir değerden oluşan bir iletiyle ilişkili verilerdir. İleti özellikleri, ileti seçiciler tarafından yayınları konulara süzmek ya da kuyruklardan seçici olarak ileti almak için kullanılır. İleti özellikleri, iş verilerini ya da durum bilgilerini uygulama verilerinde saklamak zorunda kalmadan içermek için kullanılabilir. Uygulamaların MQ Message Descriptor (MQMD) ya da MQRFH2 üstbilgilerindeki verilere erişmeleri gerekmez; bu veri yapılarındaki alanlara, İleti Kuyruğu Arabirimi (MQI) işlev çağrılarını kullanarak ileti özellikleri olarak erişilebilir.

IBM MQ içindeki ileti özelliklerinin kullanımı, JMSiçindeki özelliklerin kullanımını taklit eder. Bu, bir JMS uygulamasında özellikleri ayarlayabileceğiniz ve bunları bir yordam IBM MQ uygulamasında ya da diğer bir yöntemle alabileceğiniz anlamına gelir. Bir özelliği JMS uygulamasının kullanımına sunmak için "usr" önekini atayın; daha sonra JMS ileti kullanıcı özelliği olarak kullanılabilir (önek olmadan). Örneğin, IBM MQ özelliği *usr.myproperty* (karakter dizgisi), JMS call `message.getStringProperty('myproperty')` kullanan bir JMS uygulaması tarafından erişilebilir. İki ya da daha fazla U+002E (".") içeriyorsa, JMS uygulamalarının "usr" önekine sahip özelliklere erişemediğini unutmayın. karakterler. Öneki olmayan ve U+002E (".") olmayan bir özellik karakter "usr" önekine sahip gibi işlenir. Tersi durumda, JMS uygulamasında ayarlanan bir kullanıcı özelliğine IBM MQ uygulamasında "usr" eklenerek erişilebilir. bir MQINQMP çağrısında sorulmuş özellik adının öneki.

İleti özellikleri ve ileti uzunluğu

Bir IBM MQ kuyruk yöneticisindeki herhangi bir iletiyle birlikte akabilecek özelliklerin boyutunu denetlemek için *MaxPropertiesLength* kuyruk yöneticisi özniteliğini kullanın.

Genel olarak, özellikleri ayarlamak için MQSETMP kullandığınızda, bir özelliğin büyüklüğü özellik adının byte olarak uzunluğunun yanı sıra, MQSETMP çağrısına geçirilen byte cinsinden özellik değerinin uzunluğudur. Özellik adının karakter takımı ve özellik değeri, iletinin hedefe iletilmesi sırasında değişebilir, çünkü bunlar Unicode 'a dönüştürülebilir; bu durumda özellik boyutu değişebilir.

Bir MQPUT ya da MQPUT1 çağrısında, iletinin özellikleri kuyruk ve kuyruk yöneticisine ilişkin iletinin uzunluğunu hesaba katmaz, ancak kuyruk yöneticisi tarafından algılanan özelliklerin uzunluğuna (MQI çağrıları kullanılarak ayarlanıp ayarlanmadıklarına bakılmaksızın) göre sayılır.

Özelliklerin boyutu özellik uzunluğu üst sınırını aşarsa, ileti MQRC_19TIES_TOO_BIG ile reddedilir. Özelliklerin boyutu gösterimine bağlı olduğundan, maksimum özellik uzunluğunu brüt düzeyde ayarlamanız gerekir.

Arabellek özellikler içeriyorsa, bir uygulamanın arabelleği *MaxMsgLength*değerinden büyük olan bir iletiyi başarıyla yerleştirmesi mümkündür. Bunun nedeni, MQRFH2 öğeleri olarak gösterildiğinde bile, ileti özelliklerinin iletinin uzunluğuna doğru sayılmamesidir. MQRFH2 üstbilgi alanları özellik uzunluğuna yalnızca bir ya da daha çok klasör varsa ve üstbilgideki her klasör özellikler içeriyorsa eklenir. MQRFH2 üstbilgisinde bir ya da daha fazla klasör varsa ve herhangi bir klasör özellikler içermiyorsa, MQRFH2 üstbilgi alanları ileti uzunluğuna doğru sayılır.

Bir MQGET çağrısında, iletinin özellikleri, kuyruk ve kuyruk yöneticisi ilgili olduğu sürece iletinin uzunluğunu hesaba katmaz. Ancak, özellikler ayrı olarak sayıldığından, bir MQGET çağrısının döndürdüğü arabellek *MaxMsgLength* özniteliğinin değerinden büyük olabilir.

Uygulamalarınızın *MaxMsgLength* değerini sorgulamasına izin vermeyin ve MQGET çağrılmadan önce bu büyüklükte bir arabellek ayırın; bunun yerine, yeterince büyük olduğunu düşündüğünüz bir arabellek ayırın. MQGET başarısız olursa, *DataLength* parametresinin büyüklüğüne göre yönlendirilen bir arabellek ayırın.

MQGET çağrısının *DataLength* değiştirgesi, MQGMO yapısında bir ileti tanıtıcısı belirtilmediyse, uygulama verilerinin ve sağladığınız arabellekte döndürülen özelliklerin byte cinsinden uzunluğunu döndürür.

MQPUT çağrısının *Arabellek* değiştirgesi, gönderilecek uygulama iletisi verilerini ve ileti verilerinde gösterilen özellikleri içerir.

Her ileti için ileti tanımlayıcısı ya da uzantısı dışında, ileti özellikleri için uzunluk sınırı 100 MB 'dir.

İç gösterimindeki bir özelliğin boyutu, adın uzunluğu artı değerinin boyutu ve özelliğe ilişkin bazı denetim verileridir. İletiye bir özellik eklendikten sonra özellikler kümesi için bazı denetim verileri de vardır.

Özellik adları

Özellik adı bir karakter dizgisidir. Belirli kısıtlamalar, uzunluğu ve kullanılabilecek karakter kümesi için geçerlidir.

Özellik adı, büyük ve küçük harfe duyarlı bir karakter dizgisidir; bağlam tarafından tersi belirtilmedikçe +4095 karakterle sınırlıdır. Bu sınır, MQ_MAX_PROPERTY_NAME_LENGTH değişmezinde bulunur.

Bir ileti özelliği MQI çağrısı kullanırken bu uzunluk üst sınırını aşarsanız, çağrı başarısız olur; neden kodu MQRC_NAME_LENGTH_ERR.

JMSiçinde özellik adı uzunluğu üst sınırı olmadığından, bir JMS uygulamasının MQRFH2 yapısında saklandığında geçerli bir IBM MQ özellik adı olmayan geçerli bir JMS özellik adı ayarlayabilir.

Bu durumda, ayrıştırıldığında özellik adının yalnızca ilk 4095 karakteri kullanılır; aşağıdaki karakterler kesilir. Bu, seçicileri kullanan bir uygulamanın bir seçim dizgisiyle eşleşmemesine ya da beklenmediğinde bir dizgiyle eşleşmemesine neden olabilir; çünkü birden çok özellik aynı adla kesilebilir. Bir özellik adı kesildiğinde, WebSphereMQ bir hata günlüğü iletisi verir.

Tüm özellik adları, adın bir parçası olarak U+002E (.) Unicode karakterine izin verilmemesi dışında, Java Tanıtıcıları için Java Dil Belirtimi tarafından tanımlanan kurallara uymalıdır. Java Tanıtıcıları ile ilgili kurallar, özellik adları için JMS belirtiminde bulunanlara eşittir.

Beyaz alan karakterleri ve karşılaştırma işleçleri yasaklanmıştır. Bir özellik adında gömülü boş değer kullanılmasına izin verilir, ancak önerilmez. Gömülü boş değerler kullanırsanız, değişken uzunluklu dizgileri belirtmek için MQCHARV yapısıyla kullanıldığında MQVS_NULL_TERMINATED değişmezinin kullanılmasını önler.

Uygulamalar özellik adlarına dayalı olarak ileti seçebildiği için özellik adlarını basit tutun ve seçici ile adın karakter kümesi arasındaki dönüştürme, seçimin beklenmeyen bir şekilde başarısız olmasına neden olabilir.

IBM MQ özellik adları, özelliklerin mantıksal gruplaması için U+002E (.) karakterini kullanır. Bu, özellikler için ad alanını böler. Aşağıdaki önekleri içeren özellikler, küçük ya da büyük harflerin herhangi bir karışımında ürün tarafından kullanılmak üzere ayrılmıştır:

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

Ad çakışmalarını önlemenin iyi bir yolu, tüm uygulamaların ileti özelliklerine Internet etki alanı adlarıyla önek eklemesini sağlamaktır. Örneğin, `ourcompany.com` etki alanı adını kullanarak bir uygulama geliştiriyorsanız, tüm özellikleri `com.ourcompany`önekiyle adarabilirsiniz. Bu adlandırma kuralı, özelliklerin kolay seçilmesini de sağlar; örneğin, bir uygulama `com.ourcompany.%`ile başlayan tüm ileti özelliklerini sorgulayabilir.

Özellik adlarının kullanımıyla ilgili ek bilgi için Özellik adı kısıtlamaları konusuna bakın.

Özellik adı kısıtlamaları
Bir özelliği adlandırdığınızda, belirli kurallara uymanız gerekir.

Özellik adları için aşağıdaki kısıtlamalar geçerlidir:

1. Bir özellik şu dizgilerle başlamamalıdır:
   - "JMS"- IBM MQ classes for JMStarafından kullanılmak üzere ayrılmıştır.
   - "usr.JMS"-geçerli değil.

   Tek kural dışı durumlar, JMS özellikleri için eşanlamlılar sağlayan aşağıdaki özelliklerdir:

Özellik	**Eşanlamlısı:**
JMSCorrelationID	Kök.MQMD.CorrelId ya da jms.Cid

Özellik	Eşanlamlısı:
JMSDeliveryMode	Kök.MQMD.Persistence ya da jms.Dlv
JMSDestination (JMS Hedefi)	jms.Dst
JMSExpiration	Kök.MQMD.Expiry ya da jms.Exp
JMSMessageID	Kök.MQMD.MsgId
JMS Önceliği	Kök.MQMD.Priority ya da jms.Pri
Yeniden Teslim edilen JMS	Kök.MQMD.BackoutCount (Geri Sayım)
JMSReplyTo (URI olarak kodlanmış bir dizgi)	Kök.MQMD.ReplyToQ ya da Root.MQMD.ReplyToQMgr ya da jms.Rto
JMSTimestamp	Kök.MQMD.PutDate ya da Root.MQMD.PutTime ya da jms.Tms
JMSType	mcd.Type ya da mcd.Set ya da mcd.Fmt yazın
JMSXAppID	Kök.MQMD.PutApplName
JMSXDeliveryCount	Kök.MQMD.BackoutCount (Geri Sayım)
JMSXGroupID	Kök.MQMD.GroupId ya da jms.Gid
JMSXGroupSeq	Kök.MQMD.MsgSeqNumber ya da jms.Seq
JMSXUserID	Kök.MQMD.UserIdentifier

Bu eşanlamlılar, bir MQI uygulamasının JMS özelliklerine IBM MQ classes for JMS istemci uygulamasına benzer şekilde erişmesini sağlar. Bu özelliklerden yalnızca JMSCorrelationID, JMSReplyTo, JMSType, JMSXGroupIDve JMSXGroupSeq MQI kullanılarak ayarlanabilir.

IBM MQ classes for JMS içinden kullanılabilen JMS_IBM_ * özelliklerinin MQI kullanılarak kullanılamadığına dikkat edin. JMS_IBM_ * özellik başvurusunda bulunulan alanlara MQI uygulamaları tarafından başka yollardan erişilebilir.

2. BIR öZELLIK, "NULL", "TRUE", "FALSE", "NOT", "AND", "OR", "BETWEEN", "LIKE", "IN", "IS" VE "ESCAPE" KARIŞIMLARINDA ÇAĞRILMAMALIDIR. Bunlar, seçim dizgilerinde kullanılan SQL anahtar sözcüklerinin adlarıdır.
3. Bir özellik adı başlangıcı " mq " "mq_usr" başındaki herhangi bir küçük harf ya da büyük harf karışımında yalnızca bir "içerebilir." karakter (U+002E). Birden çok "." bu önekleri içeren özelliklerde karakterlere izin verilmez.
4. İki "." karakterler arasında başka karakterler içermelidir; sıradüzende boş bir noktanızolamaz. Benzer şekilde, bir özellik adı "." ile bitemez. karakter.
5. Bir uygulama "a.b" özelliğini ve daha sonra "a.b.c" özelliğini ayarlarsa, "b" sıradüzeninde bir değer mi yoksa başka bir mantıksal gruplama mı içerdiği belirsizdir. Böyle bir sıradüzen "karışık içerik" dir ve bu desteklenmez. Karma içeriğe neden olan bir özellik ayarlanmasına izin verilmez.

Bu kısıtlamalar, aşağıdaki gibi doğrulama mekanizması tarafından uygulanır:

- İleti tanıtıcısı yaratıldığında geçerlilik denetimi istendiyse, MQSETMP-İleti özelliğini ayarla çağrısı kullanılarak bir özellik ayarlanırken özellik adları doğrulanır. Bir özelliğin geçerliliğini denetleme girişimi özellik adı belirtimindeki bir hata nedeniyle başarısız olursa, tamamlanma kodu şu nedenden ötürü MQCC_FAILED olur:
  - MQRC_19TY_NAME_ERROR, nedenler için 1-4.
  - Neden 5 için MQRC_MIXED_CONTENT_NOT_ALLOWED.
- Doğrudan MQRFH2 öğeleri olarak belirtilen özelliklerin adlarının MQPUT çağrısıyla doğrulanacağı garanti edilmez.

Özellik olarak ileti tanımlayıcı alanları
Çoğu ileti tanımlayıcı alanı özellik olarak kabul edilebilir. Özellik adı, ileti tanımlayıcı alanının adına bir önek eklenerek oluşturulur.

Bir MQI uygulaması, bir ileti tanımlayıcı alanında (örneğin, bir seçici dizgisinde ya da ileti özelliği API ' larını kullanarak) bulunan bir ileti özelliğini tanımlamak isterse, aşağıdaki sözdizimini kullanın:

Özellik adı	İleti tanımlayıcı alanı
Root.MQMD.*Alan*	*Alan*

C dili bildirimindeki MQMD yapı alanlarıyla aynı büyük/küçük harf kullanımıyla *Field* belirtin. Örneğin, `Root.MQMD.AccountingToken` özellik adı, ileti tanımlayıcısının `AccountingToken` alanına erişir.

İleti tanımlayıcısının `StrucId` ve `Version` alanlarına, gösterilen sözdizimi kullanılarak erişilemez.

İleti tanımlayıcı alanları, diğer özellikler için olduğu gibi hiçbir zaman MQRFH2 üstbilgisinde gösterilmez.

İleti verileri, kuyruk yöneticisi tarafından onurlandırılan bir MQMDE ile başlıyorsa, MQMDE alanlarına, açıklanan `Root.MQMD.`*`Field`* gösterimi kullanılarak erişilebilir. Bu durumda MQMDE alanları, özellikler perspektifinden MQMD ' nin mantıksal bir parçası olarak işlenir. Bkz. MQMDE ' ye Genel Bakış.

Özellik veri tipleri ve değerleri
Bir özellik bir boole, bir byte dizgisi, bir karakter dizgisi ya da bir kayan nokta ya da tamsayı olabilir. Özellik, bağlam tarafından tersi belirtilmedikçe, veri tipi aralığında geçerli herhangi bir değeri saklayabilir.

Bir özellik değerinin veri tipi aşağıdaki değerlerden biri olmalıdır:

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

Bir özellik var olabilir, ancak tanımlı değeri yoktur; boş değerli bir özelliktir. Boş değerli bir özellik, sıfır uzunluklu değeri olan, tanımlı ancak boş bir değeri olması için bir byte özelliğinden (MQBYTE []) ya da karakter dizgisi özelliğinden (MQCHAR []) farklıdır.

Bayt dizgisi, JMS ya da XMSiçinde geçerli bir özellik veri tipi değil. *usr* klasöründe bayt dizgisi özelliklerini kullanmamanız önerilir.

Kuyruklardan ileti seçilmesi

Bir MQGET çağrısında MsgId ve CorrelId alanlarını kullanarak ya da bir MQOPEN ya da MQSUB çağrısında SelectionString kullanarak kuyruklardan ileti seçebilirsiniz.

Seçiciler
İleti seçici, bir uygulama tarafından yalnızca seçim dizgisinin temsil ettiği Yapılandırılmış Sorgu Dili (SQL) sorgusuna uyan özelliklere sahip iletilere ilgisini kaydetmek için kullanılan değişken uzunluklu bir dizedir.

MQSUB ve MQOPEN işlev çağrılarını kullanarak seçim

MQSUB ve MQOPEN çağrılarını kullanarak seçim yapmak için MQCHARV tipinde bir yapı olan *`SelectionString`*kullanılır.

SelectionString yapısı, kuyruk yöneticisine değişken uzunluklu bir seçim dizgisi geçirmek için kullanılır.

Seçici dizgisiyle ilişkilendirilen CCSID, MQCHARV yapısının VSCCSID alanıyla ayarlanır. Kullanılan değer, seçici dizgileri için desteklenen bir CCSID olmalıdır. Desteklenen kod sayfalarının listesi için Kod Sayfası Dönüştürme konusuna bakın.

IBM MQ destekli Unicode dönüştürmesi olmayan bir CCSID belirlenmesi, MQRC_SOURCE_CCSID_ERROR hatasıyla sonuçlanır. Bu hata, seçici kuyruk yöneticisine sunulduğunda (MQSUB, MQOPEN ya da MQPUT1 çağrısında) döndürülür.

VSCCSID alanının varsayılan değeri, seçim dizgisinin CCSID 'sinin kuyruk yöneticisi CCSID 'sine eşit olduğunu gösteren MQCCSI_APPL 'dir ya da istemci aracılığıyla bağlıysa istemci CCSID 'dir. MQCCSI_APPL değişmezi, derlemeden önce yeniden tanımlayan bir uygulama tarafından geçersiz kılınabilir.

MQCHARV seçicisi boş değerli bir dizgiyi gösteriyorsa, o ileti tüketicisi için seçim olmaz ve iletiler seçici kullanılmamış gibi teslim edilir.

Bir seçim dizgisinin uzunluk üst sınırı yalnızca *VSLength*MQCHARV alanıyla tanımlanabilecek değerlerle sınırlıdır.

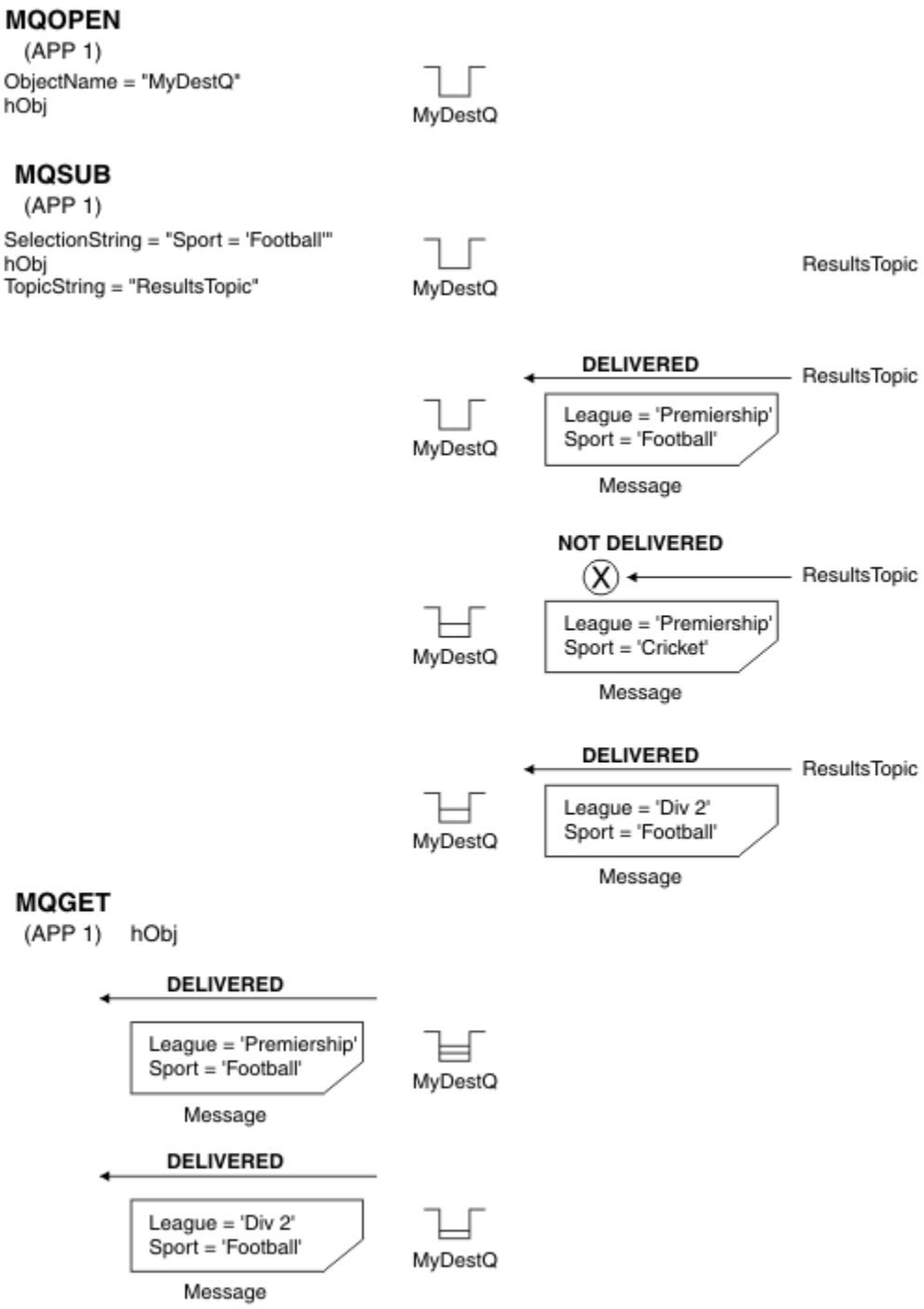
Bir arabellek belirtdiyseniz ve VSBufSizeiçinde pozitif bir arabellek uzunluğu varsa, MQSO_RESUME abonelik seçeneği kullanılarak bir MQSUB çağrısının çıkışında SelectionString döndürülür. Arabellek belirtmezseniz, MQCHARV ' nin VSLkalınlık alanında yalnızca seçim dizgisinin uzunluğu döndürülür. Sağlanan arabellek alanı döndürmek için gereken alandan küçükse, sağlanan arabellekte yalnızca VSBufSize byte döndürülür.

Bir uygulama, önce kuyruk tanıtıcısı (MQOPEN için) ya da abonelik (MQSUB için) kapatılmadan seçim dizgisini değiştiremez. Daha sonra, sonraki bir MQOPEN ya da MQSUB çağrısında yeni bir seçim dizgisi belirtilebilir.

MQOPEN
Açılan tanıtıcıyı kapatmak için MQCLOSE kullanın ve sonraki bir MQOPEN çağrısında yeni bir seçim dizgisi belirtin.

MQSUB
Döndürülen abonelik tanıtıcısını (hSub) kapatmak için MQCLOSE komutunu kullanın ve sonraki bir MQSUB çağrısında yeni bir seçim dizgisi belirtin.

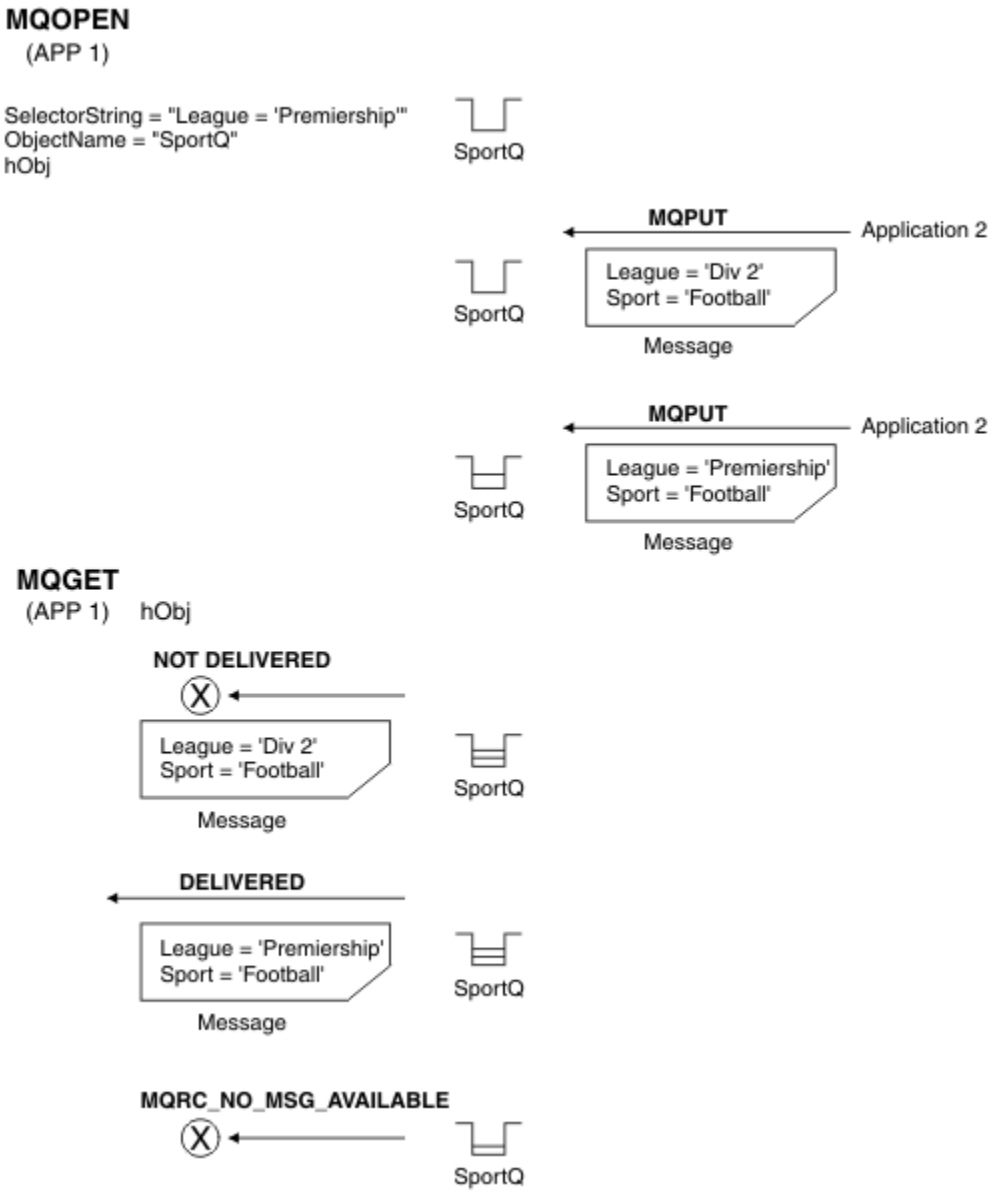
Şekil 3 sayfa 31 içinde MQSUB çağrısının kullanıldığı seçim süreci gösterilir.



Şekil 3. MQSUB çağrısı kullanılarak seçim

MQSD yapısındaki *SelectionString* alanı kullanılarak MQSUB çağrısına bir seçici iletilebilir. Bir seçiciyi MQSUB üzerinde geçirmenin etkisi, hedef kuyrukta yalnızca, abone olunan konuya (sağlanan bir seçim dizgisiyle eşleşen) yayınlanan iletilerin kullanılabilir olmasıdır.

Şekil 4 sayfa 32 içinde, MQOPEN çağrısının kullanıldığı seçim süreci gösterilir.



Şekil 4. MQOPEN çağrısı kullanılarak seçim

MQOD yapısındaki *SelectionString* alanı kullanılarak MQOPEN çağrısına bir seçici iletilebilir. Bir seçiciyi MQOPEN çağrısına geçirmenin etkisi, ileti tüketicisine yalnızca açık kuyruktaki, seçiciyle eşleşen iletilerin teslim edilmesiyle olur.

MQOPEN çağrısındaki seçicinin ana kullanımı, bir uygulamanın yalnızca bir seçiciyle eşleşen bir kuyruktaki iletileri almayı seçebildiği noktadan noktaya iletişim durumu içindir. Önceki örnek, iki iletinin MQOPEN tarafından açılan bir kuyruğa konduğu, ancak bir seçiciyle eşleşen tek ileti olduğu için alan uygulama tarafından yalnızca birinin alındığı basit bir senaryoyu gösterir.

Kuyrukta, verili seçiciyle eşleşen başka ileti olmadığından, sonraki MQGET çağrıları MQRC_NO_MSG_AVAILABLE ile sonuçlanır.

İlgili kavramlar

"Seçim dizgisi kuralları ve kısıtlamaları" sayfa 39
Seçicileri kullanırken olası sorunları önlemek için seçim dizgilerinin nasıl yorumlandığına ve karakter kısıtlamalarına ilişkin bu kuralları tanıyın.

Seçim davranışı

IBM MQ seçim davranışına genel bakış.

MQMDE yapısındaki alanlar, MQMD ise, ilgili ileti tanımlayıcısı özelliklerinin ileti özellikleri olarak kabul edilir:

- MQFMT_MD_EXTENSION biçimine sahip
- Hemen ardından geçerli bir MQMDE yapısı gelir
- Sürüm bir mi, yoksa yalnızca varsayılan sürüm iki alanlarını mı içeriyor?

Bir seçim dizgisinin, ileti özellikleriyle eşleşen herhangi bir özellik gerçekleşmeden önce TRUE ya da FALSE değerine çözülmesi mümkündür. Örneğin, seçim dizgisi `"TRUE <> FALSE"`olarak ayarlanırsa bu durum söz konusu olabilir. Bu tür bir erken değerlendirmenin yalnızca seçim dizgisinde ileti özelliği başvurusu olmadığında gerçekleşmesi garanti edilir.

Bir seçim dizgisi, herhangi bir ileti özelliği dikkate alınmadan önce TRUE olarak çözümlenirse, tüketici tarafından abone olunan konuya yayınlanan tüm iletiler teslim edilir. Herhangi bir ileti özelliği dikkate alınmadan önce bir seçim dizgisi FALSE olarak çözülürse, seçiciyi sunan işlev çağrısında MQRC_SELECTOR_ALWAYS_FALSE neden kodu ve tamamlama kodu MQCC_FAILED döndürülür.

Bir ileti herhangi bir ileti özelliği içermese bile (üstbilgi özellikleri dışında), seçim için uygun olabilir. Bir seçim dizgisi var olmayan bir ileti özelliğine başvururşa, bu özelliğin NULL ya da 'Unknown ' değerine sahip olduğu varsayılır.

Örneğin, bir ileti `'Color IS NULL'` gibi bir seçim dizgisini karşılamaya devam edebilir; burada `'Color'` , iletide ileti özelliği olarak bulunmaz.

Genişletilmiş bir ileti seçimi sağlayıcısı olmadığı sürece, seçim yalnızca iletinin kendisiyle değil, bir iletiyle ilişkilendirilmiş özelliklerde gerçekleştirilebilir. Seçim, ileti bilgi yükü üzerinde yalnızca genişletilmiş bir ileti seçimi sağlayıcısı varsa gerçekleştirilebilir.

Her ileti özelliğiyle ilişkilendirilmiş bir tip vardır. Bir seçim gerçekleştirdiğinizde, ileti özelliklerini sınamak için ifadelerde kullanılan değerlerin doğru tipte olduğundan emin olmanız gerekir. Tip uyuşmazlığı oluşursa, söz konusu ifade FALSE olarak çözülür.

Seçim dizgisi ve ileti özelliklerinin uyumlu tipleri kullanmasını sağlamak sizin sorumluluğunuzdadır.

Seçim ölçütleri etkin olmayan sürekli aboneler adına uygulanmaya devam eder, bu nedenle yalnızca başlangıçta sağlanan seçim dizgisiyle eşleşen iletiler alıkonur.

Kalıcı bir abonelik alter (MQSO_ALTER) ile sürdürüldüğünde seçim dizgileri değiştirilemez. Sürekli bir abone etkinliği sürdürdüğünde farklı bir seçim dizgisi sunulursa, uygulamaya MQRC_SELECTOR_NOT_ALTERABLE döndürülür.

Bir kuyrukta seçim ölçütlerine uyan bir ileti yoksa, uygulamalar MQRC_NO_MSG_AVAILABLE dönüş kodunu alır.

Bir uygulama özellik değerlerini içeren bir seçim dizgisi belirttiyse, yalnızca eşleşen özellikleri içeren iletiler seçilebilir. Örneğin, bir abone "a = 3" seçim dizgisini belirtir ve 'a' nın var olmadığı ya da 3 'e eşit olmadığı özellikler içeren bir ileti yayınlanır. Abone bu iletiyi hedef kuyruğuna almıyor.

İleti sistemi performansı

Kuyruktan ileti seçilmesi, IBM MQ ' in kuyruktaki her iletiyi sırayla incelemesini gerektirir. Seçim ölçütleriyle eşleşen bir ileti bulununcaya ya da incelenecek başka ileti kalmayıncaya kadar iletiler incelenir. Bu nedenle, ileti seçimi derin kuyruklarda kullanılırsa ileti alışverişi başarımı zarar görür.

Seçim JMSCorrelationID ya da JMSMessageIDtemelinde olduğunda derin kuyruklarda ileti seçimini eniyilemek için formun seçim dizgisini kullanın:

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

Burada:

- *correlation_id* , standart bir IBM MQ ilinti tanıtıcısı içeren bir dizgidir.
- *message_id* , standart IBM MQ ileti tanıtıcısını içeren bir Dizgidir.

Not: Seçici, özelliklerden yalnızca birine başvurmalıdır. Bu biçimlerden birine sahip bir seçicinin kullanılması, JMSCorrelationID üzerinde seçim yaparken performansta önemli bir artış sunar ve JMSMessageIDiçin marjinal bir performans iyileştirmesi sunar. Daha fazla bilgi için bkz "JMS içindeki ileti seçiciler" sayfa 139.

Karmaşık seçicilerin kullanılması

Seçiciler birçok bileşen içerebilir, örneğin:

a ve b ya da c ve d ya da e ve f ya da g ve h ya da i ve j... ya da y ve z

Bu tür karmaşık seçicilerin kullanımı ciddi performans etkileri ve aşırı kaynak gereksinimleri olabilir. Bu nedenle IBM MQ , bir sistem kaynağı eksikliğiyle sonuçlanbilecek aşırı karmaşık seçicileri işlemeyi başaramayarak sistemi korur. Koruma, 100 'den fazla sınama içeren seçim dizgilerinde ya da IBM MQ , işletim sistemi yığınının boyutuna ilişkin sınırın yaklaştığını saptadığında oluşabilir. Koruma sınırlarına ulaşılmadığından emin olmak için, uygun platformlarda, birçok bileşenle birlikte seçim dizgilerinin kullanımını iyice denemeniz ve sınamanız gerekir.

Seçicilerin performansı ve karmaşıklığı, bileşenleri birleştirmek için ek parantezler kullanılarak basitleştirilerek geliştirilebilir. Örneğin:

(a ve b ya da c ve d) ya da (e ve f ya da g ve h) ya da (i ve j) ...

İlgili kavramlar

"Seçim dizgisi kuralları ve kısıtlamaları" sayfa 39
Seçicileri kullanırken olası sorunları önlemek için seçim dizgilerinin nasıl yorumlandığına ve karakter kısıtlamalarına ilişkin bu kuralları tanıyın.

İleti seçici sözdizimi

IBM MQ ileti seçicisi, SQL92 koşullu ifade sözdiziminin bir alt kümesine dayalı sözdizimine sahip bir dizedir.

Bir ileti seçicinin değerlendirildiği sıra, öncelik düzeyinde soldan sağa doğrudur. Bu sırayı değiştirmek için parantezleri kullanabilirsiniz. Önceden tanımlanmış seçici hazır bilgileri ve işleç adları burada büyük harfle yazılır; ancak, bunlar büyük/küçük harfe duyarlı değildir.

Seçici API aracılığıyla sağlanırsa, IBM MQ ileti seçicinin sunulduğu sırada sözdizimsel doğruluğunu doğrular. Seçim dizgisinin sözdizimi yanlışsa ya da bir özellik adı geçerli değilse ve genişletilmiş ileti seçimi sağlayıcısı yoksa, uygulamaya MQRC_SELECTION_NOT_AVAILABLE döndürülür. Seçim dizgisinin sözdizimi yanlışsa ya da abonelik sürdürüldüğünde bir özellik adı geçerli değilse, uygulamaya bir MQRC_SELECTOR_SYNTAX_ERROR döndürülür. Özellik ayarlandığında (MQCMHO_VALIDATEyerine MQCMHO_NONE ayarlanarak) özellik adı geçerlilik denetimi geçersiz kılındıysa ve bir uygulama daha sonra geçersiz özellik adına bir ileti yerleştirirse, bu ileti hiçbir zaman seçilmez.

IBM MQ , yönetici tarafından tanımlanan bir abonelik seçicinin EXTENDEDdeğerine sahip **DISPLAY SUB** değiştirgesiyle **SELTYPE** belirtildiği gibi genişletilmiş ileti sözdizimini kullandığını belirlerse, seçici sunulduğunda herhangi bir hata döndürülmez. Bu durumda, seçim dizgisinin sözdiziminin denetlenmesi yayınlama zamanına kadar ertelenir (bkz. MQRC_SELECTION_NOT_AVAILABLE).

Bir seçici şunları içerebilir:

- Hazır bilgiler:
  - Dizgi hazır bilgileri tek tırnak içine alınır. İki ardışık tek tırnak işareti tek tırnak işaretini temsil eder. Örnekler: 'literal' ve 'literal'. Java dizgi hazır bilgileri gibi, bunlar da Unicode karakter kodlamasını kullanır. Bir dizgi hazır bilgisini çevrelemek için çift tırnak işareti kullanamazsınız. Tek tırnak işaretleri arasında herhangi bir bayt sırası kullanılabilir.
  - Bayt dizgisi, çift tırnak içine alınmış ve öneki 0xolan bir ya da daha çok onaltılı karakter çiftidir. Örnekler: "0x2F1C" ya da "0XD43A". Bayt diziliminin uzunluğu en az bir bayt olmalıdır. Seçici byte

dizgisi MQTYPE_BYTE_STRINGtipinde bir ileti özelliğiyle eşleşirse, baştaki ya da sondaki sıfır için özel bir işlem gerçekleştirilmez. Baytlar başka bir karakter olarak işlenir. Endianness de dikkate alınmaz. Seçici ve özellik byte dizgilerinin uzunluğu eşit olmalı ve byte sırası aynı olmalıdır.

Eşleşen bayt dizilimi seçimleri örnekleri (*myBytes* = 0AFC23):

- "myBytes = "0x0AFC23"" = TRUE

Aşağıdaki dizgi seçimleri eşleşmiyor:

- "myBytes = "0xAFC23"" = MQRC_SELECTOR_SYNTAX_ERROR (bayt sayısı ikinin katı olmadığı için)
- "myBytes = "0x0AFC2300"" = FALSE (karşılaştırmada sondaki sıfır önemli olduğundan)
- "myBytes = "0x000AFC23"" = FALSE (karşılaştırmada baştaki sıfır önemli olduğu için)
- "myBytes = "0x23FC0A"" = FALSE (endianness dikkate alınmadığı için)

– Onaltılı sayılar sıfır ile başlar, ardından büyük ya da küçük xgelir. Hazır bilginin geri kalanı bir ya da daha fazla geçerli onaltılı karakter içeriyor. Örnekler: 0xA, 0xAF, 0X2020.

– 0-7 aralığındaki bir ya da daha fazla basamağın izlediği baştaki bir sıfır, her zaman sekizli sayının başlangıcı olarak yorumlanır. Bunun gibi sıfır önekli bir ondalık sayıyı gösteremezsiniz; örneğin, 9 geçerli bir sekizli sayı olmadığı için 09 bir sözdizimi hatası döndürür. Sekizli sayı örnekleri: 0177, 0713.

– Tam sayısal hazır bilgi, 57, -957ve +62gibi ondalık noktası olmayan sayısal bir değerdir. Tam sayısal bir hazır bilginin sonunda büyük ya da küçük harf L olabilir; bu, sayının nasıl saklandığını ya da yorumlandığını etkilemez. IBM MQ , -9,223,372,036,854,775,808 - 9,223,372,036,854,775,807aralığındaki tam sayıları destekler.

– Yaklaşık sayısal hazır bilgi, 7E3 ya da -57.9E2gibi bilimsel gösterimde sayısal bir değer ya da 7., -95.7ya da +6.2gibi ondalık içeren bir sayısal değerdir. IBM MQ , -1.797693134862315E+308 - 1.797693134862315E+308aralığındaki sayıları destekler.

Anlamlı, isteğe bağlı bir işaret karakterinden sonra gelmelidir (+ ya da -). Anlamlı değer bir tamsayı ya da bir kesir olmalıdır. Anlamsal bir parçanın başında bir sayı olması gerekmez.

Büyük ya da küçük E , isteğe bağlı bir üstel başlangıcını gösterir. Üstel bir ondalık taban içeriyor ve üstel parçasının başına isteğe bağlı bir işaret karakteri konabilir.

Yaklaşık sayısal hazır bilgiler bir F ya da D karakteriyle (büyük ve küçük harfe duyarlı değil) sonlandırılabilir. Bu sözdizimi, tek ya da çift duyarlıklı sayıları etiketlemek için kullanılan diller arası yöntemi destekler. Bu karakterler isteğe bağlıdır ve yaklaşık bir sayısal hazır bilginin nasıl saklanacağını ya da işleneceğini etkilemez. Bu sayılar her zaman çift duyarlıklı olarak saklanır ve işlenir.

– Boole hazır bilgileri TRUE ve FALSE.

Not: Sonlu olmayan IEEE-754 gösterimleri (NaN, +Infinity, -Infinity gibi) seçim dizgilerinde desteklenmez. Bu nedenle, bu değerler bir ifadede işlenen olarak kullanılamaz. Negatif sıfır, matematiksel işlemler için pozitif sıfır olarak değerlendirilir.

- Tanıtıcılar:

Tanıtıcı, geçerli bir tanıtıcı başlangıç karakteriyle başlaması ve ardından sıfır ya da daha fazla geçerli tanıtıcı kısım karakteriyle başlaması gereken değişken uzunluklu bir karakter dizisidir. Tanıtıcı adlarına ilişkin kurallar, ileti özelliği adlarına ilişkin kurallarla aynıdır, ek bilgi için bkz. “Özellik adları” sayfa 26 ve “Özellik adı kısıtlamaları” sayfa 27 .

Not: Seçim, ileti bilgi yükü üzerinde yalnızca genişletilmiş bir ileti seçimi sağlayıcısı varsa gerçekleştirilebilir.

Tanıtıcılar, üstbilgi alanı başvuruları ya da özellik başvurulardır. Bir ileti seçicindeki bir özellik değerinin tipi, özelliği ayarlamak için kullanılan tipe karşılık gelmelidir; ancak, mümkün olduğu yerde sayısal yükseltme gerçekleştirilir. Bir tip uyuşmazlığı oluşursa, ifadenin sonucu FALSEolur. Bir iletide var olmayan bir özelliğe gönderme yapılırsa, bu özelliğin değeri NULLolur.

Özellikler için alma yöntemleri için geçerli olan tip dönüştürmeleri, bir özellik ileti seçici ifadesinde kullanıldığında geçerli değildir. Örneğin, bir özelliği dizgi değeri olarak ayarlarsanız ve sonra sayısal değer olarak sorgulamak için bir seçici kullanırsanız, ifade FALSEdeğerini döndürür.

Özellik adlarıyla ya da MQMD alan adlarıyla eşleştirilen JMS alan ve özellik adları, seçim dizgisindeki geçerli tanıtıcılardır. IBM MQ , tanınan JMS alanı ve özellik adlarını ileti özelliği değerleriyle eşler. Ek bilgi için bkz. “JMS içindeki ileti seçiciler” sayfa 139 . Örneğin, `"JMSPriority >="` seçim dizgisi, yürürlükteki iletinin `jms` klasöründe bulunan `Pri` özelliğinde seçim yapar.

- Taşma/taşma:

  Hem ondalık hem de yaklaşık sayısal sayılar için aşağıdaki koşullar tanımlanmamıştır:

  – Tanımlı aralığın dışında bir sayı belirtme
  – Taşmaya ya da taşmaya neden olacak bir aritmetik ifade belirtilmesi

  Bu koşullar için denetim gerçekleştirilmez.

- Beyaz alan:

  Boşluk, form besleme, yeni satır, satır başı, yatay sekme ya da dikey sekme olarak tanımlanır. Aşağıdaki Unicode karakterleri beyaz alan olarak tanınır:

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

- İfadeler:

  – Seçici, koşullu bir ifadedir. Doğru eşleşmeleri değerlendiren bir seçici; false ya da unknown olarak değerlendirilen bir seçici eşleşmiyor.
  – Aritmetik ifadeler kendilerinden, aritmetik işlemlerden, tanıtıcılardan (tanıtıcı değeri sayısal hazır bilgi olarak işlenir) ve sayısal hazır bilgilerden oluşur.
  – Koşullu ifadeler kendilerinden, karşılaştırma işlemlerinden ve mantıksal işlemlerden oluşur.

- İfadelerin değerlendirilme sırasını ayarlamak için standart bracketing () desteklenir.
- Mantıksal işleçler öncelik sırasıyla: `NOT`, `AND`, `OR`.
- Karşılaştırma işleçleri: =, >, >=, <, <=, <> (eşit değil).

  – İki bayt dizilimi yalnızca dizgiler aynı uzunluktuysa ve bayt sırası eşitse eşittir.
  – Yalnızca aynı tipte değerler karşılaştırılabilir. Bir özel durum, tam sayısal değerleri karşılaştırmak ve yaklaşık sayısal değerleri karşılaştırmak için geçerlidir (gereken tip dönüştürme, Java sayısal promosyon kurallarına göre tanımlanır). Farklı tipleri karşılaştırma girişimi varsa, seçici her zaman false olur.
  – Dizgi ve Boole karşılaştırması = ve <>ile sınırlıdır. İki dizgi yalnızca aynı karakter sırasını içeriyorsa eşittir.

- Öncelik sırasına göre aritmetik işleçler:
  - +, - birli.
  - * çarpma ve / bölme.
  - + ekleme ve - çıkarma.
  - NULL değerinde aritmetik işlemler desteklenmez. Bunlar denenirse, tüm seçici her zaman false olur.
  - Aritmetik işlemler Java sayısal yükseltmesi kullanmalıdır.
- arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 ve arithmetic-expr3 karşılaştırma işleci:
  - `Age BETWEEN 15 and 19` , `age>= 15 AND age <= 19`ile eşdeğerdir.
  - `Age NOT BETWEEN 15 and 19` , `age < 15 OR age > 19`ile eşdeğerdir.
  - Bir BETWEEN işleminin ifadelerinden herhangi biri NULL ise, işlemin değeri false olur. Bir NOT BETWEEN işleminin ifadelerinden herhangi biri NULLise, işlemin değeri true olur.
- tanıtıcı `[NOT] IN (string-literal1, string-literal2,...)` karşılaştırma işleci; burada tanıtıcı bir Dizgi ya da NULL değerine sahiptir.
  - `Country IN ('UK', 'US', 'France')` , `'UK'` için true, `'Peru'`için false değeridir. `(Country = 'UK') OR (Country = 'US') OR (Country = 'France')`ifadesiyle eşdeğerdir.
  - `Country NOT IN ('UK', 'US', 'France')` , `'UK'` için false, `'Peru'`için true değeridir. `NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France'))`ifadesiyle eşdeğerdir.
  - Bir IN ya da NOT IN işleminin tanıtıcısı NULL ise, işlemin değeri bilinmiyor.
- `identifier [NOT] LIKE` *`pattern-value`* `[ESCAPE` *`escape-character`* `]` karşılaştırma işleci; burada `identifier` bir dizgi değerine sahiptir. *örüntü-değeri* bir dizgi hazır bilgisidir; burada _ herhangi bir tek karakteri ve % herhangi bir karakter sırasını (boş sıra da içinde olmak üzere) gösterir. Diğer tüm karakterler kendilerini savunurlar. İsteğe bağlı *kaçış karakteri* , *örüntü-değeri*içindeki _ ve % özel anlamlarına kaçış karakteri olarak kullanılan tek karakterli bir dizgi hazır bilgisidir. LIKE işleci yalnızca iki dizgi değerini karşılaştırmak için kullanılmalıdır.
  - `phone LIKE '12%3'` , 123 ve 12993 için true ve 1234 için false değeridir.
  - `word LIKE 'l_se'` , kayıp için doğru, false ise gevşek için geçerlidir.
  - `underscored LIKE '\_%' ESCAPE '\'` , `_foo` için true, `bar`için false değeridir.
  - `phone NOT LIKE '12%3'` , 123 ve 12993 için false ve 1234için true değeridir.
  - Bir LIKE ya da NOT LIKE işleminin tanıtıcısı NULLise, işlemin değeri bilinmemektedir.

  Not: İki dizgi değerini karşılaştırmak için LIKE işleci kullanılmalıdır. `Root.MQMD.CorrelId` değeri, karakter dizilimi değil, 24 baytlık bir dizidir. `Root.MQMD.CorrelId LIKE 'ABC%'` seçici dizgisi, ayrıştırıcı tarafından sözdizimsel olarak geçerli olarak kabul edilir, ancak false olarak değerlendirilir. Bir bayt dizisini bir karakter dizgisiyle karşılaştırırken LIKE kullanılamaz.
- `identifier IS NULL` karşılaştırma işleci, bir NULL üstbilgi alanı değeri ya da eksik bir özellik değeri için test eder.
- `identifier IS NOT NULL` karşılaştırma işleci, boş olmayan bir üstbilgi alanı değerinin ya da bir özellik değerinin varlığına ilişkin sınamalar.
- Boş değerler

  NULL değerlerini içeren seçici ifadelerinin değerlendirilmesi, özet olarak SQL 92 NULL anlambilimi tarafından tanımlanır:
  - SQL, bir NULL değerini bilinmeyen olarak işler.
  - Bilinmeyen bir değerle karşılaştırma ya da aritmetik her zaman bilinmeyen bir değer verir.
  - IS NULL ve IS NOT NULL işleçleri, bilinmeyen bir değeri TRUE ve FALSE değerlerine dönüştürür.

  Boole işleçleri üç değerli mantık kullanır (T=TRUE, F=FALSE, U=UNKNOWN)

Çizelge 1. Mantık A AND B olduğunda Boole işleci sonucunun değeri		
İşletmen A	**Operatör B**	**Sonuç (A VE B)**
T	F	F
T	U	U
T	T	T
F	T	F
F	U	F
F	F	F
U	T	U
U	U	U
U	F	F

Çizelge 2. Mantık A OR B olduğunda Boole işleci sonucunun değeri		
İşletmen A	**Operatör B**	**Sonuç (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

Çizelge 3. Mantık NOT A olduğunda Boole işleci sonucunun değeri	
İşletmen A	**Sonuç (NOT A)**
T	F
F	T
U	U

Aşağıdaki ileti seçici, ileti tipi otomobili, rengi mavi ve ağırlığı 2500 lbs 'den fazla olan iletileri seçer:

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

SQL sabit ondalık karşılaştırmayı ve aritmetiği desteklese de, ileti seçiciler desteklemez. Bu nedenle, tam sayısal hazır bilgiler ondalık olmayan değerlerle sınırlıdır. Bu nedenle, yaklaşık bir sayısal değer için alternatif bir gösterim olarak ondalık içeren sayısal değerler de vardır.

SQL açıklamaları desteklenmiyor.

İlgili kavramlar

"İleti Özellikleri" sayfa 25

Bir uygulamanın işlenecek iletileri seçmesini ya da MQMD ya da MQRFH2 üstbilgilerine erişmeden bir iletiyle ilgili bilgileri almasını sağlamak için ileti özelliklerini kullanın. IBM MQ ve JMS uygulamaları arasındaki iletişimi de kolaylaştırır.

İlgili başvurular

MsgHandle

MQBUFMH-Arabelleği ileti tanıtıcısına dönüştür

Seçim dizgisi kuralları ve kısıtlamaları

Seçicileri kullanırken olası sorunları önlemek için seçim dizgilerinin nasıl yorumlandığına ve karakter kısıtlamalarına ilişkin bu kuralları tanıyın.

- Yayınlama/abone olma ileti alışverişi için ileti seçimi, yayınlayıcı tarafından gönderildiği şekilde iletide gerçekleşir. Bkz. Seçim dizgileri.
- Eşdeğerlik, tek bir eşittir karakteri kullanılarak test edilir; örneğin, `a = b` doğrudur, ancak `a == b` yanlıştır.
- Birçok programlama dili tarafından 'buna eşit değil' seçeneğini temsil etmek için kullanılan bir işleç `!=`' dir. Bu gösterim, <> için geçerli bir eşanlamlı değil; Örneğin, `a <> b` geçerlidir, ancak `a != b` geçerli değildir.
- Tek tırnak işaretleri yalnızca ' (U+0027) karakteri kullanılır. Benzer şekilde, yalnızca bayt dizilimlerini çevrelemek için kullanıldığında geçerli olan çift tırnak işaretleri, " (U+0022) karakteri.
- &, &&, | ve || simgeleri, mantıksal bağlaç/ayırmanın eşanlamlıları değildir; örneğin, `a && b` , `a AND` bolarak belirtilmelidir.
- `*` ve ? joker karakterleri, % ve _ile eşanlamlı değildir.
- `20 < b < 30` gibi bileşik ifadeler içeren seçiciler geçerli değil. Ayrıştırıcı, soldan sağa aynı önceliğe sahip işleçleri değerlendirir. Bu nedenle örnek, anlamlı olmayan `(20 < b) < 30`olur. Bunun yerine ifade `(b > 20) AND (b < 30)`olarak yazılmalıdır.
- Bayt dizilimleri çift tırnak içine alınmalıdır; tek tırnak işareti kullanılırsa, bayt dizilimi dizgi hazır bilgisi olarak alınır. `0x` ' nin ardından gelen karakter sayısı (karakterlerin temsil ettiği sayı değil), ikinin katı olmalıdır.
- `IS` anahtar sözcüğü eşittir karakterinin eşanlamlısı değil. Bu nedenle, `a IS 3` ve `b IS 'red'` seçim dizgileri geçerli değil. `IS` anahtar sözcüğü yalnızca `IS NULL` ve `IS NOT NULL` durumlarını desteklemek için vardır.

İlgili kavramlar

“Seçim davranışı” sayfa 33

IBM MQ seçim davranışına genel bakış.

İlgili başvurular

Seçim dizgileri

İleti seçiciler kullanılırken UTF-8 ve Unicode ile ilgili önemli noktalar

Bir seçim dizgisinin ayrılmış anahtar sözcüklerini oluşturan tek tırnak işareti içine alınmayan karakterler, Temel Latin Unicode (U+0000 ile U+0007Farasında) biçiminde girilmelidir. Alfasayısal karakterlerin diğer kod noktası gösterimlerini kullanmak geçerli değildir. Örneğin, 1 sayısı Unicode 'da U+0031 olarak ifade edilmelidir; Fullwidth Digit eşdeğerinin U+FF11 ya da Arapça eşdeğerinin U+0661kullanılması geçersizdir.

İleti özelliği adları, geçerli herhangi bir Unicode karakter sırası kullanılarak belirtilebilir. UTF-8 kodlamalı seçim dizgilerinde bulunan ileti özelliği adları, çok baytlık karakterler içerse bile doğrulanır. Çok baytlık UTF-8 geçerlilik denetimi sıkıdır ve ileti özelliği adları için geçerli UTF-8 sıralarının kullanıldığından emin olmanız gerekir. Unicode Temel Çok Dilli Plane 'nin (yukarıdaki U + FFFF) ötesindeki karakterler, UTF-16 olarak vekil kod noktalarıyla (X'D800'-X'DFFF') ya da UTF-8içindeki dört byte, ileti özelliği adlarında desteklenmez.

Eşitlik için karşılaştırma yapılırken özellik adları ya da değerlerle ilgili ek işlem gerçekleştirilmez. Bu, örneğin hiçbir ön/de-kompozisyon gerçekleşmediği ve ligatlara özel bir anlam verilmediği anlamına gelir. Örneğin, önceden oluşturulmuş umlaut karakteri U+00FC U+0075 + U+0308 ile eşdeğer olarak kabul

edilmez ve karakter sırası ff, Unicode U+FB00 (LATIN SMALL LIGATURE FF) ile eşdeğer olarak kabul edilmez.

Tek tırnak işareti içine alınan özellik verileri herhangi bir bayt dizisiyle gösterilebilir ve doğrulanmaz.

İletinin içeriğini seçme

Bir ileti bilgi yükü içeriği seçimine (içerik süzme olarak da bilinir) dayalı olarak abone olunabilir, ancak bu tür bir aboneliğe hangi iletilerin teslim edilmesi gerektiğine ilişkin karar doğrudan IBM MQtarafından gerçekleştirilemez; bunun yerine, iletileri işlemek için genişletilmiş bir ileti seçimi sağlayıcısı (örneğin, IBM Integration Bus) gerekir.

Bir uygulama bir ya da daha fazla abonenin iletinin içeriğinde seçim dizgisi seçtiği bir konu dizgisinde yayınlandığında, IBM MQ genişletilmiş ileti seçimi sağlayıcısının yayını ayrıştırmasını ister ve IBM MQ ' e yayının her abone tarafından belirtilen seçim ölçütleriyle bir içerik süzgeciyle eşleşip eşleşmediğini bildirir.

Genişletilmiş ileti seçimi sağlayıcısı, yayının abonenin seçim dizgisiyle eşleştiğini belirlerse, ileti aboneye teslim edilmeye devam eder.

Genişletilmiş ileti seçimi sağlayıcısı yayının eşleşmediğini belirlerse, ileti aboneye teslim edilmez. Bu, MQPUT ya da MQPUT1 çağrıının başarısız olmasına neden olabilir; neden kodu MQRC_PUBLICATION_FAILURE. Genişletilmiş ileti seçimi sağlayıcısı yayını ayrıştıramazsa, neden kodu MQRC_CONTENT_ERROR döndürülür ve MQPUT ya da MQPUT1 çağrısı başarısız olur.

Genişletilmiş ileti seçimi sağlayıcısı kullanılamıyorsa ya da abonenin yayını alıp almaması gerektiğini saptayamıyorsa, MQRC_SELECTION_NOT_UNAVAILABLE neden kodu döndürülür ve MQPUT ya da MQPUT1 çağrısı başarısız olur.

Bir içerik süzgeciyle bir abonelik yaratıldığında ve genişletilmiş ileti seçimi sağlayıcısı kullanılamıyorsa, MQSUB çağrısı MQRC_SELECTION_NOT_, kullanılabilir neden koduyla başarısız olur. İçerik süzgeci içeren bir abonelik sürdürülüyorsa ve genişletilmiş ileti seçimi sağlayıcısı kullanılamıyorsa, MQSUB çağrısı MQRC_SELECTION_NOT_ALLOWED uyarısını döndürür, ancak aboneliğin sürdürülmesine izin verilir.

İlgili başvurular

Seçim dizgileri

IBM MQ iletilerinin zamanuyumsuz tüketimi

Zamanuyumsuz tüketim, MQI (Message Queue Interface; İleti Kuyruğu Arabirimi) uzantıları kümesini kullanır; MQI, MQCB ve MQCTL ' yi çağırır; bu, bir MQI uygulamasının bir kuyruk kümesindeki iletileri tüketmek üzere yazılmasına olanak sağlar. İletiler, iletiyi ileten uygulama tarafından tanımlanan bir 'kod birimi' çağrılarak ya da iletiyi gösteren bir simge çağrılarak uygulamaya teslim edilir.

Uygulama ortamlarının en basitinde, kod birimi bir işlev göstergesi tarafından tanımlanır, ancak diğer ortamlarda kod birimi bir program ya da modül adıyla tanımlanabilir.

İletilerin zamanuyumsuz tüketiminde aşağıdaki terimler kullanılır:

İleti tüketicisi
: Uygulama gereksinmesiyle eşleşen bir program ya da işlev varsa, bir iletiyle çağrılacak bir program ya da işlev tanımlamanızı sağlayan bir programlama yapısı.

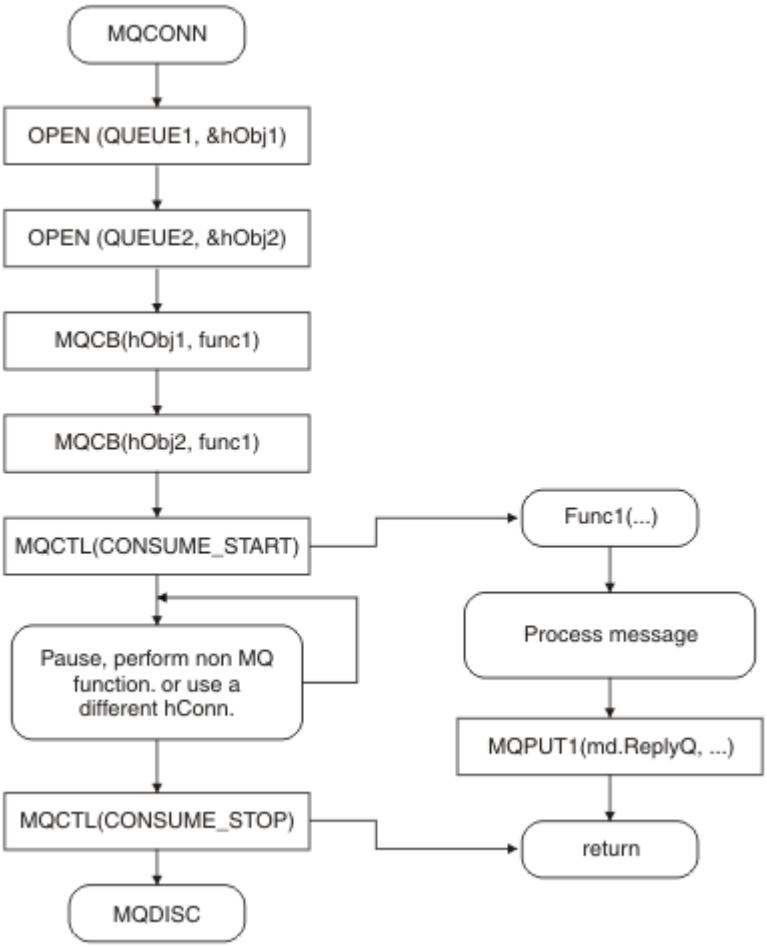
Olay işleyici
: Kuyruk yöneticisi susturması gibi zamanuyumsuz bir olay oluştuğunda çağrılması için bir program ya da işlev tanımlamanızı sağlayan bir programlama yapısı.

Geri çağırma
: Bir İleti Tüketicisi ya da Olay İşleyici yordamına gönderme yapmak için kullanılan soysal bir terim.

Zamanuyumsuz tüketim, özellikle birden çok giriş kuyruğunun ya da aboneliğini işleyen yeni uygulamaların tasarlanmasını ve uygulanmasını basitleştirebilir. Ancak, birden fazla giriş kuyruğu kullanıyorsanız ve iletileri öncelik sırasına göre işliyorsanız, öncelik sırası her bir kuyruk içinde bağımsız olarak gözlemlenir: Bir kuyruktan yüksek öncelikli iletilerden önce düşük öncelikli iletiler alabilir. Birden çok kuyruk arasındaki ileti sırası garanti edilmez. API çıkışlarını kullanıyorsanız, bunları MQCB ve MQCTL çağrılarını içerecek şekilde değiştirmeniz gerekebileceğini de unutmayın.

Aşağıdaki şekillerde, bu işlevi nasıl kullanabileceğinize ilişkin bir örnek verilmiştir.

Şekil 5 sayfa 41 içinde iki kuyruktan ileti alan çok iş parçacıklı bir uygulama gösterilir. Bu örnek, tek bir işleve teslim edilmekte olan tüm iletileri gösterir.

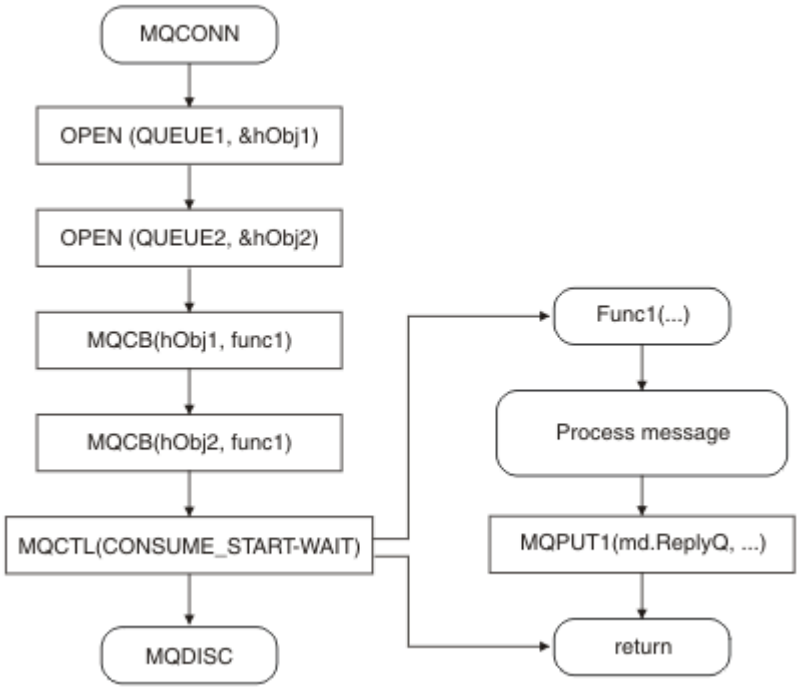


Şekil 5. İki kuyruktan alan Standart İleti Odaklı Uygulama

z/OS z/OSüzerinde, ana denetim iş parçacığı sona ermeden önce bir MQDISC çağrısı yayınlamalıdır. Bu, geri çağırma iş parçacıklarının sistem kaynaklarını sona erdirmesini ve serbest bırakmasını sağlar.

Şekil 6 sayfa 42 Bu örnek akış, iki kuyruktan ileti alan tek bir iş parçacıklı uygulamayı gösterir. Bu örnek, tek bir işleve teslim edilmekte olan tüm iletileri gösterir.

Zamanuyumsuz vakadan farkı, denetimin tüm tüketiciler devre dışı kalıncaya kadar MQCTL yayıncısına geri dönmememeleridir; bu, bir tüketicinin MQCTL STOP isteği yayınlamış olması ya da kuyruk yöneticisinin susturması olmasıdır.



Şekil 6. İki kuyruktan tek iş parçacıklı ileti odaklı uygulama tüketimi

İleti grupları

İletilerin sıralanmasına izin vermek için gruplar içinde iletiler oluşabilir.

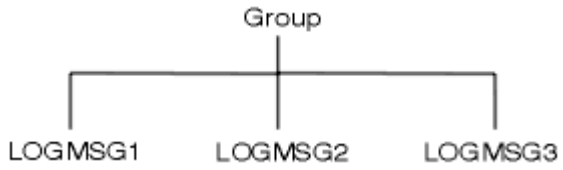
İleti grupları, birden çok iletinin birbiriyle ilişkili olarak işaretlenmesine ve mantıksal bir siparişin gruba uygulanmasına izin verir (bkz. "Mantıksal ve fiziksel sıralama" sayfa 742). Çoklu platformlarişletim sistemlerinde, ileti bölümlemesi büyük iletilerin daha küçük bölümlere ayrılmasını sağlar. Gruplanmış ya da bölümlenmiş iletileri bir konuya koyarken kullanamazsınız.

Bir grup içindeki sıradüzen aşağıdaki gibidir:

Grup
Bu, sıradüzendeki en yüksek düzeydir ve bir *GroupId*ile tanımlanır. Aynı *GroupId*içeren bir ya da daha çok iletiden oluşur. Bu iletiler kuyrukta herhangi bir yerde saklanabilir.

Not: Burada *ileti* terimi, kuyruktaki bir öğeyi belirtmek için kullanılır; örneğin, MQGMO_COMPLETE_MSG belirtmeyen tek bir MQGET tarafından döndürüleceği gibi.

Şekil 7 sayfa 42 içinde bir grup mantıksal ileti gösterilmektedir:



Şekil 7. Mantıksal ileti grubu

Bir kuyruk açarak ve MQOO_BIND_ON_GROUP belirterek, bu kuyruğa gönderilen bir gruptaki tüm iletileri kuyruğun aynı eşgörünümüne gönderilmeye zorlayabilirsiniz. BIND_ON_GROUP seçeneğine ilişkin ek bilgi için İleti yakınlıklarıbaşlıklı konuya bakın.

Mantıksal ileti
Bir grup içindeki mantıksal iletiler, *GroupId* ve *MsgSeqNumber* alanları tarafından tanımlanır. *MsgSeqNumber* , bir gruptaki ilk ileti için 1 'den başlar ve bir ileti bir grupta değilse, alanın değeri 1 'dir.

Aşağıdaki işlemleri gerçekleştirmek için bir grup içindeki mantıksal iletileri kullanın:

- Sıralamayı doğrulayın (iletinin iletildiği koşullar altında bu garanti edilmezse).
- Uygulamaların benzer iletileri (örneğin, aynı sunucu eşgörünümü tarafından işlenmesi gerekenler) gruplamasına izin verin.

Bir gruptaki her ileti, bölümlere bölünmedikçe, bir fiziksel iletiden oluşur. Her ileti mantıksal olarak ayrı bir iletidir ve yalnızca MQMD ' deki *GroupId* ve *MsgSeqNumber* alanlarının gruptaki diğer iletilerle ilişkileri olması gerekir. MQMD ' deki diğer alanlar bağımsızdır; bazıları gruptaki tüm iletiler için aynı olabilir, diğerleri farklı olabilir. Örneğin, bir gruptaki iletilerin biçim adları, CCSID ' leri ve kodlamaları farklı olabilir.

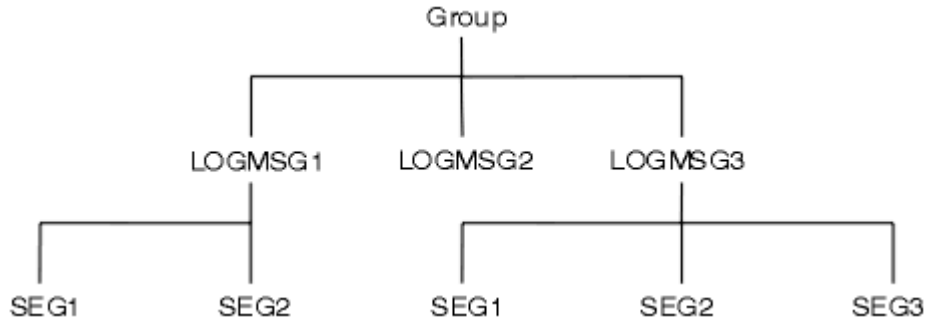
Bölüm

Kesimler, koyma ya da alma uygulaması ya da kuyruk yöneticisi (iletinin geçtiği araya giren kuyruk yöneticileri de içinde olmak üzere) için çok büyük olan iletileri işlemek için kullanılır. Daha fazla bilgi için “İleti bölümlemesi” sayfa 760 başlıklı konuya bakın.

Tek bir ileti, *segment*adı verilen daha küçük iletilere ayrılır. İletinin bir bölümü *GroupId*, *MsgSeqNumber*ve *Offset* alanlarıyla tanıtılır. *Offset* alanı, bir ileti içindeki ilk bölüm için sıfırdan başlar.

Her bölüm, bir gruba ait olabilecek bir fiziksel iletiden oluşur (Şekil 8 sayfa 43 içinde bir grup içindeki iletilerin bir örneği gösterilmektedir). Bir bölüm mantıksal olarak tek bir iletinin parçasıdır; bu nedenle, MQMD ' deki yalnızca *MsgId*, *Offset*ve *MsgFlags* alanları aynı iletinin ayrı kısımları arasında farklılık göstermelidir. Bir kesim gelmezse, neden kodu MQRC_INCOMPLETE_GROUP ya da MQRC_INCOMPLETE_MSG döndürülür.

Şekil 8 sayfa 43 içinde, bazıları kesimlere ayrılmış bir grup mantıksal ileti gösterilmektedir:



Şekil 8. Bölümlenmiş iletiler

z/OS IBM MQ for z/OSüzerinde bölümleme desteklenmez.

Yayınlama/Abone Olma ile bölümlenmiş ya da gruplanmış iletileri kullanamazsınız.

İlgili kavramlar

“İleti bölümlemesi” sayfa 760
İletilerin bölümlenmesine ilişkin bilgi edinmek için bu bilgileri kullanın. Bu özellik IBM MQ for z/OS üzerinde ya da IBM MQ classes for JMSkullanan uygulamalar tarafından desteklenmez.

İlgili başvurular

“Mantıksal ve fiziksel sıralama” sayfa 742
Kuyruklardaki iletiler (her bir öncelik düzeyi içinde) *fiziksel* ya da *mantıksal* sırada oluşabilir.

MQMD-İleti tanımlayıcı

İleti kalıcılığı

Kalıcı iletiler, günlüklere ve kuyruk veri dosyalarına yazılır. Bir hata sonrasında kuyruk yöneticisi yeniden başlatılırsa, günlüğe kaydedilen verilerden gerektiğinde bu kalıcı iletileri kurtarır. Kalıcı olmayan iletiler, bir kuyruk yöneticisi durursa, ister bir işletmen komutunun sonucu olsun, ister sisteminizin bir bölümünün başarısız olması nedeniyle atılır.

z/OS z/OS üzerindeki bir bağlaşım olanağında (CF) saklanan kalıcı olmayan iletiler bu durum için kural dışı bir durumdur. CF kullanılabilir olduğu sürece devam eder.

Bir ileti yarattığınızda, varsayılan değerleri kullanarak ileti tanımlayıcıyı (MQMD) kullanıma hazırlarsanız, iletiye ilişkin kalıcılık, MQOPEN komutunda belirtilen kuyruğun **DefPersistence** özniteliğinden alınır. Diğer bir seçenek olarak, iletiyi kalıcı ya da kalıcı olmayan olarak tanımlamak için MQMD yapısının *Persistence* alanını kullanarak iletinin kalıcılığını ayarlayabilirsiniz.

Kalıcı iletiler kullandığınızda uygulamanızın performansı etkilenir; bu etkinin kapsamı, makinenin G/Ç altsisteminin performans özelliklerine ve her bir platformda eşitleme noktası seçeneklerini nasıl kullandığınıza bağlıdır:

- Geçerli iş biriminin dışındaki kalıcı bir ileti, her koyma ve alma işleminde diske yazılır. Bkz. “İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818.
- z/OS ALW IBM idışındaki tüm platformlar için, yalnızca iş birimi kesinleştirildiğinde geçerli iş birimi içindeki kalıcı bir ileti günlüğe kaydedilir ve iş birimi birçok kuyruk işlemi içerebilir.

Hızlı ileti alışverişi için kalıcı olmayan iletiler kullanılabilir. Hızlı iletilerle ilgili daha fazla bilgi için bkz. İletilerin güvenliği .

Not: Bir iş birimi içinde kalıcı iletiler yazılması ve bir birimin ya da işin dışına kalıcı iletiler yazılması, uygulamalarınız için önemli performans sorunlarına neden olabilir. Bu, özellikle her iki işlem için de aynı hedef kuyruk kullanıldığında geçerlidir.

Teslim edilmeyen iletiler

Kuyruk yöneticisi bir iletiyi kuyruğa koyamadığında, çeşitli seçenekleriniz vardır.

Şunları yapabilirsiniz:

- İletiyi kuyruğa koymayı yeniden deneyin.
- İletinin gönderene geri gönderilmesini isteyin.
- İletiyi gönderilmeyen iletiler kuyruğuna koyun.

Ek bilgi için bkz.“Yordamsal program hatalarının işlenmesi” sayfa 994.

Geriletilmeyen iletiler

Bir iş biriminin denetimi altındaki bir kuyruktan gelen iletiler işlenirken, iş birimi bir ya da daha çok iletiden oluşabilir. Bir geriletme oluşursa, kuyruktan alınan iletiler kuyrukta yeniden kullanılır ve başka bir iş biriminde yeniden işlenebilir. Belirli bir iletinin işlenmesi soruna neden oluyorsa, iş birimi geri çekilmiştir. Bu, bir işleme döngüsüne neden olabilir. Kuyruğa konan iletiler kuyruktan kaldırılır.

Bir uygulama, MQMD ' nin *BackoutCount* alanını test ederek böyle bir döngüye yakalanan iletileri saptayabilir. Uygulama durumu düzeltebilir ya da bir işlece uyarı verebilir.

Multi Geriletme sayısı her zaman kuyruk yöneticisini yeniden başlatmaya devam eder. **HardenGetBackout** özniteliğinde herhangi bir değişiklik yoksayılır.

z/OS Paylaşılan kuyruklar için geriletme sayısı her zaman kuyruk yöneticisini yeniden başlatmaya devam eder. z/OSüzerindeki diğer tüm yapılandırmalar için, özel kuyruklara ilişkin geriletme sayısının kuyruk yöneticisini yeniden başlatmaya devam etmesinden emin olmak için *HardenGetBackout* özniteliğini MQQA_BACKOUT_HARDENED olarak ayarlayın; tersi durumda, kuyruk yöneticisinin yeniden başlatılması gerekirse, her ileti için doğru bir geriletme sayısını korumaz. Özniteliğin bu şekilde ayarlanması, fazladan işleme maliyetini ekler.

İletileri kesinleştirme ve geri çekmeyle ilgili daha fazla bilgi için bkz. “İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818.

Yanıtın gönderileceği kuyruk ve kuyruk yöneticisi

Gönderdiğiniz bir iletiye yanıt olarak ileti alabileceğiniz durumlar vardır:

- Bir istek iletisine yanıt olarak gönderilen yanıt iletisi
- Beklenmeyen bir olay ya da süre bitimine ilişkin rapor iletisi
- Bir COA (Varış Onayı) ya da COD (Teslim Doğrulaması) olayıyla ilgili bir rapor iletisi
- Bir PAN (Pozitif İşlem Bildirimi) ya da NAN (Negatif İşlem Bildirimi) olayıyla ilgili bir rapor iletisi

MQMD yapısını kullanarak, *ReplyToQ* alanında yanıt ve rapor iletilerinin gönderilmesini istediğiniz kuyruğun adını belirtin. *ReplyToQMgr* alanında, yanıt kuyruğunun iyesi olan kuyruk yöneticisinin adını belirtin.

ReplyToQMgr alanını boş bırakırsanız, kuyruk yöneticisi kuyruktaki ileti tanımlayıcısında aşağıdaki alanların içeriğini ayarlar:

ReplyToQ
ReplyToQ uzak kuyruğun yerel tanımıysa, *ReplyToQ* alanı uzak kuyruğun adına ayarlanır; tersi durumda bu alan değişmez.

ReplyToQMgr
ReplyToQ uzak kuyruğun yerel bir tanımlamasıysa, *ReplyToQMgr* alanı uzak kuyruğun iyesi olan kuyruk yöneticisinin adına ayarlanır; tersi durumda *ReplyToQMgr* alanı, uygulamanızın bağlı olduğu kuyruk yöneticisinin adına ayarlanır.

Not: Bir kuyruk yöneticisinin bir iletiyi teslim etmek için birden çok girişimde bulunmasını isteyebilir ve başarısız olursa iletinin atılması isteğinde bulunabilirsiniz. İleti, teslim edilemedikten sonra atılmazsa, uzak kuyruk yöneticisi iletiyi teslim edilmeyen ileti kuyruğuna koyar (bkz. “Teslim edilmeyen ileti kuyruğunun kullanılması” sayfa 997).

İleti bağlamı

İleti bağlamı bilgileri, iletiyi alan uygulamanın iletiyi başlatan hakkında bilgi edinmesini sağlar.

Alma uygulaması aşağıdakileri yapmak isteyebilir:

- Gönderen uygulamanın doğru yetki düzeyine sahip olup olmadığını denetleyin
- Gerçekleştirmesi gereken herhangi bir iş için gönderen uygulamayı şarj edebilmesi için bir muhasebe işlevi gerçekleştirin
- Birlikte çalıştığı tüm iletilerin bir denetim izini tut

Kuyruğa ileti koymak için MQPUT ya da MQPUT1 çağrısını kullandığınızda, kuyruk yöneticisinin ileti tanımlayıcısına varsayılan bağlam bilgileri ekleyeceğini belirtebilirsiniz. Uygun yetki düzeyine sahip uygulamalar ek bağlam bilgileri ekleyebilir. Bağlam bilgilerini nasıl belirteceğiniz hakkında daha fazla bilgi için bkz. “İleti bağlamı bilgilerinin denetlenmesi” sayfa 728.

Kullanıcı bağlamı, aşağıdaki rapor iletisi tiplerini oluştururken kuyruk yöneticisi tarafından kullanılır:

- Teslimde onayla
- Son kullanma tarihi

Bu rapor iletileri oluşturulduğunda, raporun hedefi üzerinde kullanıcı bağlamı + koyma ve + passid yetkisi için denetlenir. Kullanıcı bağlamının yeterli yetkisi olmadığı durumlarda, rapor iletisi, tanımlanmışsa, gitmeyen iletiler kuyruğuna yerleştirilir. Gitmeyen iletiler kuyruğu yoksa, rapor iletisi atılır.

Tüm bağlam bilgileri, ileti tanımlayıcısının bağlam alanlarında saklanır. Bilgi tipi kimlik, kaynak ve kullanıcı bağlamı bilgilerine dayanır.

Kimlik bağlamı

Kimlik bağlamı bilgileri, iletiyi ilk olarak kuyruğa koyan uygulama kullanıcısını tanımlar. Uygun yetkili uygulamalar aşağıdaki alanları belirleyebilir:

- Kuyruk yöneticisi, *UserIdentifier* alanını kullanıcıyı tanıtan bir adla doldurur; kuyruk yöneticisinin bunu yapabilme şekli, uygulamanın çalıştığı ortama bağlıdır.

- Kuyruk yöneticisi, *AccountingToken* alanını iletiyi koyan uygulamadan belirlediği bir simge ya da sayıyla doldurur.
- Uygulamalar, kullanıcı hakkında (örneğin, şifrelenmiş bir parola) eklemek istedikleri ek bilgiler için *ApplIdentityData* alanını kullanabilir.

IBM MQ for Windowsaltında bir ileti yaratıldığında, *AccountingToken* alanında bir Windows sistem güvenlik tanıtıcısı (SID) saklanır. SID, *UserIdentifier* alanını tamamlamak ve bir kullanıcının kimlik bilgilerini oluşturmak için kullanılabilir.

Kuyruk yöneticisinin *UserIdentifier* ve *AccountingToken* alanlarını nasıl dolduracağına ilişkin bilgi için, UserIdentifier ve AccountingTokeniçindeki bu alanların açıklamalarına bakın.

İletileri bir kuyruk yöneticisinden diğerine ileten uygulamalar, diğer uygulamaların iletiyi oluşturan kişinin kimliğini bilmeleri için kimlik bağlamı bilgilerini de aktarmalıdır.

Kaynak bağlam

Kaynak bağlam bilgileri, iletiyi, iletinin saklandığı kuyruğa koyan uygulamayı açıklar. İleti tanımlayıcı, kaynak bağlam bilgileri için aşağıdaki alanları içerir:

- *PutApplType* , iletiyi koyan uygulamanın tipini tanımlar (örneğin, bir CICS hareketi).
- *PutApplName* , iletiyi koyan uygulamanın adını tanımlar (örneğin, bir işin ya da hareketin adı).
- *PutDate* , iletinin kuyruğa konduğu tarihi tanımlar.
- *PutTime* , iletinin kuyruğa konduğu saati tanımlar.
- *ApplOriginData* , bir uygulamanın iletinin kökeni hakkında içermek istediği ek bilgileri tanımlar. Örneğin, kimlik verilerinin güvenilir olup olmadığını belirtmek için uygun yetkili uygulamalar tarafından ayarlanabilir.

Kaynak bağlam bilgileri genellikle kuyruk yöneticisi tarafından sağlanır. Greenwich Ortalama Saati (GMT), *PutDate* ve *PutTime* alanları için kullanılır. PutDate ve PutTimeiçindeki bu alanların açıklamalarına bakın.

Yeterli yetkisi olan bir uygulama kendi bağlamını sağlayabilir. Bu, tek bir kullanıcı, kaynaklandığı bir iletiyi işleyen her bir sistemde farklı bir kullanıcı kimliğine sahip olduğunda muhasebe bilgilerinin korunmasını sağlar.

IBM MQ nesneler

Bu bilgiler, kuyruk yöneticileri, kuyruk paylaşım grupları, kuyruklar, denetim konusu nesneleri, ad, süreç tanımlamaları, kimlik doğrulama bilgileri nesneleri, kanallar, depolama sınıfları, dinleyiciler ve hizmetler gibi IBM MQ nesnelerine ilişkin ayrıntıları sağlar.

Kuyruk yöneticileri, bu nesnelerin özelliklerini (öznitelikler olarak bilinir) tanımlar. Bu özniteliklerin değerleri, IBM MQ ' in bu nesneleri işleme şeklini etkiler. Uygulamalarınızda, bu nesneleri denetlemek için İleti Kuyruğu Arabirimi 'ni (MQI) kullanırsınız. Nesneler, bir programdan adreslendiğinde bir *nesne tanımlayıcısı* (MQOD) tarafından tanımlanır.

Örneğin, nesneleri tanımlamak, değiştirmek ya da silmek için IBM MQ komutlarını kullandığınızda, kuyruk yöneticisi bu işlemleri gerçekleştirmek için gereken yetki düzeyine sahip olup olmadığınızı denetler. Benzer şekilde, bir uygulama bir nesneyi açmak için MQOPEN çağrısı kullandığında, kuyruk yöneticisi, uygulamanın o nesneye erişime izin vermeden önce gerekli yetki düzeyine sahip olup olmadığını denetler. Denetimler, açılmakta olan nesnenin adına yapılır.

İlgili kavramlar

“İleti bağlamı bilgilerinin denetlenmesi” sayfa 728
Kuyruğa ileti koymak için MQPUT ya da MQPUT1 çağrısını kullandığınızda, kuyruk yöneticisinin ileti tanımlayıcısına varsayılan bağlam bilgileri ekleyeceğini belirtebilirsiniz. Uygun yetki düzeyine sahip uygulamalar ek bağlam bilgileri ekleyebilir. Bağlam bilgilerini denetlemek için MQPMO yapısındaki seçenekler alanını kullanabilirsiniz.

İlgili başvurular

“İleti bağlamıyla ilgili MQOPEN seçenekleri” sayfa 720

Bağlam bilgilerini bir kuyruğa yerleştirirken bir iletiyle ilişkilendirmek istiyorsanız, kuyruğu açarken ileti bağlamı seçeneklerinden birini kullanmanız gerekir.

Windows Microsoft Transaction Server uygulamalarının hazırlanması ve çalıştırılması

Bir MTS uygulamasını IBM MQ MQI client uygulaması olarak çalışacak şekilde hazırlamak için, ortamınız için uygun olan bu yönergeleri izleyin.

IBM MQ kaynaklarına erişen Microsoft Transaction Server (MTS) uygulamalarının geliştirilmesine ilişkin genel bilgi için IBM MQ Help Center olanağındaki MTS bölümüne bakın.

Bir MTS uygulamasını IBM MQ MQI client uygulaması olarak çalışacak şekilde hazırlamak için uygulamanın her bileşeni için aşağıdakilerden birini yapın:

- Bileşen MQI için C dili bağ tanımlarını kullanıyorsa, “Windows ' da C programlarını hazırlama” sayfa 973 içindeki yönergeleri izleyin, ancak bileşeni mqic.libyerine mqicxa.lib kitaplığına bağlayın.
- Bileşen IBM MQ C++ sınıflarını kullanıyorsa, “Windows üzerinde C++ programları oluşturma” sayfa 528 içindeki yönergeleri izleyin, ancak bileşeni imqc23vn.libyerine imqx23vn.lib kitaplığına bağlayın.
- Bileşen MQI için Visual Basic dil bağ tanımlarını kullanıyorsa, “Windows ' da Visual Basic programlarının hazırlanması” sayfa 977 içindeki yönergeleri izleyin, ancak Visual Basic projesini tanımladığınızda **Koşullu Derleme Bağımsız Değişkenleri** alanına `MqType=3` yazın.

IBM MQ uygulamaları için tasarımla ilgili önemli noktalar

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğine karar verdiğinizde, IBM MQtarafından sunulan özellikleri nasıl kullanacağınıza karar vermeniz gerekir.

Bir IBM MQ uygulamasını tasarlarken aşağıdaki soruları ve seçenekleri göz önünde bulundurun:

Uygulama tipi

Başvurunuzun amacı nedir? Geliştirebileceğiniz farklı uygulama tipleriyle ilgili bilgi için aşağıdaki bağlantılara bakın:

- Sunucu
- Müşteri
- Yayınla/abone ol
- Web hizmetleri
- Kullanıcı çıkışları, API çıkışları ve kurulabilir hizmetler

Ayrıca, IBM MQyönetimini otomatikleştirmek için kendi uygulamalarınızı da yazabilirsiniz. Daha fazla bilgi için IBM MQ Yönetim Arabirimi (MQAI) ve Yönetim görevlerinin otomatikleştirilmesibaşlıklı konuya bakın.

Programlama dili

IBM MQ , uygulama yazmak için bir dizi farklı programlama dilini destekler. Daha fazla bilgi için bkz. “IBM MQ için uygulama geliştirilmesi” sayfa 5.

Birden fazla platform için uygulamalar

Uygulamanız birden çok platformda çalışacak mı? Bugün kullandığınız platformdan farklı bir platforma geçmek için bir stratejiniz var mı? Bu sorulardan birinin yanıtı evet ise, programlarınızı platform bağımsızlığı için kodladığınızdan emin olun.

Örneğin, C kullanıyorsanız, ANSI standardında C kodu. Platforma özgü işlev daha hızlı ya da daha verimli olsa da, platforma özgü eşdeğer bir işlev yerine standart bir C kitaplığı işlevini kullanın. Kural dışı durum, koddaki verimlilik çok yüksek olduğunda, #ifdefkomutunu kullanarak her iki durum için de kod oluşturmanız gerekir. Örneğin:

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

Kuyruk tipleri

Her gereksinim duyduğunuzda bir kuyruk yaratmak mı, yoksa önceden ayarlanmış kuyrukları kullanmak mı istiyorsunuz? Bir kuyruğu kullanmayı tamamladığınızda silmek istiyor musunuz, yoksa yeniden mi kullanacaksınız? Uygulamanın bağımsızlığı için diğer ad kuyruklarını kullanmak istiyor musunuz? Desteklenen kuyruk tiplerini görmek için Kuyruklarkonusuna bakın.

z/OS Paylaşılan kuyrukların, kuyruk paylaşım gruplarının ve kuyruk paylaşım grubu kümelerinin kullanılması (yalnızca IBM MQ for z/OS)

Kuyruk paylaşım gruplarıyla paylaşılan kuyrukları kullandığınızda mümkün olan kullanılabilirlik, ölçeklenebilirlik ve iş yükü dengeleme özelliklerinden yararlanmak isteyebilirsiniz. Ek bilgi için Paylaşılan kuyruklar ve kuyruk paylaşım grupları konusuna bakın.

Ortalama ve en yoğun ileti akışlarını tahmin etmek ve iş yükünü dağıtmak için kuyruk paylaşım grubu kümelerini kullanmayı da düşünebilirsiniz. Ek bilgi için Paylaşılan kuyruklar ve kuyruk paylaşım grupları konusuna bakın.

Kuyruk yöneticisi kümelerini kullanma

Basitleştirilmiş sistem yönetiminden ve kümeleri kullandığınızda mümkün olan daha fazla kullanılabilirlik, ölçeklenebilirlik ve iş yükü dengeleme özelliklerinden yararlanmak isteyebilirsiniz.

İleti tipleri

Basit iletiler için veri paketlerini kullanmak isteyebilirsiniz, ancak diğer durumlar için (yanıt beklediğiniz iletiler) isteyebilirsiniz. Bazı iletilerinize farklı öncelikler atamak isteyebilirsiniz. İleti tasarlama hakkında daha fazla bilgi için bkz. “İletiler için tasarım teknikleri” sayfa 55.

Yayınlama/abone olma ya da noktadan noktaya ileti sisteminin kullanılması

Yayınlama/abone olma ileti sistemi kullanılarak, gönderen bir uygulama, IBM MQ iletisinde paylaşmak istediği bilgileri IBM MQ publish? tarafından yönetilen standart bir hedefe gönderir ve IBM MQ ' in bu bilgilerin dağıtımını işlemesini sağlar. Hedef uygulamanın aldığı bilgilerin kaynağı hakkında hiçbir şey bilmesi gerekmez, yalnızca bir ya da daha fazla konuya ilgi duymasını sağlar ve kullanılabilir olduğunda bu bilgileri alır. Yayınlama/abone olma ileti sistemi hakkında daha fazla bilgi için bkz. İleti alışverişi yayınlama/abone olma.

Noktadan noktaya ileti alışverişi kullanan bir uygulama, iletiyi alan bir uygulamanın iletiyi alacağını bildiği belirli bir kuyruğa gönderir. Alan uygulama, iletileri belirli bir kuyruktan alır ve içerikleri üzerinde işlem yapar. Bir uygulama genellikle hem gönderen hem de alıcı olarak işlev görür, başka bir uygulamaya sorgu gönderir ve bir yanıt alır.

IBM MQ programlarınızın denetlenmesi

Bazı programları otomatik olarak başlatmak ya da belirli bir ileti kuyruğa gelinceye kadar beklemek isteyebilirsiniz (IBM MQ *triggering* özelliğini kullanarak, bkz. “Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829). Diğer bir seçenek olarak, kuyruktaki iletiler yeterince hızlı işlenmediğinde (Özel işlemden geçirme olayları konusunda açıklandığı gibi IBM MQ *özel işlemden geçirme olayları* özelliğini kullanarak) uygulamanın başka bir eşgörünümünü başlatmak isteyebilirsiniz.

Uygulamanızı bir IBM MQ istemcisinde çalıştırma

Tam MQI istemci ortamında desteklenir ve bir yordam dilinde yazılmış hemen hemen tüm IBM MQ uygulamaları IBM MQ MQI clientüzerinde çalıştırılmak üzere yeniden bağlanabilir. IBM MQ MQI client üzerindeki uygulamayı MQI kitaplığına değil, MQIC kitaplığına bağlayın. z/OS Get (signal) on z/OS desteklenmiyor.

Not: IBM MQ istemcisinde çalışan bir uygulama eşzamanlı olarak birden çok kuyruk yöneticisine bağlanabilir ya da MQCONN ya da MQCONNX çağrısında yıldız işareti (*) olan bir kuyruk yöneticisi adını kullanabilir. Bu işlev kullanılamayacağı için, istemci kitaplıkları yerine kuyruk yöneticisi kitaplıklarına bağlanmak istiyorsanız uygulamayı değiştirin.

Ek bilgi için bkz. “IBM MQ MQI client ortamında uygulamaları çalıştırma” sayfa 882 .

Uygulama performansı

Tasarım kararları, IBM MQ uygulamalarının performansını artırmaya ilişkin öneriler için uygulama performansınızı etkileyebilir, bkz. “Uygulama tasarımı ve performansıyla ilgili önemli noktalar” sayfa 57 IBM i ve “IBM i uygulamaları için tasarım ve performans konuları” sayfa 60 .

Gelişmiş IBM MQ teknikleri

Daha gelişmiş uygulamalar için, yanıtların ilintilendirilmesi ve IBM MQ bağlam bilgilerinin oluşturulması ve gönderilmesi gibi bazı gelişmiş IBM MQ tekniklerini kullanmak isteyebilirsiniz. Daha fazla bilgi için “Gelişmiş uygulamalar için tasarım teknikleri” sayfa 58 başlıklı konuya bakın.

Verilerinizin güvenliğini sağlama ve bütünlüğünü koruma

İletinin kabul edilebilir bir kaynaktan gönderildiğini sınamak için iletiyle geçirilen bağlam bilgilerini kullanabilirsiniz. Verilerinizin diğer kaynaklarla tutarlı kaldığından emin olmak için IBM MQ ya da işletim sisteminiz tarafından sağlanan eşitleme olanaklarını kullanabilirsiniz (daha fazla ayrıntı için bkz. “İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818). Önemli iletilerin teslim edilmesini sağlamak için IBM MQ iletilerinin *kalıcı saklama* özelliğini kullanabilirsiniz.

IBM MQ uygulamalarının sınanması

IBM MQ programlarına ilişkin uygulama geliştirme ortamı, diğer uygulamalardan farklı değildir; bu nedenle, IBM MQ izleme olanaklarının yanı sıra aynı geliştirme araçlarını da kullanabilirsiniz.

z/OS CICS uygulamalarını IBM MQ for z/OSile test ederken, CICS Execution Diagnostic Facility (CEDF) olanağını kullanabilirsiniz. CEDF, tüm CICS hizmetlerine yapılan çağrıların yanı sıra her MQI çağrısının girişini ve çıkışını tuzaklar. Ayrıca, CICS ortamında, her MQI çağrısından önce ve sonra tanılama bilgileri sağlamak için bir API geçiş çıkış programı yazabilirsiniz. Bunun nasıl yapılacağını öğrenmek için bkz. “IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852.

IBM i IBM i uygulamalarını test ederken, standart Hata Ayıklayıcı 'yı kullanabilirsiniz. Bunu başlatmak için STRDBG komutunu kullanın.

Kural dışı durumların ve hataların işlenmesi

Teslim edilmeyen iletilerin nasıl işleneceğini ve kuyruk yöneticisi tarafından size bildirilen hata durumlarının nasıl çözüleceğini göz önünde bulundurmanız gerekir. Bazı raporlar için, MQPUT ile ilgili rapor seçeneklerini ayarlamanız gerekir.

İlgili kavramlar

IBM MQ teknik genel bakış

“z/OS uygulamaları için tasarım ve performans konuları” sayfa 62
Uygulama tasarımı, performansı etkileyen en önemli faktörlerden biridir. Performansa dahil olan tasarım faktörlerinden bazılarını anlamak için bu konuyu kullanın.

“IBM MQ için uygulama geliştirilmesi” sayfa 5
İleti göndermek ve almak için uygulamalar geliştirebilir ve kuyruk yöneticilerinizi ve ilgili kaynakları yönetebilirsiniz. IBM MQ , birçok farklı dilde ve çerçevede yazılmış uygulamaları destekler.

“Uygulama geliştirme kavramları” sayfa 6
IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

“Kuyruğa alma için yordam uygulaması yazılması” sayfa 692
Kuyruğa alma uygulamaları yazma, bir kuyruk yöneticisine bağlanma ve bağlantı kesme, yayınlama/abone olma ve nesneleri açma ve kapatma hakkında bilgi edinmek için bu bilgileri kullanın.

“İstemci yordamsal uygulamaları yazılıyor” sayfa 875
IBM MQ üzerinde yordam dili kullanarak istemci uygulamaları yazmak için bilmeniz gerekenleri.

“.NET uygulamalarının geliştirilmesi” sayfa 533
IBM MQ classes for .NET , .NET uygulamalarının bir IBM MQ MQI client olarak IBM MQ ' e bağlanmasına ya da bir IBM MQ sunucusuna doğrudan bağlanmasına izin verir.

“C++ uygulamaları geliştirilmesi” sayfa 505
IBM MQ , IBM MQ nesnelerine eşdeğer C++ sınıflarını ve dizi veri tiplerine eşdeğer bazı ek sınıfları sağlar. MQI aracılığıyla kullanılamayan bazı özellikler sağlar.

“IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma” sayfa 78
IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , IBM MQile verilen Java ileti alışverişi sağlayıcılarıdır. JMS ve Jakarta Messaging belirtimlerinde tanımlanan arabirimleri gerçekleştirmenin yanı sıra, bu ileti alışverişi sağlayıcıları Java ileti sistemi API 'sine iki uzantı kümesi ekler.

“kullanmaIBM MQ classes for Java” sayfa 334
Bir Java ortamında IBM MQ kullanın. IBM MQ classes for Java , bir Java uygulamasının IBM MQ istemcisi olarak IBM MQ ' e bağlanmasına ya da IBM MQ kuyruk yöneticisine doğrudan bağlanmasına izin verir.

Desteklenen programlama dillerinde uygulama adının belirtilmesi

IBM MQ 9.2.0' den önce, Java ya da JMS istemci uygulamalarında bir uygulama adı belirtebilirsiniz. IBM MQ 9.2.0 ' den bu özellik, IBM MQ for Multiplatformsüzerindeki diğer programlama dillerine genişletilir.

Uygulama adının nasıl kullanıldığını

Uygulama adı şu kişinin çıkışıyla yazılır:

- runmqsc DISPLAY CONN APPLTAG
- runmqsc DISPLAY QSTATUS TYPE (HANDLE) APPLTAG
- Runmqsc DISPLAY CHSTATUS RAPPLTAG
- MQMD.PutApplName
- Uygulama etkinliği izlemesi

Uygulama adı, uygulama etkinliği izlemesi yapılandırılırken de kullanılır. Java dışı uygulamalar için varsayılan uygulama adı, Windows ve IBM idışında yürütülür dosyanın kısaltılmış adıdır.

Windows Windows' da varsayılan ad, soldaki 28 karaktere kesilmiş, tam olarak nitelenmiş yürütülebilir addır.

IBM i IBM i' da varsayılan ad, iş adıdır.

Java uygulamaları için bu, soldan 28 karaktere kısaltılmış paket adının başına eklenen sınıf adıdır.

Daha fazla bilgi için bkz. PutApplAdı.

IBM MQ 9.2.0' den IBM MQ for Multiplatforms üzerindeki uygulamalar, uygulama adlarını yönetimsel olarak ya da çeşitli programlama yöntemlerini kullanarak ayarlayabilir. Bu, uygulama etkinliği izlemesini yapılandırırken ya da çeşitli **runmqsc** komutlarından çıktı aldığınızda, uygulamaların platformdan bağımsız daha anlamlı bir ad sağlamasına olanak sağlar.

IBM MQ 9.2.0' den tek tip bir kümedeki uygulamaları yeniden dengeleyebilirsiniz. Bunu gerçekleştirmek için anlamlı uygulama adları kullanılır.

Desteklenen karakterler

Uygulama adını nasıl belirttiğinize ilişkin ek bilgi için bkz. “Önerilen uygulama adı karakterleri” sayfa 51 .

Programlama dilleri

C içindeki IBM MQ kitaplıklarına ve diğer programlama dillerine çözülen uygulamaların uygulama adını nasıl sağlayabileceğine ilişkin daha fazla bilgi için bkz. “Programlama dili bağlantıları” sayfa 53 .

Yönetilen .NET uygulamaları

Yönetilen .NET uygulamalarının uygulama adını nasıl sağlayabileceğine ilişkin bilgi için bkz. “Yönetilen .NET uygulamaları” sayfa 53 .

XMS uygulamaları

XMS uygulamalarının uygulama adını nasıl sağlayabileceğine ilişkin bilgi için bkz. “XMS uygulamalar” sayfa 54 .

Java ve JMS bağ tanımları uygulamaları

ALW

Java ve JMS uygulamalarının uygulama adını nasıl sağlayabileceğine ilişkin bilgi için bkz. “Java ve JMS bağ tanımları uygulamaları” sayfa 54 .

İlgili kavramlar

Uygulama etkinliği izlemesi

Tek biçimli kümeler hakkında

İlgili başvurular

MQCNO

IBM i üzerinde MQCNO

Desteklenen programlama dillerinde uygulama adının kullanılması

Uygulama adının IBM MQ ' in desteklediği çeşitli dillerde nasıl seçildiğini öğrenmek için bu bilgileri kullanın.

Önerilen uygulama adı karakterleri

Uygulama adları, kuyruk yöneticisi alanının **CodedCharSetId** özniteliği tarafından belirtilen karakter kümesinde olmalıdır. Bu öznitelikle ilgili daha fazla bilgi için Kuyruk yöneticisine ilişkin özniteliklerbaşlıklı konuya bakın.

Ancak, uygulama IBM MQ MQI clientolarak çalışıyorsa, uygulama adı istemcinin karakter kümesinde ve kodlamasında olmalıdır.

Uygulama adının kuyruk yöneticileri arasında düzgün bir şekilde geçmesini sağlamak ve kaynak izleme konuları aracılığıyla uygulama kaynağı izlemesine izin vermek için, uygulama adları yalnızca tek baytlık yazdırılabilir karakterler içermelidir.

Notlar:

- Uygulama adlarında eğik çizgi ve ve ve imi karakterlerinin kullanılmasını da önlemelisiniz.
- V 9.3.0 Uygulama adlarında ve işareti karakteri kullanmaktan kaçınmalısınız. Ve imi içeren uygulama adları için sistem konusu STATAPP metrikleri üretilmez.

Bu, adı aşağıdaki şekilde sınırlar:

- Alfasayısal karakterler: `A-Z`, `a-z`ve `0-9`

  Not: EBCDIC Katakana kullanan sistemlerde uygulama adlarında küçük harfli `a-z` karakterlerini kullanmamalısınız.
- Boşluk karakteri
- EBCDIC ' de değişmeyen yazdırılabilir karakterler: `+ < = > % * ' ( ) , _ - . : ; ?`
- V 9.3.0 /karakteri. Adı eğik çizgi içeren bir uygulamaya ilişkin etkinlik izleme ya da STATAPP sistemi konu metriklerine abone olurken, eğik çizgi karakterlerini ve işareti karakteriyle değiştirmeniz gerekir. Örneğin, "DEPT1/APPS/STOCKQUOTE" adlı bir uygulamaya ilişkin STATAPP metriklerini almak için "$SYS/MQ/INFO/QMGR/QMBASIC/Monitor/STATAPP/DEPT1&APPS&STOCKQUOTE/INSTANCE" konu dizgisine abone olmanız gerekir. Amqsact ve amqsrua örnek uygulamaları, abonelikleri oluşturulurken otomatik olarak eğik çizgi karakterlerini ampersanlara dönüştürür.

Karakterleri ayarlama

Aşağıdaki çizelge, IBM MQ ' in desteklediği dillerde uygulama adının seçilmesini sağlayan araçları özetler. Adın seçildiği araç, öncelik sırasına göre, en yüksek öncelik sırasıdır.

	C bağ tanımları ve istemci	**Java bağ tanımları ve istemci**	**JMS bağ tanımları ve istemci**	**Yönetilen .NET istemcisi**	**Yönetilmeyen .NET bağ tanımları ve istemcisi**	**Yönetilen XMS istemcisi**	**Yönetilmeyen. XMS bağ tanımları ve istemcisi**
Bağlantı özelliği geçersiz kılma değeri		Java bağlantı özelliği geçersiz kılma değeri		.Net bağlantı özelliği geçersiz kılma değeri	.Net bağlantı özelliği geçersiz kılma değeri		
Geçersiz kılınan özellik		Java geçersiz kılınan özellik		.NET geçersiz kılınmış özelliği	.NET geçersiz kılınmış özelliği		
MQOrtamı		Java MQEnvironment		.NET MQEnvironment	.NET MQEnvironment		
Bağlantı üreticisi özelliği			Bağlantı üreticisi özelliği			Bağlantı üreticisi özelliği	Bağlantı üreticisi özelliği
JMSAdmin			JMSAdmin			JMSAdmin	JMSAdmin
MQCNO	Bağlanma seçenekleri						
Ortam değişkeni	Ortam değişkenleri				Ortam değişkenleri		Ortam değişkenleri
mqclient.ini (Yalnızca istemci bağlantıları için geçerlidir)	istemci bağlantıları				istemci bağlantıları		istemci bağlantıları
Java sınıf adı		Java sınıf adı	Java sınıf adı				
Varsayılan ad	Varsayılan ad			.NET Varsayılan adı	.NET Varsayılan adı	.NET Varsayılan adı	.NET Varsayılan adı

Not: C bağ tanımları ve istemci kolonu aşağıdaki programlama dilleri için de geçerlidir:

- COBOL
- Çevirici
- Visual Basic
- RPG

Programlama dili bağlantıları

C dilinde ve diğer programlama dillerinde IBM MQ kitaplıklarına çözülen uygulamalar, uygulama adını aşağıdaki şekillerde sağlayabilir.

Bağlantı yöntemleri, en yüksek olandan başlayarak öncelik sırasına göre listelenir.

Multi **Bağlanma seçenekleri**

- ALW MQCNO

  Not: z/OS Bir IBM MQ for z/OS kuyruk yöneticisine bağlanırken, uygulama adını ancak istemci kipi bağlantılarını kullanarak ya da IBM MQ classes for JMS ya da IBM MQ classes for Java uygulamalarını kullanarak ayarlayabilirsiniz.
- IBM i IBM iüzerinde MQCNO

ALW **Ortam değişkenleri**

Henüz bir uygulama adı seçmediyseniz, kuyruk yöneticisine yönelik bağlantıyı tanımlamak için **MQAPPLNAME** ortam değişkenini kullanabilirsiniz. Örneğin:

```
export MQAPPLNAME=ExampleApplName
```

Yalnızca ilk 28 karakterin kullanıldığını ve bu karakterlerin tümüyle boşluk ya da boş değer olmaması gerektiğini unutmayın.

Not: Öznitelik, yalnızca desteklenen programlama dilleri, yönetilmeyen .NETve yönetilmeyen XMS bağlantıları için geçerlidir.

ALW **İstemci yapılanış kütüğü**

Henüz bir uygulama adı seçmediyseniz ve bağlantı bir istemci bağlantıysa, kuyruk yöneticisiyle bağlantıyı tanımlamak için istemci yapılanış dosyasında (örneğin, `mqclient.ini`) aşağıdaki bilgileri belirtebilirsiniz.

```
Connection:
    ApplName=ExampleApplName
```

Notlar:

1. Yalnızca ilk 28 karakter kullanılır ve bu karakterlerin tümü boş ya da boş olmamalıdır.
2. Öznitelik yalnızca desteklenen programlama dilleri, yönetilmeyen .NETve yönetilmeyen XMS bağlantılarında istemci bağlantıları için geçerlidir.

Daha fazla bilgi için bkz. IBM MQ MQI client yapılandırma dosyası, `mqclient.ini`.

Varsayılan ad

Uygulama adını seçmeye devam etmezseniz, varsayılan ad kullanılmaya devam eder; bu ad, işletim sisteminin görüntülediği yol ve yürütülebilir adın çoğunu içerir. Daha fazla bilgi için bkz. PutApplAdı.

Yönetilen .NET uygulamaları

Yönetilen .NET uygulamaları, uygulama adını aşağıdaki şekillerde sağlayabilir.

Bağlantı yöntemleri, en yüksek olandan başlayarak öncelik sırasına göre listelenir.

Bağlantı özelliği geçersiz kılma değeri

Uygulamalara yönelik bağlantı ayrıntıları geçersiz kılma dosyasını aşağıdaki şekilde sağlayabilirsiniz:

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

`overrideConnectionDetailsFile` ile belirtilen dosya, öneki `mqj`olan özelliklerin bir listesini içerir. Uygulamaların **`mqj.APPNAME`** özelliğini tanımlaması gerekir; burada **`mqj.APPNAME`** özelliği, kuyruk yöneticisine yönelik bağlantıyı tanımlamak için kullanılan adı belirtir.

Adın yalnızca ilk 28 karakteri kullanılır. Örneğin:

```
mqj.APPNAME=ExampleApplName
```

Geçersiz Kılınan özellik

`MQC.APPNAME_PROPERTY` değişmezi *APPNAME*değeriyle tanımlandı. Artık bu özelliği, adın ilk 28 karakterini kullanarak **`MQQueueManager`** oluşturucusuna geçirebilirsiniz. Örneğin:

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

Daha fazla bilgi için bkz “.NET içinde yönetilen ve yönetilmeyen işlemler” sayfa 627.

MQEnvironment

AppName özelliği **`MQEnvironment`**sınıfına eklenir ve yalnızca ilk 28 karakter kullanılır. Örneğin:

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

Varsayılan ad

Uygulama adını, önceki metinde açıklanan yöntemlerden herhangi biriyle sağlamadıysanız, uygulama adı otomatik olarak yürütülebilir ad (ve sığacak yolun çoğu) olarak ayarlanır.

XMS uygulamalar

Bağlantı yöntemleri, en yüksek olandan başlayarak öncelik sırasına göre listelenir.

Bağlantı üreticisi özelliği

XMS uygulamaları, **`XMSC.WMQ_APPLICATIONNAME`** özelliğini kullanarak bağlantı üreticisine uygulama adını sağlayabilir ("*XMSC_WMQ_APPNAME*") JMS ' ye benzer bir şekilde. En çok 28 karakter belirtebilirsiniz.

Daha fazla bilgi için, bkz. “XMS .NET yönetilen nesneler oluşturma” sayfa 635 ve “XMS iletisinin özellikleri” sayfa 642.

JMSAdmin

Yönetim araçlarında özellik kısaca "**`APPLICATIONNAME`**" veya "**`APPNAME`**" olarak bilinir.

Java ve JMS bağ tanımları uygulamaları

Bağlantı yöntemleri, en yüksek olandan başlayarak öncelik sırasına göre listelenir.

ALW Java ve JMS istemci uygulamaları zaten bir uygulama adı belirtebilir ve bu IBM MQ for Multiplatforms üzerinde MQCNO **ApplName** alanını kullanarak bağ tanımlama uygulamalarına genişletildi.

Bağlantı özelliği geçersiz kılma değeri

Application name özelliği, geçersiz kılabileceğiniz bağlantı özellikleri listesine eklendi. Daha fazla bilgi için bkz. IBM MQ bağlantı özelliğini geçersiz kılma özelliğinin kullanılması.

Uyarı: Bağlantı özellikleri ve Bağlantı Özelliği Geçersiz Kılma dosyasını kullanma şekli hem IBM MQ classes for Java hem de .NETiçin aynıdır.

Geçersiz Kılınan özellik

MQC.APPNAME_PROPERTY değişmezi *APPNAME*değeriyle tanımlandı. Artık bu özelliği, adın ilk 28 karakterini kullanarak **MQQueueManager** oluşturucusuna geçirebilirsiniz. Daha fazla bilgi için bkz. IBM MQ classes for Javaiçinde bağlantı özelliği geçersiz kılma değerini kullanma.

MQEnvironment

AppName özelliği **MQEnvironment** sınıfına eklenir ve yalnızca ilk 28 karakter kullanılır.

Daha fazla bilgi için bkz "IBM MQ classes for Java için IBM MQ ortamını ayarlama" sayfa 360.

Java sınıf adı

Önceki metinde uygulama adını herhangi bir yolla belirtmediyseniz, uygulama adı ana sınıf adından türetilir.

Daha fazla bilgi için bkz "IBM MQ classes for Java için IBM MQ ortamını ayarlama" sayfa 360.

Uyarı: IBM i IBM i üzerinde ana sınıf adı sorgulanamıyacaktır, bu nedenle Java için IBM MQ client kullanılır.

İlgili kavramlar

"IBM MQ classes for Java için IBM MQ ortamını ayarlama" sayfa 360
Bir uygulamanın istemci kipinde bir kuyruk yöneticisine bağlanması için, uygulamanın kanal adını, anasistem adını ve kapı numarasını belirtmesi gerekir.

İlgili başvurular

MQCNO

IBM i üzerinde MQCNO

İletiler için tasarım teknikleri

Seçicilere ve ileti özelliklerine ilişkin dikkate alınacak noktalar da içinde olmak üzere, iletileri tasarlamanıza yardımcı olacak önemli noktalar.

Tasarım aşamasında dikkate alınacak şeyler

İletiyi kuyruğa koymak için MQI çağrısı kullandığınızda bir ileti yaratırsınız. Çağrıya giriş olarak, bir *ileti tanımlayıcısı* (MQMD) içinde bazı denetim bilgilerini ve başka bir programa göndermek istediğiniz verileri girersiniz. Ancak tasarım aşamasında, aşağıdakileri göz önünde bulundurmanız gerekir, çünkü bunlar mesajlarınızı oluşturma şeklinizi etkiler:

Kullanılacak ileti tipi

İleti gönderip başka bir işlem yapmayacağınız basit bir uygulama mı tasarlıyorsunuz? Yoksa bir soruya cevap mı istiyorsun? Bir soru soruyorsanız, ileti tanımlayıcısına yanıtı almak istediğiniz kuyruğun adını ekleyebilirsiniz.

İstek ve yanıt iletilerinizin zamanuyumlu olmasını istiyor musunuz? Bu, yanıtın isteğinize yanıt vermesi için bir zamanaşımı süresi belirlediğiniz anlamına gelir ve bu süre içinde yanıtı almazsanız, yanıt bir hata olarak işlenir.

Ya da zamanuyumsuz çalışmayı mı tercih edersiniz, böylece süreçleriniz ortak zamanlama sinyalleri gibi belirli olayların oluşmasına bağlı olmak zorunda kalmaz mı?

Diğer bir husus, tüm mesajlarınızın bir iş birimi içinde olup olmadığıdır.

İletilere farklı öncelikler atama

Her iletiye bir öncelik değeri atayabilir ve kuyruğun, iletilerini öncelik sırasına göre tutması için kuyruğu tanımlayabilirsiniz. Bunu yaparsanız, başka bir program kuyruktan bir ileti aldığında, her zaman en yüksek önceliğe sahip iletiyi alır. Kuyruk iletilerini öncelik sırasına göre tutmazsa, kuyruktan ileti alan bir program, iletileri kuyruğa eklendikleri sırayla alır.

Programlar, ileti kuyruğa konduğunda kuyruk yöneticisinin atadığı tanıtıcıyı kullanarak da bir ileti seçebilir. Diğer bir seçenek olarak, iletilerinizin her biri için kendi tanıtıcılarınızı oluşturabilirsiniz.

Yeniden başlatma kuyruk yöneticisinin iletiler üzerindeki etkisi

Kuyruk yöneticisi, gerektiğinde IBM MQ günlük dosyalarından yeniden başlatıldığında bunları kurtararak tüm kalıcı iletileri korur. Kalıcı olmayan iletiler ve geçici dinamik kuyruklar korunmaz. Atılmasını istemediğiniz iletiler yaratıldığında kalıcı olarak tanımlanmalıdır. AIX and Linux sistemlerinde IBM MQ for Windows ya da IBM MQ için bir uygulama yazarken, günlük dosyası sınırlarına göre çalışacak bir uygulama tasarlama riskini azaltmak için sisteminizin günlük dosyası ayırma ile ilgili olarak nasıl ayarlandığını bildiğinizden emin olun.

z/OS Paylaşılan kuyruklardaki iletiler (yalnızca IBM MQ for z/OS üzerinde kullanılabilir) Bağlaşım olanağı (CF) içinde tutulur, kalıcı olmayan iletiler, CF kullanılabilir olduğu sürece kuyruk yöneticisinin yeniden başlatılmasında korunur. CF başarısız olursa, kalıcı olmayan iletiler kaybolur.

İletilerin alıcısına kendinle ilgili bilgi verme

Genellikle, kuyruk yöneticisi kullanıcı kimliğini ayarlar, ancak uygun yetkili uygulamalar da bu alanı ayarlayabilir; böylece, kendi kullanıcı kimliğinizi ve alan programın muhasebe ya da güvenlik amacıyla kullanabileceği diğer bilgileri ekleyebilirsiniz.

Alma kuyruklarının miktarı

Multi Bir iletinin birden çok kuyruğa konması gerekiyorsa, bir konuyu ya da dağıtım listesini yayınlayabilirsiniz.

z/OS Bir iletinin birden çok kuyruğa konması gerekecekse, bir konuyu yayınlayabilirsiniz.

Seçiciler ve ileti özellikleri

İletiler, ana ileti bilgi yükünün yanı sıra bunlarla ilişkilendirilmiş meta veriler de olabilir. Bu ileti özellikleri, ek veri sağlanmasında yararlı olabilir.

Bu ek verinin, hakkında bilgi edinmek için önemli olan iki yönü vardır:

- Özellikler Advanced Message Security (AMS) korumasına tabi değildir. Verilerinizi korumak için AMS kullanmak istiyorsanız, bunu ileti özelliklerine değil, bilgi yüküne koyun.
- Özellikler, ileti seçimini gerçekleştirmek için kullanılabilir.

Seçicilerin kullanılmasının ilk başta standart ileti kuralını bozmasına dikkat etmek önemlidir. Kuyruk yöneticisi bu iş yükü için eniyilendiğinden, performans nedenleriyle karmaşık seçicilerin sağlanması önerilmez. Kuyruk yöneticisi, ileti özelliklerinin dizinlerini saklamaz, bu nedenle bir iletinin aranması doğrusal bir arama olmalıdır. Kuyruk ne kadar derin olursa, seçici o kadar karmaşık olur ve bir iletiyle eşleşen seçicinin performansı olumsuz etkileme olasılığı da o kadar düşük olur.

Karmaşık seçim gerekiyorsa, iletilere IBM Integration Busgibi herhangi bir uygulama ya da işleme motorunu farklı hedeflere kullanarak süzgeç uygulanması önerilir. Diğer bir seçenek olarak, konu sıradüzeninin kullanılması yararlı olabilir.

Not: IBM MQ classes for Java seçicilerin kullanımını desteklemez; seçicileri kullanmak istiyorsanız, bunların JMS API aracılığıyla yapılması gerekir.

Uygulama tasarımı ve performansıyla ilgili önemli noktalar

Kötü program tasarımının performansı etkilemesinin çeşitli yolları vardır. Programın kendisi iyi performans gösterebildiği, ancak diğer görevlerin başarımını etkilediği için bunları algılamak zor olabilir. Bu konuda, IBM MQ çağrıları yapan programlara özgü birkaç sorun açıklanmıştır.

Verimli uygulamalar tasarlamanıza yardımcı olacak birkaç fikir aşağıda verilmiştir:

- Uygulamanızı, bir kullanıcının düşünme süresiyle paralel olarak işlemeye devam eden şekilde tasarlayın:
  - Bir pano görüntüleyin ve uygulama başlatılırken kullanıcının yazmaya başlamasına izin verin.
  - Farklı sunuculardan paralel olarak gereksinim duyduğunuz verileri alın.
- Bağlantıları ve kuyrukları tekrar tekrar açmak, kapatmak, bağlamak ve bağlantısını kesmek yerine yeniden kullanacaksanız açık tutun.
- Ancak, tek bir ileti koyan bir sunucu uygulaması MQPUT1kullanmalıdır.
- Kuyruk yöneticileri, 4 KB ile 100 KB arasındaki iletiler için optimize edilmiştir. Çok büyük iletiler verimsiz; her biri 1 MB 'lik 100 ileti göndermek, tek bir 100 MB' lik iletiden daha iyi olabilir. Çok küçük mesajlar da verimsiz. Kuyruk yöneticisi, tek baytlık bir ileti için 4 KB ' lik iletiyle aynı miktarda iş yapar.
- İletilerinizi aynı anda kesinleştirilebilmeleri ya da yedeklenebilmeleri için bir iş birimi içinde tutun.
- Kurtarılabilir olması gerekmeyen iletiler için kalıcı olmayan seçeneği kullanın.
- Bir dizi hedef kuyruğa ileti göndermeniz gerekiyorsa, dağıtım listesi kullanmayı düşünün.

İleti uzunluğunun etkisi

Bir iletideki veri miktarı, iletiyi işleyen uygulamanın performansını etkileyebilir. Uygulamanızdan en iyi performansı elde etmek için yalnızca temel verileri bir iletiye gönderin. Örneğin, bir banka hesabının borçlandırılması isteğinde, istemciden sunucu uygulamasına iletilmesi gereken tek bilgi, hesap numarası ve banka hesabının tutarıdır.

İleti kalıcılığını etkisi

Kalıcı iletiler genellikle günlüğe kaydedilir. İletilerin günlüğe kaydedilmesi uygulamanızın performansını azaltır, bu nedenle yalnızca temel veriler için kalıcı iletiler kullanın. Bir iletideki veriler, kuyruk yöneticisi durursa ya da başarısız olursa atılabilirse, kalıcı olmayan bir ileti kullanın.

z/OS Kalıcı iletilere ilişkin MQPUT ve MQGET işlemleri, işlemleri kaydetmek için yeterli kurtarma günlüğü yeri olmadığında engellenir. Bu tür bir koşul, kuyruk yöneticisi iş günlüğünde CSQJ110E ve CSQJ111Ailetileriyle gösterilir. Bu tür koşulların yönetilmesi ve önlenmesi için izleme süreçlerinin gerçekleştirildiğinden emin olun.

Belirli bir iletiyi arama

MQGET çağrısı genellikle bir kuyruktan ilk iletiyi alır. Belirli bir iletiyi belirtmek için ileti tanımlayıcısında ileti ve ilinti tanıtıcılarını (*MsgId* ve *CorrelId*) kullanırsanız, kuyruk yöneticisi o iletiyi buluncaya kadar kuyrukta arama yapmak zorundadır. MQGET çağrısının bu şekilde kullanılması, uygulamanızın performansını etkiler.

Farklı uzunluktaki iletileri içeren kuyruklar

Uygulamanız sabit uzunluklu iletileri kullanamazsa, arabellekleri dinamik olarak büyütün ve tipik ileti boyutuna uyacak şekilde daraltın. Uygulama, arabellek çok küçük olduğu için başarısız olan bir MQGET çağrısı gönderirse, ileti verilerinin büyüklüğü döndürülür. Arabelleğin uygun şekilde yeniden boyutlandırılması ve MQGET çağrısına yeniden izin verilmesi için uygulamanıza kod ekleyin.

Not: `MaxMsgLength` özniteliğini belirtik olarak ayarlamazsanız, varsayılan olarak 4 MB ' ye ayarlanır; bu, uygulama arabellek boyutunu etkilemek için kullanılırsa çok verimsiz olabilir.

Eşitleme noktalarının sıklığı

Eşitleme noktası içinde çok sayıda MQPUT ya da MQGET çağrısı veren programlar, bunları kesinleştirmeden performans sorunlarına neden olabilir. Etkilenen kuyruklar şu anda erişilemez olan iletilerle doldurulabilirken, diğer görevler bu iletileri almak için bekliyor olabilir. Bunun depolama açısından ve ileti almaya çalışan görevlerle bağlantılı iş parçacıkları açısından etkileri vardır.

MQPUT1 çağrısının kullanılması

MQPUT1 çağrıyı yalnızca kuyruğa konacak tek bir iletiniz varsa kullanın. Birden çok ileti koymak istiyorsanız, MQOPEN çağrısının ardından bir dizi MQPUT çağrısı ve tek bir MQCLOSE çağrısı kullanın.

Kullanılmakta olan iş parçacığı sayısı

Windows IBM MQ for Windowsiçin, bir uygulama çok sayıda iş parçacığı gerektirebilir. Her kuyruk yöneticisi işlemine izin verilen uygulama iş parçacığı sayısı üst sınırı ayrılır.

Uygulamalar çok fazla iş parçacığı kullanabilir. Uygulamanın bu olasılığı dikkate alıp almadığını ve bu tür bir oluşumu durdurmak ya da raporlamak için eylemler gerçekleştirdiğini göz önünde bulundurun.

Kalıcı iletileri eşitleme noktası altına koy

Kalıcı iletiler konmalı ve uyumluluk noktası altına alınmalıdır. Bunun nedeni, eşitleme noktasının dışında kalıcı bir ileti alınırken, alma başarısız olursa, uygulamanın iletinin kuyruktan alınıp alınımadığını ve iletinin alınıp alınımadığını bilmesinin bir yolu yoktur ve ileti varsa, o zaman da kaybolmuştur. Eşitleme noktası altında kalıcı iletiler alınırken herhangi bir hata oluşursa, hareket geriye işlenir ve kalıcı ileti hala kuyrukta olduğu için kaybolmaz.

Benzer şekilde, kalıcı iletiler konurken bunları eşitleme noktası altına koyun. Sürekli iletileri eşitleme noktası altına koymanın ve almanın diğer bir nedeni, IBM MQ içindeki kalıcı ileti kodunun eşitleme noktası için büyük ölçüde eniyilenmiş olmasıdır. Yani sürekli mesajları senkronize noktası altına koymak ve almak, sürekli mesajları senkronize noktasının dışına koymaktan ve almaktan daha hızlıdır.

Uygulamanız kalıcı iletileri eşitleme noktasının dışına koyarsa, kuyruk yöneticisi uygulama adına örtük bir eşitleme noktası yaratıp yaratamayabileceğini denetler. Kuyruk yöneticisi bunu yapabiliyorsa, o eşitleme noktasının içine koyma işlemini içerir ve otomatik olarak kesinleştirir. Daha ayrıntılı açıklama için bkz. “Çoklu platformlarda örtük eşitleme noktası” sayfa 826 .

Ancak, IBM MQ içindeki kalıcı olmayan kod, eşitleme noktasının dışında olacak şekilde eniyilendiğinden, kalıcı olmayan iletileri eşitleme noktasının dışına koymak ve almak daha hızlıdır. Kalıcı ileti diskte kalıcı olarak saklandığından, kalıcı iletilerin yerleştirilmesi ve alıkonması disk hızlarına gider. Ancak, kalıcı olmayan iletileri koymak ve almak, syncpoint kullanırken bile disk yazmaya dahil olmadığı için CPU hızlarına gider.

Bir uygulama iletileri alıyorsa ve iletilerin kalıcı olup olmadığını önceden bilmiyorsa, GMO seçeneği MQGMO_SYNCPOINT_IF_PERSISTENT kullanılabilir.

Gelişmiş uygulamalar için tasarım teknikleri

Daha gelişmiş uygulamalar tasarlarken, ileti beklemek, yanıtları ilintilendirmek, bağlam bilgilerini ayarlamak ve kullanmak, uygulamaları otomatik olarak başlatmak, rapor oluşturmak ve kümelemeyi kullanırken ileti yakınlıklarını kaldırmak gibi göz önünde bulundurmak isteyebileceğiniz bazı teknikler vardır.

Basit bir IBM MQ uygulaması için, uygulamanızda hangi IBM MQ nesnelerini kullanacağınıza ve hangi ileti tiplerini kullanmak istediğinize karar vermeniz gerekir. Daha gelişmiş bir uygulama için, aşağıdaki bölümlerde tanıtılan tekniklerden bazılarını kullanmak isteyebilirsiniz.

İleti bekleniyor

Bir kuyruğa hizmet veren bir program aşağıdaki işlemleri yaparak iletileri bekleyebilir:

- Bir ileti gelinceye ya da belirli bir zaman aralığı sona erinceye kadar bekleme (bkz. “İleti bekleniyor” sayfa 764).
- z/OS Yalnızca IBM MQ for z/OS üzerinde, bir ileti geldiğinde programın bilgilendirilmesi için bir sinyal ayarlama. Daha fazla bilgi için bkz “Sinyal” sayfa 765.
- Bir ileti geldiğinde yönlendirilecek bir geri çağırma çıkışı oluşturma; bkz. “IBM MQ iletilerinin zamanuyumsuz tüketimi” sayfa 40.
- Bir iletinin gelip gelmediğini görmek için kuyrukta düzenli çağrılar yapma (*yoklama*). Bu genellikle tavsiye edilmez, çünkü performans etkileri olabilir.

Yanıtları ilintilendirilmesi

IBM MQ uygulamalarında, bir program bir iş yapması için istekte bulunan bir ileti aldığında, program genellikle istekte bulunana bir ya da daha çok yanıt iletisi gönderir.

İstekte bulunanın bu yanıtları özgün isteğiyle ilişkilendirmesine yardımcı olmak için bir uygulama, her iletinin tanımlayıcısında bir *ilinti tanıtıcısı* alanı ayarlayabilir. Programlar daha sonra, istek iletisinin ileti tanıtıcısını yanıt iletilerinin ilinti tanıtıcısı alanına kopyalar.

Bağlam bilgilerini ayarlama ve kullanma

Bağlam bilgileri , iletileri oluşturan kullanıcıyla ilişkilendirmek ve iletiyi oluşturan uygulamayı tanımlamak için kullanılır. Bu tür bilgiler, güvenlik, muhasebe, denetim ve sorun belirleme için yararlıdır.

Bir ileti yarattığınızda, kuyruk yöneticisinin varsayılan bağlam bilgilerini iletinizle ilişkilendirmesini isteyen bir seçenek belirtebilirsiniz.

Bağlam bilgilerini kullanma ve ayarlama hakkında daha fazla bilgi için bkz. “İleti bağlamı” sayfa 45.

IBM MQ programlarını otomatik olarak başlatma

İletiler kuyruğa geldiğinde programı otomatik olarak başlatmak için IBM MQ *triggering* komutunu kullanın.

Bir programın bu kuyruğu işlemeye başlaması için, kuyruktaki tetikleme koşullarını ayarlayabilirsiniz:

- Kuyruğa her ileti geldiğinde
- Kuyruğa ilk ileti geldiğinde
- Kuyruktaki ileti sayısı önceden tanımlanmış bir sayıya ulaştığında

Tetikleme hakkında daha fazla bilgi için bkz. “Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829. Tetikleme, bir programı otomatik olarak başlatmanın sadece bir yoludur. Örneğin,IBM MQ dışı olanakları kullanarak bir programı zamanlayıcıda otomatik olarak başlatabilirsiniz.

Multi Çoklu platformlarsistemlerinde IBM MQ , kuyruk yöneticisi başlatıldığında IBM MQ programlarını başlatmak için hizmet nesnelerini tanımlayabilir; bkz. Hizmet nesneleri.

IBM MQ raporları oluşturma

Bir uygulama içinde aşağıdaki raporları isteyebilirsiniz:

- Kural dışı durum raporları
- Süre bitimi raporları
- Varış sırasında onayla (COA) raporları
- Teslimatta onayla (COD) raporları
- Pozitif eylem bildirimi (PAN) raporları
- Negatif eylem bildirimi (NAN) raporları

Bunlar “Rapor iletileri” sayfa 19içinde açıklanmıştır.

Kümeler ve ileti yakınlıkları

Aynı kuyruk için birden çok tanımlama içeren kümeleri kullanmaya başlamadan önce, ilgili ileti alışverişi gerektirip gerektirmediğini görmek için uygulamalarınızı inceleyin.

Bir küme içinde, bir ileti, uygun kuyruk eşgörünümünü barındıran herhangi bir kuyruk yöneticisine yöneltilebilir. Bu nedenle, ileti yakınlıkları olan uygulamaların mantığı bozulabiliyor.

Örneğin, aralarında soru ve yanıt biçiminde akan bir dizi iletiye dayanan iki uygulamanız olabilir. Tüm soruların aynı kuyruk yöneticisine gönderilmesi ve tüm yanıtların diğer kuyruk yöneticisine geri gönderilmesi önemli olabilir. Bu durumda, iş yükü yönetimi yordamının iletileri, uygun kuyruğun bir eşgörünümünü barındıran herhangi bir kuyruk yöneticisine göndermemesi önemlidir.

Mümkünse, yakınlıkları kaldırın. İleti yakınlarının kaldırılması, uygulamaların kullanılabilirliğini ve ölçeklenebilirliğini artırır.

Daha fazla bilgi için bkz. İleti yakınlıkları.

IBM i uygulamaları için tasarım ve performans konuları

Uygulama tasarımının, iş parçacıklarının ve depolamanın performansı nasıl etkileyebileceğini anlamak için bu bilgileri kullanın.

Bu bilgiler iki bölüme ayrılır:

- “Uygulama tasarımıyla ilgili önemli noktalar” sayfa 60
- “Belirli performans sorunları” sayfa 61

Uygulama tasarımıyla ilgili önemli noktalar

Kötü program tasarımının performansı etkilemesinin çeşitli yolları vardır. Bu sorunların saptanması zor olabilir, çünkü program diğer görevlerin başarımını etkilerken iyi performans gösterir. IBM MQ for IBM i çağrısı yapan programlara özgü bazı sorunlar aşağıdaki bölümlerde açıklanmıştır.

Uygulama tasarımıyla ilgili daha fazla bilgi için bkz. “IBM MQ uygulamaları için tasarımla ilgili önemli noktalar” sayfa 47.

İleti uzunluğunun etkisi
IBM MQ for IBM i , iletilerin 100 MB ' ye kadar veri tutmasına izin verse de, bir iletideki veri miktarı, iletiyi işleyen uygulamanın performansını etkiler. Uygulamanızdan en iyi performansı elde etmek için, yalnızca bir iletideki temel verileri gönderin; örneğin, bir banka hesabının borçlandırılması isteğinde, istemciden sunucu uygulamasına iletilmesi gereken tek bilgi hesap numarası ve banka hesabının tutarıdır.

İleti kalıcılığını etkisi
Kalıcı iletiler günlüğe kaydedildi. İletilerin günlüğe kaydedilmesi uygulamanızın başarımını azaltır, bu nedenle yalnızca önemli veriler için kalıcı iletiler kullanın. Bir iletideki veriler, kuyruk yöneticisi durursa ya da başarısız olursa atılabilirse, kalıcı olmayan bir ileti kullanın.

Belirli bir iletiyi arama
MQGET çağrısı genellikle bir kuyruktan ilk iletiyi alır. Belirli bir iletiyi belirtmek için ileti tanımlayıcısında ileti ve ilinti tanıtıcılarını (*MsgId* ve *CorrelId*) kullanırsanız, kuyruk yöneticisi o iletiyi buluncaya kadar kuyruğu aramalıdır. MQGET çağrısının bu şekilde kullanılması, uygulamanızın performansını etkiler.

Farklı uzunluktaki iletileri içeren kuyruklar
Bir kuyruktaki iletiler farklı uzunluktaysa, bir iletinin boyutunu saptamak için, uygulamanız MQGET çağrıyı *BufferLength* alanı sıfır olarak ayarlanarak kullanabilir; böylece, çağrı başarısız olsa da, ileti verilerinin boyutunu döndürür. Daha sonra uygulama, ilk çağrısında ölçtüğü iletinin tanıtıcısını ve doğru büyüklükte bir arabelleği belirterek çağrıyı yineleyebilir. Ancak, aynı kuyruğa hizmet veren başka uygulamalar varsa, ikinci MQGET çağrısı, başka bir uygulamanın iki çağrı arasında geçen süre içinde aldığı bir iletiyi aramak için zaman harcadığı için uygulamanızın performansının azaldığını fark edebilirsiniz.

Uygulamanız değişmez uzunluklu iletileri kullanamazsa, bu sorunun başka bir çözümü, kuyruğun kabul edebileceği ileti büyüklüğü üst sınırını bulmak için MQINQ çağrısının kullanılmasıdır; daha sonra, MQGET çağrısında bu değeri kullanın. Bir kuyruğa ilişkin ileti büyüklüğü üst sınırı, kuyruğun **MaxMsgLen** özniteliğinde saklanır. Ancak bu kuyruk özniteliğinin değeri, 2 GB ' den büyük olabilecek IBM MQ for IBM itarafından izin verilen üst sınır olabileceği için, bu yöntem büyük miktarlarda depolama kullanabilir.

Eşitleme noktalarının sıklığı
Eşitleme noktası içinde çok sayıda MQPUT çağrısı veren programlar, bunları kesinleştirmeden performans sorunlarına neden olabilir. Etkilenen kuyruklar şu anda kullanılamayan iletilerle doldurulabilirken, diğer görevler bu iletileri almak için bekliyor olabilir. Bu sorunun depolama açısından ve ileti almaya çalışan görevlerle bağlantılı iş parçacıkları açısından etkileri vardır.

MQPUT1 çağrısının kullanılması
MQPUT1 çağrıyı yalnızca kuyruğa konacak tek bir iletiniz varsa kullanın. Birden çok ileti koymak istiyorsanız, MQOPEN çağrısının ardından bir dizi MQPUT çağrısı ve tek bir MQCLOSE çağrısı kullanın.

Kullanılmakta olan iş parçacığı sayısı
Bir uygulama birçok iş parçacığı gerektirebilir. Her kuyruk yöneticisi işlemine izin verilen iş parçacığı sayısı üst sınırı ayrılır. Bazı uygulamalar sorun çıkarıyorsa, tasarımlarının çok fazla iş parçacığı kullanmasından kaynaklanıyor olabilir. Uygulamanın bu olasılığı dikkate alıp almadığını ve bu tür bir oluşumu durdurmak ya da raporlamak için eylemler gerçekleştirdiğini göz önünde bulundurun. IBM i ' in izin verdiği iş parçacığı sayısı üst sınırı 4.095 'tir. Ancak, varsayılan değer 64 'tür. IBM MQ , işlemlerinin 63 'e kadar iş parçacığını kullanılabilir kılar.

Belirli performans sorunları

Bu bölümde, depolama ve düşük performans sorunları açıklanır.

Depolama sorunları
`CPF0907. Serious storage condition may exist` sistem iletisini alırsanız, IBM MQ for IBM i kuyruk yöneticileriyle ilişkili alanı dolduruyor olabilirisiniz.

Uygulamanız ya da IBM MQ for IBM i yavaş mı çalışıyor?
Uygulamanız yavaş çalışıyorsa, uygulamanın bir döngüde olduğunu ya da kullanılamayan bir kaynağı beklediğini gösterebilir. Bu yavaş çalışma bir performans sorunundan da kaynaklanıyor olabilir. Bunun nedeni sisteminizin kapasitesinin sınırlarına yakın çalışması olabilir. Bu tip bir sorun, genellikle sabah ve öğleden sonra en yoğun sistem yükleme zamanlarında en kötüsüdür. (Ağınız birden çok saat dilimine yayılırsa, en yüksek sistem yükü başka bir zamanda oluşmuş gibi görünebilir.)

Performans düşmesinin sistem yüküne bağlı olmadığını, ancak bazen sistem hafifçe yüklendiğinde, kötü tasarlanmış bir uygulama programının sorumlu olduğunu bulursanız. Bu sorun, yalnızca belirli kuyruklara erişildiğinde ortaya çıkan bir sorun olarak ortaya çıkabilir.

QTOTJOB ve QADLTOTJ, araştırılmaya değer sistem değerleridir.

Aşağıdaki belirtiler IBM MQ for IBM i ' in yavaş çalıştığını gösterebilir:

- Sisteminiz MQSC komutlarına yanıt vermekte yavaşsa.
- Kuyruk derinliğinin yinelenmesi, kuyruğun büyük miktarda kuyruk etkinliği beklediğiniz bir uygulama için yavaş işlendiğini gösterir.
- IBM MQ izleme çalışıyor mu?

Linux Linux on Power Systems - Little Endian uygulamaları için tasarımla ilgili önemli noktalar

Linux on Power Systems - Little Endian yalnızca 64 bitlik uygulamaları desteklediğinden, IBM MQ içinde 32 bitlik uygulamalar için destek sağlanmaz.

İlgili kavramlar
“IBM MQ uygulamaları için tasarımla ilgili önemli noktalar” sayfa 47
Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğine karar verdiğinizde, IBM MQtarafından sunulan özellikleri nasıl kullanacağınıza karar vermeniz gerekir.

z/OS uygulamaları için tasarım ve performans konuları

Uygulama tasarımı, performansı etkileyen en önemli faktörlerden biridir. Performansa dahil olan tasarım faktörlerinden bazılarını anlamak için bu konuyu kullanın.

Kötü program tasarımının performansı etkilemesinin çeşitli yolları vardır. Bu sorunların saptanması zor olabilir, çünkü program diğer görevlerin başarımını etkilerken iyi performans gösterir. MQI çağrıları yapan programlara özgü bazı sorunlar aşağıdaki kısımlarda gösterilmiştir.

Uygulama tasarımıyla ilgili daha fazla bilgi için bkz. "IBM MQ uygulamaları için tasarımla ilgili önemli noktalar" sayfa 47.

İleti uzunluğunun etkisi

IBM MQ for z/OS , iletilerin 100 MB ' ye kadar veri tutmasına izin verse de, bir iletideki veri miktarı, iletiyi işleyen uygulamanın performansını etkiler. Uygulamanızdan en iyi performansı elde etmek için yalnızca temel verileri bir iletiye gönderin. Örneğin, bir banka hesabının borçlandırılması isteğinde, istemciden sunucu uygulamasına iletilmesi gereken tek bilgi, hesap numarası ve borçlandırılacak tutardır.

İleti kalıcılığını etkisi

Kalıcı iletiler günlüğe kaydedilir. İletilerin günlüğe kaydedilmesi uygulamanızın performansını azaltır, bu nedenle yalnızca temel veriler için kalıcı iletiler kullanın. Bir iletideki veriler, kuyruk yöneticisi durursa ya da başarısız olursa atılabilirse, kalıcı olmayan bir ileti kullanın.

Kalıcı iletilere ilişkin veriler günlük arabelleklerine yazılır. Bu arabellekler, aşağıdaki durumda günlük veri kümelerine yazılır:

- Kesinleştirme gerçekleştiğinde
- Bir ileti, eşitleme noktasından alındı ya da çıktı
- WRTHRSH arabellekleri dolduruldu

Bir iş biriminde birden çok iletinin işlenmesi, iletilerin her iş birimi için bir tane işlenmesinden ya da zamanuyumlu noktanın dışında kalmasından daha az giriş/çıkışa neden olabilir.

Belirli bir iletiyi arama

MQGET çağrısı genellikle ilk iletiyi bir kuyruktan alır. İleti ve ilinti tanıtıcılarını kullanırsanız (**MsgId** ve **CorrelId**) Belirli bir iletiyi belirtmek için ileti tanımlayıcıda, kuyruk yöneticisi o iletiyi buluncaya kadar kuyrukta arama yapar. MQGET 'in bu şekilde kullanılması, belirli bir iletiyi bulmak için IBM MQ ' in tüm kuyruğu taraması gerekebileceğinden, uygulamanızın performansını etkiler.

Kuyruk yöneticisinin kuyruktaki MQGET işlemlerinin hızını artırmak için kullanılabilecek bir dizini korumasını istediğinizi belirtmek için **IndexType** kuyruk özniteliğini kullanabilirsiniz. Ancak, bir dizini korumak için küçük bir performans indirimi vardır, bu nedenle yalnızca kullanmanız gerekirse bir dizin oluşturun. İleti tanıtıcılarından ya da ilinti tanıtıcılarından oluşan bir dizin oluşturmayı seçebilir ya da iletilerin sıralı olarak alındığı kuyruklar için dizin oluşturmamayı seçebilirsiniz. Aynı değere sahip değil, birçok farklı anahtar değerine sahip olmaya çalışın. Örneğin, Balance1, Balance2ve Balance3, Dengeli üç değil. Paylaşılan kuyruklar için doğru **IndexType**olmalıdır. **IndexType** kuyruk özniteliğinin ayrıntıları için bkz. IndexType.

Dizinlenmiş kuyrukları kullanarak kuyruk yöneticisi yeniden başlatma süresini etkilemesini önlemek için, CSQ6SYSP makrosunda QINDXBLD (NOWAIT) parametresini kullanın. Bu, kuyruk yöneticisi yeniden başlatmanın, kuyruk dizini oluşturma işleminin tamamlanmasını beklemeden tamamlanmasını sağlar.

IndexType özniteliğinin ve diğer nesne özniteliklerinin tam açıklaması için bkz. Nesnelerin öznitelikleri.

Farklı uzunluktaki iletileri içeren kuyruklar

İletinin beklenen boyutuyla eşleşen bir arabellek boyutunu kullanarak bir ileti alın. İletinin çok uzun olduğunu gösteren dönüş kodunu alırsanız, daha büyük bir arabellek alın. Alma işlemi bu şekilde başarısız olduğunda, döndürülen veri uzunluğu, dönüştürülmemiş ileti verilerinin boyutudur. MQGET çağrısında MQGMO_CONVERT belirtirseniz ve dönüştürme sırasında veriler genişler, yine de arabelleğe sığmayabilir; bu durumda arabellek boyutunu daha da artırmanız gerekir.

MQGET ' i sıfır arabellek uzunluğuyla yayınlarsanız, iletinin boyutunu döndürür ve uygulama bu büyüklükte bir arabellek alabilir ve get komutunu yeniden verebilir. Kuyruğu işleyen birden çok uygulamanız varsa, özgün uygulama alma işlemini yeniden yaptığında başka bir uygulama iletiyi önceden işlemiş olabilir. Zaman zaman büyük iletileriniz varsa, yalnızca bu iletiler için büyük bir arabellek almanız ve ileti işlendikten sonra serbest bırakmanız gerekebilir. Bu, tüm uygulamalarda büyük arabellekler varsa sanal saklama alanı sorunlarının azaltılmasına yardımcı olur.

Uygulamanız sabit uzunluklu iletileri kullanamazsa, bu sorunun başka bir çözümü, kuyruğun kabul edebileceği ileti büyüklüğü üst sınırını bulmak için MQINQ çağrısının kullanılmasıdır ve MQGET çağrısında bu değeri kullanın. Bir kuyruğa ilişkin ileti büyüklüğü üst sınırı, kuyruğun **MaxMsgL** özniteliğinde saklanır. Bu yöntem büyük miktarlarda depolama kullanabilir; ancak **MaxMsgL** değeri, IBM MQ for z/OStarafından izin verilen üst sınır olan 100 MB ' ye kadar olabilir.

Not: Büyük iletiler kuyruğa yerleştirildikten sonra **MaxMsgL** değiştirgesini alçaltabilirsiniz. Örneğin, 100 MB ' lik bir ileti koyduktan sonra **MaxMsgL** değerini 50 bayt olarak ayarlayabilirsiniz. Bu, uygulamanın beklediğinden daha büyük iletiler alabileceği anlamına gelir.

Eşitleme noktalarının sıklığı

Syncpoint içindeki birçok MQPUT çağrısına bunları kesinleştirmeden yanıt veren programlar başarım sorunlarına neden olabilir. Etkilenen kuyruklar şu anda kullanılamayan iletilerle doldurulabilirken, diğer görevler bu iletileri almak için bekliyor olabilir. Bunun depolama açısından ve ileti almaya çalışan görevlerle bağlantılı iş parçacıkları açısından etkileri vardır.

Bir kuyruğu işleyen birden çok uygulamanız varsa, kural olarak genellikle aşağıdakilerden birine sahip olduğunuzda en iyi performansı elde edebilirsiniz:

- 100 kısa ileti (1 KB ' den az) ya da
- Daha büyük iletiler için bir ileti (100 KB)

her bir eşitleme noktası için. Kuyruğu işleyen tek bir uygulama varsa, her iş birimi için daha fazla iletiye sahip olmanız gerekir.

MAXUMSGS kuyruk yöneticisi özniteliğiyle bir görevin alabileceği ya da tek bir kurtarma birimine koyabileceği iletilerin sayısını sınırlayabilirsiniz. Bu öznitelikle ilgili bilgi için MQSC komutlarıiçindeki **ALTER QMGR** komutuna bakın.

MQPUT1 çağrısının avantajları

MQPUT1 çağrıyı yalnızca kuyruğa konacak tek bir iletiniz varsa kullanın. Birden çok ileti koymak istiyorsanız, MQOPEN çağrısının ardından bir dizi MQPUT çağrısı ve tek bir MQCLOSE çağrısı kullanın.

Bir kuyruk yöneticisi kaç ileti içerebilir

Yerel Kuyruklar

Bir kuyruk yöneticisinin tutabileceği yerel ileti sayısı, temel olarak sayfa kümelerinin boyutudur. En çok 100 sayfa kümeniz olabilir (önerilen sayfa kümesi 0 ve sayfa kümesi 1, sistemle ilgili nesneler ve kuyruklar içindir). Genişletilmiş biçimde bir sayfa kümesi kullanabilir ve bir sayfa kümesinin kapasitesini artırabilirsiniz.

Paylaşılan Kuyruklar

Paylaşılan kuyrukların sığası, bağlaşım olanağının (CF) büyüklüğüne bağlıdır. IBM MQ , temel depolama birimlerinin giriş ve öğe olduğu CF listesi yapılarını kullanır. Her ileti, ilişkili MQMD ' yi ve diğer ileti verilerini içeren 1 giriş ve birden çok öğe olarak saklanır. Tek bir ileti tarafından kullanılan öğelerin sayısı, iletinin boyutuna ve CFLEVEL (5) için MQPUT zamanında yürürlükte olan boşaltma kurallarına bağlıdır. İleti verileri Db2 ya da SMDS ' ye boşaltıldığında daha az öğe gerekir. İleti boşaltıldığında ileti verileri erişimi daha yavaş olur. İleti boşaltmayla ilişkili başarım ve CPU ek yükünün daha fazla karşılaştırılması için bkz. Performance Supportpac MP1H .

Performansı etkileyen

Performans, iletilerin ne kadar hızlı işlenebileceği ve ileti başına ne kadar CPU gerektiği anlamına da gelebilir.

İletilerin ne kadar hızlı işlenebildiğini etkileyen

Kalıcı iletiler için en büyük etki, günlük veri kümelerinin hızıdır. Günlük veri kümelerinin hızı, bulundukları DASD ' ye bağlıdır. Bu nedenle, çekişmeyi azaltmak için günlük verileri kümesini düşük kullanılmış birimlere yerleştirmeye dikkat edilmelidir. MQ günlüklerinin paylaştırma işlemi, G/Ç başına yazılan birden çok sayfa olduğunda günlük başarımını artırır. Z Yüksek Başarımlı Fiber Bağlantı (zHPF), G/Ç altsistemi meşgul olduğunda G/Ç yanıt süresi için önemli bir başarıma sahiptir.

Bir iletiyi alma ve yerleştirme isteği olduğunda, kuyruğa erişim, kuyruğun bütünlüğünü koruma isteği sırasında kilitlenir. Planlama amacıyla, isteğin tamamı için kilitli olan kuyruğu göz önünde bulundurun. Yani bir koyma süresi 100 mikrosaniye ise ve saniyede 10.000 'den fazla isteğiniz varsa, gecikmeler yaşayabilirsiniz. Pratikte bundan daha iyisini elde edebilirsiniz, ama bu iyi bir genel kuraldır. Başarımı artırmak için farklı kuyruklar kullanabilirsiniz.

Bunun olası nedenleri şunlar olabilir:

- Her CICS hareketinin kullandığı ortak bir yanıt kuyruğunu kullanma
- Her CICS hareketine kuyruğa benzersiz bir yanıt verilir
- CICS bölgesine ilişkin bir kuyruğa yanıt ve CICS bölgesindeki tüm hareketler bu kuyruğu kullanır.

Yanıt, bir saniyenin istek sayısına ve isteklerin yanıt süresine bağlıdır.

İletilerin bir sayfa kümesinden okunması gerekirse, iletiler arabellek havuzundaki iletilerle karşılaştırıldığında daha yavaş olur. Arabellek havuzuna sığmayacak kadar çok iletiniz varsa, bunlar diske dökülacaktır. Bu nedenle, arabellek havuzunun kısa ömürlü mesajlarınız için yeterince büyük olduğundan emin olmanız gerekir. Çok saat sonra işlediğiniz iletileriniz varsa, bunlar diske dökülecekleri için, bu iletilere ilişkin bir alma işleminin arabellek havuzundaki iletilerden daha yavaş olmasını beklemelisiniz.

Paylaşılan bir kuyruk için, iletilerin hızı Coupling Facility 'nin hızına bağlıdır. Fiziksel işlemci içindeki bir CF, dış CF ' den daha hızlı olabilir. CF yanıt süresi, CF ' nin ne kadar meşgul olduğuna bağlıdır. Örneğin Hursley sistemlerinde, CF %17 meşgul olduğunda yanıt süresi 14 mikrosaniye idi. CF %95 meşgul olduğunda yanıt süresi 45 mikrosaniye idi.

MQ istekleriniz çok fazla CPU kullanıyorsa, bu, iletilerin ne kadar hızlı işlendiğini etkileyebilir. Mantıksal Bölüm (LPAR) CPU için kısıtlandığından, uygulamalar CPU ' yu beklerken geciktirilir.

İleti başına CPU miktarı

Genel olarak daha büyük iletiler daha fazla CPU kullanır, bu nedenle mümkünse büyük (x MB) iletilerden kaçının.

Kuyruklardan belirli iletiler alınırken, kuyruk yöneticisinin doğrudan iletiye gidebilmesi için kuyruk dizinlenmelidir (bu nedenle, kuyruğun tüm taramasından kaçınır). Kuyruk dizinlenmezse, kuyruk baştan taranır ve iletiyi arar. Kuyrukta 1000 ileti varsa, 1000 iletinin tümünü taraması gerekebilir. Sonuç çok fazla gereksiz CPU kullanımı.

TLS kullanan kanallar, iletinin şifrelenmesi nedeniyle ek bir maliyete sahiptir.

MQ V7 ' de, **CORRELID** ya da **MSGID**ek olarak, iletileri seçici dizgisiyle de seçebilirsiniz. Her mesajın aranması gerekir, bu yüzden kuyrukta çok mesaj varsa bu pahalıdır.

Bir uygulamanın OPEN KOYMA CLOSE işlemini PUT1 PUT1yerine yapması daha verimlidir.

CICS içinde tetikleme

Tetiklenen bir kuyruk için ileti geliş hızı düşük olduğunda, önce tetikleyicinin kullanılması etkili olur. İleti varış hızı saniyede 10 iletiden fazlaysa, ilk işlemi tetiklemek daha verimli olur, daha sonra işlemin bir iletiyi işlemesini ve sonraki iletiyi almasını sağlar ve bu şekilde devam eder. Kısa bir süre içinde bir ileti gelmediyse (örneğin, 0.1 ile 1 saniye arasında), işlem sona erer. Yüksek verim hızında, iletileri işlemek ve iletilerin oluşturulmasını önlemek için birden çok işlemin çalıştırılmasını gerektirebilirsiniz. Üretilen her tetikleyici ileti için bu, bir koyma ve bir tetikleme iletisi alma gerektirir; bu da iletinin maliyetini iki katına çıkartır.

Desteklenen bağlantı ya da eşzamanlı kullanıcı sayısı

Her bağlantı, kuyruk yöneticisi içindeki sanal saklama alanını kullanır; bu nedenle, koşutzamanlı kullanıcılar ne kadar çok saklama alanı kullanırsa, o kadar çok saklama alanı kullanılır. Çok büyük bir arabellek havuzuna ve çok sayıda kullanıcıya gereksinim duyarsanız, sanal saklama alanı için kısıtlanmış olabilirsiniz ve arabellek havuzlarınızın büyüklüğünü azaltmanız gerekebilir.

Güvenlik kullanılıyorsa, kuyruk yöneticisi uzun bir süre kuyruk yöneticisindeki bilgileri önbelleğe alır. Kuyruk yöneticisi içinde kullanılan sanal saklama alanı miktarı etkilenir.

CHINIT yaklaşık 10.000 bağlantıyı destekleyebilir. Bu, sanal saklama alanıyla sınırlıdır. Bir bağlantı, örneğin TLS kullanılarak daha fazla depolama kullanıyorsa, bağlantı başına depolama artar, bu da **CHINIT** ' in daha az bağlantıyı destekleyebileceği anlamına gelir. Büyük iletileri işlerken, bunlar **CHINIT**içindeki arabellekler için daha fazla depolama gerektirir; böylece, **CHINIT** daha az iletiyi destekleyebilir.

Uzak kuyruk yöneticisine yönelik bağlantılar istemci bağlantılarından daha verimlidir. Örneğin, her MQ istemcisi isteği için iki ağ akışı gerekir (biri istek için, diğeri yanıt için). Uzak kuyruk yöneticisine bir kanal gönderildiğinde, yanıt geri gelmeden önce ağ üzerinden 50 kişi gönderilebilir. Büyük bir istemci ağı düşünüyorsanız, dağıtılmış bir kutuda yoğunlaştırıcı kuyruk yöneticisini kullanmak daha verimli olabilir ve yoğunlaştırıcının içine ve dışına tek bir kanal gelebilir.

Performansı etkileyen diğer şeyler

Günlük veri kümesi en az 1000 silindir boyutunda olmalıdır. Günlükler bundan küçükse, denetim noktası etkinliği çok sık olabilir. Meşgul bir sistemde denetim noktası genellikle her 15 dakikada bir ya da daha uzun olmalıdır, çok yüksek çıkışlarda bundan daha az olabilir. Bir denetim noktası oluştuğunda, arabellek havuzları taranır ve 'eski' iletiler ve değiştirilen sayfalar diske yazılır. Denetim noktaları çok sıksa, bu başarımı etkileyebilir. LOGLOAD değeri, denetim noktası sıklığını da etkileyebilir. Kuyruk yöneticisi olağandışı bir şekilde sona ererse, yeniden başlatma sırasında 3 denetim noktasına geri okunması gerekebilir. En iyi denetim noktası aralığı, bir denetim noktası alındığında etkinlik ile kuyruk yöneticisi yeniden başlatıldığında okunması gereken günlük verileri miktarı arasındaki dengedir.

Bir kanal başlatılırken oluşan önemli bir yük var. Genellikle kanalın sık sık başlatılması ve durması yerine, bir kanalı başlatmak ve bağlı bırakmak daha iyidir.

İlgili bilgiler

MP1K: IBM MQ for z/OS 9.0 Performans Raporu

z/OS IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları

Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

- IMS uygulamalarında eşitleme noktalarını ve MQI çağrılarını kullanmak için bkz. “IMS uygulamalarının IBM MQ kullanılarak yazılması” sayfa 66.

- IBM MQ - IMS köprüsünü kullanan uygulamalar yazmak için bkz. “IMS köprü uygulamalarının yazılması” sayfa 70.

IBM MQ for z/OSüzerinde IMS ve IMS köprü uygulamaları hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “IMS uygulamalarının IBM MQ kullanılarak yazılması” sayfa 66
- “IMS köprü uygulamalarının yazılması” sayfa 70

İlgili kavramlar

“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

“Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

“Kuyruktaki iletilerin konması” sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

“Kuyruktan ileti alma” sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

“Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

“Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

“MQI ve kümelerle çalışma” sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

“IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

z/OS IMS uygulamalarının IBM MQ kullanılarak yazılması

IMS uygulamalarında IBM MQ kullanılırken hangi MQ API çağrılarının kullanılabileceğini ve uyumluluk noktası için kullanılan mekanizmayı içeren başka noktalar da vardır.

IBM MQ for z/OSüzerinde IMS uygulamaları yazma hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “IMS uygulamalarında eşitleme noktaları” sayfa 67
- “IMS uygulamalarında MQI çağrıları” sayfa 67

Sınırlamalar

IMS bağdaştırıcısını kullanan bir uygulama tarafından kullanılan IBM MQ API çağrılarına ilişkin kısıtlamalar vardır.

Aşağıdaki IBM MQ API çağrıları, IMS bağdaştırıcısını kullanan bir uygulamada desteklenmez:

- MQCB
- MQCB_FUNCTION
- MQCTL

İlgili kavramlar

"IMS köprü uygulamalarının yazılması" sayfa 70
Bu konuda, IBM MQ - IMS köprüsünü kullanmak üzere uygulamalar yazılmasına ilişkin bilgiler yer alır.

z/OS *IMS uygulamalarında eşitleme noktaları*

Bir IMS uygulamasında, IOPCB ve CHKP (denetim noktası) için GU (benzersiz olsun) gibi IMS çağrılarını kullanarak bir eşitleme noktası oluşturabilirsiniz.

Önceki denetim noktasından bu yana yapılan tüm değişiklikleri geri almak için IMS ROLB (rollback) çağrıyı kullanabilirsiniz. Daha fazla bilgi için IMS belgelerinde ROLB call başlıklı konuya bakın.

Kuyruk yöneticisi, iki aşamalı bir kesinleştirme protokolünün katılımcısıdır; eşgüdümcü IMS syncpoint yöneticisidir.

Tüm açık tutamaçlar, IMS bağdaştırıcısı tarafından bir eşitleme noktasında (toplu ya da ileti odaklı olmayan BMP ortamı dışında) kapatılır. Bunun nedeni, MQPUT ya da MQGET çağrıları yapıldığında değil, MQCONN, MQCONNX ve MQOPEN çağrıları yapıldığında farklı bir kullanıcı sonraki iş birimini başlatabilir ve IBM MQ güvenlik denetimi gerçekleştirilir.

Ancak, bir WFI (Wait-for-Input) ya da pseudo Wait-for-Input (PWFI) ortamında, bir sonraki ileti gelinceye ya da uygulamaya bir QC durum kodu dönünceye kadar tanıtıcıları kapatmaları IMS bildirilmez IBM MQ . Uygulama IMS bölgesinde bekliyorsa ve bu tanıtıcılardan herhangi biri tetiklenen kuyruklara aitse, kuyruklar açık olduğu için tetikleme gerçekleşmez. Bu nedenle, bir WFI ya da PWFI ortamında çalışan uygulamaların, sonraki ileti için GU 'yu IOPCB' ye yapmadan önce kuyruk tanıtıcılarını belirtik olarak MQCLOSE olarak belirtmeleri gerekir.

Bir IMS uygulaması (BMP ya da MPP) MQDISC çağrısını yaparsa, açık kuyruklar kapanır, ancak örtük eşitleme noktası alınmaz. Uygulama olağan bir şekilde sona ererse, açık kuyruklar kapatılır ve örtük kesinleştirme gerçekleşir. Uygulama olağandışı bir şekilde sona ererse, açık kuyruklar kapanır ve örtük bir geriletme oluşur.

z/OS *IMS uygulamalarında MQI çağrıları*

Sunucu uygulamalarında ve Sorgu uygulamalarında MQI çağrılarının kullanımı hakkında bilgi edinmek için bu bilgileri kullanın.

Bu kısım, aşağıdaki IMS uygulaması tiplerinde MQI çağrılarının kullanımını kapsar:

- "Sunucu uygulamaları" sayfa 67
- "Sorgu uygulamaları" sayfa 69

Sunucu uygulamaları

MQI sunucusu uygulama modelinin anahattı:

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

```
.
END
```

CSQ4ICB3 örnek programı, bu modeli kullanan bir BMP ' nin somutlamasını C/370içinde gösterir. Program önce IMS ile, sonra da IBM MQile iletişim kurar:

```
main()
----
Call InitIMS
If IMS initialization successful
Call InitMQM
If IBM MQ initialization successful
Call ProcessRequests
Call EndMQM
End-if
End-if

Return
```

IMS kullanıma hazırlama, programın ileti odaklı bir BMP olarak mı, yoksa toplu iş odaklı bir BMP olarak mı çağrılacağını belirler ve IBM MQ kuyruk yöneticisi bağlantısını ve kuyruk tanıtıcılarını uygun şekilde denetler:

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

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

Return to calling function
```

IBM MQ kullanıma hazırlama, kuyruk yöneticisine bağlanır ve kuyrukları açar. İleti odaklı bir BMP 'de bu, her IMS eşitleme noktası alındıktan sonra çağrılır; toplu iş odaklı BMP' de bu yalnızca program başlatılırken çağrılır:

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

```
Else
Initialize error message
Build 'connect failed' message
Call ReportCallError to output the message
Set return code to error
End-if

Return to calling function
```

MPP 'de sunucu modelinin uygulanması, MPP' nin her çağırma için tek bir iş birimini işlemesi gerçeğinden etkilenir. Bunun nedeni, bir eşitleme noktası (GU) alındığında, bağlantı ve kuyruk tanıtıcıları kapatılır ve sonraki IMS iletisi teslim edilir. Bu sınırlama, aşağıdakilerden biri tarafından kısmen aşılabilir:

- **Tek bir iş birimi içindeki birçok iletinin işlenmesi**

  Bu şunları içerir:

  – İleti okuma

  – Gerekli güncellemeler işleniyor

  – Yanıt konuyor

  Tüm iletiler işleninceye kadar ya da ileti sayısı üst sınırı belirleninceye kadar bir döngüde. Bu durumda bir eşitleme noktası alınır.

  Yalnızca belirli uygulama tiplerine (örneğin, basit bir veritabanı güncellemesi ya da sorgusu) bu şekilde yaklaşılabilir. MQI yanıt iletileri, işlenmekte olan MQI iletisinin yaratıcısının yetkisiyle birlikte konabilse de, IMS kaynak güncellemelerinin güvenlik etkilerinin dikkatli bir şekilde ele alınması gerekir.
- **MPP çağrısı başına bir iletinin işlenmesi ve kullanılabilir tüm iletileri işlemek için MPP ' nin birden çok zamanlanmasının sağlanması.**

  MPP hareketini, IBM MQ kuyruğunda ileti olduğunda ve ona hizmet eden uygulama olmadığında zamanlamak için IBM MQ IMS tetikleyici izleme programını (CSQQTRMN) kullanın.

  Tetikleyici izleyicisi MPP ' yi başlatırsa, kuyruk yöneticisi adı ve kuyruk adı aşağıdaki COBOL kod alımlarında gösterildiği gibi programa geçirilir:

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

Uzun süreli bir görev olması beklenen sunucu modeli, bir toplu işleme bölgesinde daha iyi desteklenir, ancak BMP CSQQTRMN kullanılarak tetiklenemez.

Sorgu uygulamaları

Sorgu ya da güncelleme başlatan tipik bir IBM MQ uygulaması aşağıdaki gibi çalışır:

- Kullanıcıdan veri topla
- Bir ya da daha fazla IBM MQ iletisi koy

- Yanıt iletilerini al (bunları beklemeniz gerekebilir)
- Kullanıcıya bir yanıt sağlayın

IBM MQ kuyruklarına konan iletiler, kesinleştirilinceye kadar diğer IBM MQ uygulamaları tarafından kullanılamayacağından, bunların ya uyumluluk noktasından dışarı konması ya da IMS uygulamasının iki harekete bölünmesi gerekir.

Sorgu tek bir ileti koymayı içeriyorsa, *eşitleme noktası yok* seçeneğini kullanabilirsiniz; ancak sorgu daha karmaşıksa ya da kaynak güncellemeleri söz konusu ise, hata oluşursa ve eşitleme kullanmazsanız tutarlılık sorunları alabilirsiniz.

Bunun üstesinden gelmek için, bir programdan programa ileti anahtarı kullanarak MQI çağrılarını kullanarak IMS MPP hareketlerini bölebilirsiniz; bununla ilgili bilgi için *IMS Sistemlerarası İletişim (ISC)* başlıklı konuya bakın. Bu, bir sorgu programının MPP ' de uygulanmasını sağlar:

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

z/OS IMS köprü uygulamalarının yazılması

Bu konuda, IBM MQ - IMS köprüsünü kullanmak üzere uygulamalar yazılmasına ilişkin bilgiler yer alır.

IBM MQ - IMS köprüsü hakkında bilgi için bkz. IMS köprüsü.

IBM MQ for z/OSüzerinde IMS köprü uygulamalarının yazılmasına ilişkin daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “IMS köprüsü iletileri nasıl ele alır?” sayfa 70
- “IMS hareket programlarını IBM MQ aracılığıyla yazma” sayfa 873

İlgili kavramlar

“IMS uygulamalarının IBM MQ kullanılarak yazılması” sayfa 66
IMS uygulamalarında IBM MQ kullanılırken hangi MQ API çağrılarının kullanılabileceğini ve uyumluluk noktası için kullanılan mekanizmayı içeren başka noktalar da vardır.

z/OS *IMS köprüsü iletileri nasıl ele alır?*

Bir IMS uygulamasına ileti göndermek için IBM MQ - IMS köprüsünü kullandığınızda, iletilerinizi özel bir biçimde oluşturmanız gerekir.

İletilerinizi, hedef IMS sisteminin XCF grubunu ve üye adını belirten bir depolama sınıfıyla tanımlanmış IBM MQ kuyruklarına da koymanız gerekir. Bunlar MQ-IMS köprü kuyrukları ya da yalnızca **köprü** kuyrukları olarak bilinir.

IBM MQ-IMS köprüsü, QSGDISP (QMGR) ile tanımlandıysa ya da NOSHARE seçeneğiyle birlikte QSGDISP (SHARED) ile tanımlandıysa, köprü kuyruğuna dışlayıcı giriş erişimi (MQOO_INPUT_EXCLUSIVE) gerektirir.

Bir kullanıcının IMS uygulamasına ileti göndermeden önce IMS ' da oturum açması gerekmez. Güvenlik denetimi için MQMD yapısının *UserIdentifier* alanındaki kullanıcı kimliği kullanılır. Denetim düzeyi, IBM MQ IMSile bağlantı kurduğunda belirlenir ve IMS köprüsü için uygulama erişim denetimiiçinde açıklanır. Bu, sözde oturum açma gerçekleştirilmesini sağlar.

IBM MQ - IMS köprüsü aşağıdaki ileti tiplerini kabul eder:

- IMS hareket verilerini ve MQIIH yapısını içeren iletiler (MQIIH içinde açıklanmıştır):

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

Not:

1. Köşeli ayraç, [], isteğe bağlı çoklu kesimleri temsil eder.
2. MQIIH yapısını kullanmak için MQMD yapısının *Format* alanını MQFMT_IMS olarak ayarlayın.

- IMS hareket verilerini içeren, ancak MQIIH yapısını içermeyen iletiler:

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

IBM MQ , ileti verilerinin geçerliliğini denetleyerek, LL byte 'ları toplamının yanı sıra MQIIH ' nin uzunluğunun (varsa) ileti uzunluğuna eşit olmasını sağlar.

IBM MQ - IMS köprüsü, köprü kuyruklarından ileti aldığında, bu iletileri aşağıdaki gibi işler:

- İleti bir MQIIH yapısı içeriyorsa, köprü MQIIH 'yi doğrular (bkz. MQIIH), OTMA üstbilgilerini oluşturur ve iletiyi IMS' e gönderir. Giriş iletisinde hareket kodu belirtildi. Bu bir LTERM ise, IMS bir DFS1288E iletisiyle yanıt gönderir. Hareket kodu bir komutu gösteriyorsa, IMS komutu yürütür; tersi durumda, ileti hareket için IMS içinde kuyruğa alınır.
- İleti IMS işlem verileri içeriyorsa, ancak MQIIH yapısı yoksa, IMS köprüsü aşağıdaki varsayımları yapar:
  - Hareket kodu, kullanıcı verilerinin 5-12 bayttır.
  - Hareket, etkileşimli olmayan kipte
  - Hareket, kesinleştirme kipi 0 'da (kesinleştirme sonrası gönderme)
  - MQMD ' deki *Format* , *MFSMapName* (girişte) olarak kullanılır
  - Güvenlik kipi MQISS_CHECK

  Yanıt iletisi MQIIH yapısı olmadan da oluşturulur ve IMS çıkışının *MFSMapName* ürününden MQMD için *Format* kullanılır.

IBM MQ - IMS köprüsü, her IBM MQ kuyruğu için bir ya da iki Tpipo kullanır:

- Synchronized Tpipe, Kesinleştirme kipi 0 (COMMIT_THEN_SEND) kullanan tüm iletiler için kullanılır (bunlar IMS /DIS TMEMBER istemcisi TPIPE xxxx komutunun durum alanında SYN ile gösterilir)
- Kesinleştirme kipi 1 'i (SEND_THEN_COMMIT) kullanan tüm iletiler için eşitlenmemiş bir Tpipe kullanılır

Tpipo 'lar, ilk kullanıldığında IBM MQ tarafından oluşturulur. IMS yeniden başlatılıncaya kadar eşitlenmemiş bir Tpipe var. Eşitlenmiş Tpipo 'lar, IMS soğuk başlatılıncaya kadar var olur. Bu Tpipo ' ları kendiniz silemezsiniz.

IBM MQ - IMS köprüsünün iletilerle nasıl ilgilendiğine ilişkin ek bilgi için aşağıdaki konulara bakın:

- “IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi” sayfa 72
- “İleti IMS kuyruğuna konamazsa” sayfa 72
- “IMS köprü geribildirim kodları” sayfa 73
- “IMS köprüsündeki iletilerdeki MQMD alanları” sayfa 73
- “IMS köprüsündeki iletilerdeki MQIIH alanları” sayfa 74
- “IMS ' den yanıt iletileri” sayfa 75
- “IMS hareketlerinde alternatif yanıt PCB ' lerinin kullanılması” sayfa 75

- “IMS ' dan istenmeyen iletiler gönderme” sayfa 75
- “İleti bölümlemesi” sayfa 76
- “IMS köprüsüne/köprüsünden gelen iletiler için veri dönüştürme” sayfa 76

İlgili kavramlar

“IMS hareket programlarını IBM MQ aracılığıyla yazma” sayfa 873
IMS hareketlerini IBM MQ aracılığıyla işlemek için gereken kodlama, IMS işleminin gerektirdiği ileti biçimine ve döndürebileceği yanıt aralığına bağlıdır. Ancak, uygulamanız IMS ekran biçimlendirme bilgilerini işlerken göz önünde bulundurulması gereken birkaç nokta vardır.

z/OS *IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi*

IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesini açıklayan bir tablo.

Çizelge 4. IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi

IBM MQ ileti tipi	**Commit-then-send (kip 0)- eşitlenmiş IMS Tpipes kullanır**	**Send-then-commit (kip 1)- eşitlenmemiş IMS Tpipes kullanır**
Kalıcı IBM MQ iletileri	• Kurtarılabilir tam işlevli hareketler • Kurtarılamaz hareketler IMS tarafından reddedilir	• Fastpath işlemleri • Etkileşimli hareketler • Tam işlevli hareketler
Kalıcı olmayan IBM MQ iletileri	• Kurtarılamaz tam işlevli hareketler • IMS V8 ve APAR PQ61404 ve IMS ürününün sonraki tüm sürümleriyle kurtarılabilir hareketlere izin verilir.	• Fastpath işlemleri • Etkileşimli hareketler • Tam işlevli hareketler

Not: IMS komutları, kesinleştirme kipi 0 olan kalıcı IBM MQ iletilerini kullanamaz. Ek bilgi için Kesinleştirme kipi (commitMode) konusuna bakın.

z/OS *İleti IMS kuyruğuna konamazsa*

İleti IMS kuyruğuna konamazsa yapılacak işlemlerle ilgili bilgi edinin.

İleti IMS kuyruğuna konamazsa, IBM MQtarafından aşağıdaki işlem kullanılır:

- İleti geçersiz olduğu için bir ileti IMS ' e konamazsa, ileti teslim edilmeyen ileti kuyruğuna yerleştirilir ve sistem konsoluna bir ileti gönderilir.
- İleti geçerliyse, ancak IMStarafından reddedildiyse, IBM MQ sistem konsoluna bir hata iletisi gönderirse, ileti IMS algılama kodunu içerir ve IBM MQ iletisi teslim edilmeyen ileti kuyruğuna yerleştirilir. IMS algılama kodu 001Aise, IMS yanıt kuyruğuna hatanın nedenini içeren bir IBM MQ iletisi gönderir.

  Not: Daha önce listelenen durumlarda, IBM MQ iletiyi herhangi bir nedenle teslim edilmeyen iletiler kuyruğuna koyamazsa, ileti kaynak IBM MQ kuyruğuna döndürülür. Sistem konsoluna bir hata iletisi gönderilir ve kuyruktan başka ileti gönderilmez.

  İletileri yeniden göndermek için aşağıdakilerden ***birini*** gerçekleştirin:

  – Kuyruğun karşılığı olan IMS içindeki Tpipes öğesini durdurun ve yeniden başlatın.
  – Kuyruğu GET (DISABLED) olarak değiştirin ve yeniden GET (ENABLED) olarak değiştirin
  – IMS ya da OTMA ' yı durdurun ve yeniden başlatın
  – IBM MQ altsisteminizi durdurun ve yeniden başlatın
- İleti IMS tarafından bir ileti hatası dışında bir hata için reddedilirse, IBM MQ iletisi kaynak kuyruğa döndürülür, IBM MQ kuyruğu işlemeyi durdurur ve sistem konsoluna bir hata iletisi gönderilir.

Bir kural dışı durum raporu iletisi gerekiyorsa, köprü bunu, oluşturanın yetkisiyle yanıt kuyruğuna koyar. İleti kuyruğa konamazsa, rapor iletisi köprünün yetkisiyle teslim edilmeyen iletiler kuyruğuna yerleştirilir. DLQ ' ya konamazsa atılır.

z/OS *IMS köprü geribildirim kodları*

IMS algılama kodları genellikle CSQ2001I gibi IBM MQ konsol iletilerinde onaltılı biçimde çıkılır (örneğin, algılama kodu 0x001F). Gönderilmeyen iletiler kuyruğuna konan iletilerin girilmeyen harf üstbilgisinde görülen IBM MQ geribildirim kodları ondalık sayılardır.

IMS köprüsü geribildirim kodları 301-399 ya da NACK algılama kodu 0x001Aiçin 600-855 aralığındadır. Bunlar IMS-OTMA algılama kodlarından aşağıdaki gibi eşlenir:

1. IMS-OTMA algılama kodu onaltılı bir sayıdan ondalık sayıya dönüştürülür.
2. IBM MQ *Feedback* kodu verilerek, 1' deki hesaplamadan elde edilen sayıya 300 eklenir.
3. IMS-OTMA algılama kodu 0x001A, ondalık 26 özel bir durumdur. 600-855 aralığında bir *Geribildirim* kodu oluşturulur.
   a. IMS-OTMA neden kodu onaltılı sayıdan ondalık sayıya dönüştürülür.
   b. aiçindeki hesaplamadan elde edilen sayıya 600 eklenir ve IBM MQ *Geribildirim* kodu verir.

IMS-OTMA algılama kodları hakkında bilgi için bkz. NAK iletileri için OTMA algılama kodları.

z/OS *IMS köprüsündeki iletilerdeki MQMD alanları*

IMS köprüsünden iletilerdeki MQMD alanları hakkında bilgi edinin.

Kaynak iletinin MQMD ' si, OTMA üstbilgilerinin Kullanıcı Verileri kısmında IMS tarafından taşınır. İleti IMSiçinde oluşursa, bu, IMS Hedef Çözüm Çıkışı tarafından oluşturulur. IMS 'den alınan bir iletinin MQMD' si aşağıdaki gibi oluşturulmuştur:

StrucID
"MD"

Sürüm
MQMD_VERSION_1

Rapor
MQRO_NONE

MsgType
MQMT_REPLY

Son kullanma tarihi
MQIIH ' nin İşaretler alanında MQIIH_PASS_EXPIRATION ayarlandıysa, bu alan kalan süre bitimini içerir, aksi takdirde MQEI_UNLIMITED olarak ayarlanır

Geri bildirim
MQFB_NONE

Kodlama
MQENC.Native (z/OS sisteminin kodlaması)

CodedCharSetId
MQCCSI_Q_MGR (z/OS sisteminin CodedCharSetID)

Biçim
MQMD.Format biçimi MQFMT_IMS, tersi durumda IOPCB.MODNAME

Öncelik
MQMD.Priority

Kalıcılık
Kesinleştirme kipine bağlıdır: CM-1; kalıcılık, CM-0 ise IMS iletisinin kurtarılabilirliği ile eşleşiyorsa, giriş iletisinin MQMD.Persistence .

MsgId
MQMD.MsgId MQRO_PASS_MSG_ID ise, tersi durumda Yeni MsgId (varsayılan)

CorrelId
MQMD.CorrelId (MQRO_PASS_CORREL_ID ise), tersi durumda MQMD.MsgId (varsayılan)

BackoutCount
0

ReplyToQ
Boşluklar

ReplyToQMgr
Boşluklar (MQPUT sırasında kuyruk yöneticisi tarafından yerel qmgr adına ayarlayın)

UserIdentifier
MQMD.UserIdentifier

AccountingToken
MQMD.AccountingToken

ApplIdentityVerileri
MQMD.ApplIdentityData

PutApplTipi
Hata yoksa MQAT_XCF, yoksa MQAT_BRIDGE

PutApplAdı
<XCFgroupName> <XCFmemberName> hata yoksa, QMGR adı

PutDate
İletinin konma tarihi

PutTime
İletinin konma zamanı

ApplOriginVerileri
Boşluklar

z/OS *IMS köprüsündeki iletilerdeki MQIIH alanları*

IMS köprüsünden iletilerdeki MQIIH alanları hakkında bilgi edinin.

IMS ' den alınan bir iletinin MQIIH 'si aşağıdaki gibi oluşturulmuştur:

StrucId
"IIH"

Sürüm
1

StrucLength
84

Kodlama
MQENC_NATIVE

CodedCharSetId
MQCCSI_Q_MGR

Biçim
MQIIH.ReplyToFormat ya da MQIIH.ReplyToFormat boş değilse, IOPCB.MODNAME

İşaretler
0

LTermOverride
OTMA üstbilgisinden LTERM adı (Tpipe)

MFSMapName
OTMA üstbilgisinden eşlem adı

ReplyToBiçimi
Boşluklar

Kimlik doğrulayıcı
Yanıt iletisi bir MQ-IMS köprü kuyruğuna konuluyorsa, giriş iletisinin MQIIH.Authenticator , tersi durumda boşluklar.

TranInstanceTanıtıcısı
Etkileşimde ise, OTMA üstbilgisinden etkileşim tanıtıcısı/sunucu simgesi. IMS ' nin V14öncesi sürümlerinde, bu alan etkileşim içinde değilse her zaman boş olur. IMS V14 ' ten itibaren bu alan, etkileşim içinde olmasa da IMS tarafından ayarlanabilir.

TranState
Sohbette "C" varsa, tersi durumda boş

CommitMode
OTMA üstbilgisinden kesinleştirme kipi ("0" ya da "1")

SecurityScope
Boş

Ayrıldı
Boş

z/OS *IMS ' den yanıt iletileri*

Bir IMS hareketi ISR 'leri IOPCB 'sine yönlendirdiğinde, ileti kaynak LTERM ya da TPIPE' ye geri yönlendirilir.

Bunlar IBM MQ içinde yanıt iletileri olarak görülür. IMS ' den gelen yanıt iletileri, özgün iletide belirtilen yanıt kuyruğuna yerleştirilir. İleti, yanıt kuyruğuna konamazsa, köprünün yetkisi kullanılarak teslim edilmeyen iletiler kuyruğuna yerleştirilir. İleti, teslim edilmeyen iletiler kuyruğuna konamazsa, iletinin alınamadığını söylemek için IMS ' e negatif bir alındı bildirimi gönderilir. Daha sonra, iletinin sorumluluğu IMS' e döndürülür. Kesinleştirme kipi 0 'ı kullanıyorsanız, bu Tpipe ' dan gelen iletiler köprüye gönderilmez ve IMS kuyruğunda kalır; yani, yeniden başlatılıncaya kadar başka ileti gönderilmez. Kesinleştirme kipi 1 'i kullanıyorsanız, diğer işler devam edebilir.

Yanıtın MQIIH yapısı varsa, biçim tipi MQFMT_IMS; değilse, biçim tipi iletiyi eklerken kullanılan IMS MOD adıyla belirtilir.

z/OS *IMS hareketlerinde alternatif yanıt PCB ' lerinin kullanılması*

Bir IMS hareketi alternatif yanıt PCB 'leri (ALTPCB' ye ilişkin IST 'ler) kullandığında ya da değiştirilebilir bir PCB' ye CHNG çağrısı verdiğinde, iletinin yeniden yönlendirilip yönlendirilmeyeceğini belirlemek için ön yöneltme çıkışı (DFSYPRX0) çağrılır.

İleti yeniden yönlendirilirse, hedefi onaylamak ve üstbilgi bilgilerini hazırlamak için hedef çözüm çıkışı (DFSYDRU0) çağrılır. Bu çıkış programlarıyla ilgili bilgi için bkz. IMS içinde OTMA çıkışlarının kullanılması ve Ön yönlendirme çıkışı DFSYPRX0 .

Çıkışlarda işlem yapılmazsa, IOPCB 'ye ya da ALTPCB' ye IBM MQ kuyruk yöneticisinden başlatılan tüm IMS hareketleri aynı kuyruk yöneticisine döndürülür.

z/OS *IMS ' dan istenmeyen iletiler gönderme*

IMS 'den IBM MQ kuyruğuna ileti göndermek için, IST' lerin ALTPCB ' ye gönderdiği bir IMS işlemini başlatmanız gerekir.

IMS 'den istenmeyen iletileri yönlendirmek ve OTMA kullanıcı verilerini oluşturmak için ön yöneltme ve hedef çözüm çıkışlarını yazmanız gerekir; böylece, iletinin MQMD' si doğru bir şekilde oluşturulabilir. Bu çıkış programlarına ilişkin bilgi için Ön yöneltme çıkışı DFSYPRX0 ve Hedef çözüm kullanıcı çıkışı başlıklı konuya bakın.

Not: IBM MQ - IMS köprüsü, aldığı bir iletinin yanıt mı, yoksa istenmeyen bir ileti mi olduğunu bilmiyor. İletiyle birlikte gelen OTMA UserData 'ya dayalı olarak yanıtın MQMD ve MQIIH' yi oluşturarak, her durumda iletiyi aynı şekilde işler.

İstenmeyen iletiler yeni Tpipes yaratabilir. Örneğin, var olan bir IMS hareketi yeni bir LTERM ' ye (örneğin, PRINT01) geçtiyse, ancak uygulama çıkışın OTMA aracılığıyla teslim edilmesini gerektiriyorsa, yeni bir Tpipe (bu örnekte PRINT01 olarak adlandırılır) yaratılır. Varsayılan olarak bu, eşitlenmemiş bir Tpipe 'dir.

Uygulama iletinin kurtarılabilir olmasını gerektiriyorsa, hedef çözüm çıkış işaretini ayarlayın. Daha fazla bilgi için bkz. *IMS Özelleştirme Kılavuzu* .

z/OS *İleti bölümlemesi*

IMS hareketlerini tek ya da çok bölümlü giriş beklenirken tanımlayabilirsiniz.

Kaynak IBM MQ uygulaması, MQIIH yapısını izleyen kullanıcı girişini bir ya da daha çok LLZZ veri bölümü olarak oluşturmalıdır. IMS iletisinin tüm bölümleri, tek bir MQPUT ile gönderilen tek bir IBM MQ iletisinde bulunmalıdır.

Bir LLZZ veri kesiminin uzunluk üst sınırı IMS/OTMA (32767 bayt) tarafından tanımlanır. Toplam IBM MQ ileti uzunluğu, LL baytlarının toplamı ve MQIIH yapısının uzunluğudur.

Yanıtın tüm bölümleri tek bir IBM MQ iletisinde bulunur.

MQFMT_IMS_VAR_STRING biçimindeki iletilere ilişkin 32 KB sınırlaması üzerinde ek bir kısıtlama vardır. ASCII karışık CCSID iletisindeki veriler EBCDIC karışık CCSID iletisine dönüştürüldüğünde, SBCS ve DBCS karakterleri arasında her geçişte bir çift bayt ya da çift bayt dizilimi başlangıç bayt dizilimi başlangıç ve bitiş karakterleri eklenir. 32 KB sınırlaması, iletinin büyüklük üst sınırı için geçerlidir. Yani, iletideki LL alanı 32 KB 'yi aşamayacağı için, iletinin tüm çift bayt dizilimi başlangıç ve bitiş karakterleri de içinde olmak üzere 32 KB' yi aşmaması gerekir. İletinin oluşturulmasına izin veren uygulama buna izin vermelidir.

z/OS *IMS köprüsüne/köprüsünden gelen iletiler için veri dönüştürme*

Veri dönüştürme, bir iletiyi depolama sınıfı için tanımlanmış XCF bilgilerini içeren bir hedef kuyruğa yerleştirdiğinde, dağıtılmış kuyruğa alma olanağı (gerekli çıkışları çağırabilecek) ya da grup içi kuyruğa alma aracısı (çıkış kullanımını desteklemeyen) tarafından gerçekleştirilir. Bir ileti yayınlama/abone olma yoluyla kuyruğa teslim edildiğinde veri dönüştürme gerçekleşmez.

Gereken çıkışlar, CSQXLIB DD deyimiyle gönderme yapılan veri kümesindeki dağıtılmış kuyruğa alma olanağı için kullanılabilir olmalıdır. Bu, herhangi bir IBM MQ platformundan IBM MQ - IMS köprüsünü kullanarak bir IMS uygulamasına ileti gönderebileceğiniz anlamına gelir.

Dönüştürme hataları varsa, ileti dönüştürülmeden kuyruğa yerleştirilir; bu, sonunda köprünün üstbilgi biçimini tanıyamamasından dolayı IBM MQ - IMS köprüsü tarafından hata olarak kabul edilmesiyle sonuçlanır. Bir dönüştürme hatası ortaya çıkarsa, z/OS konsoluna bir hata iletisi gönderilir.

Genel olarak veri dönüştürmeye ilişkin ayrıntılı bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941 .

IBM MQ - IMS köprüsüne ileti gönderilmesi

Dönüştürmenin doğru gerçekleştirildiğinden emin olmak için, kuyruk yöneticisine iletinin biçiminin ne olduğunu söylemeniz gerekir.

İletinin MQIIH yapısı varsa, MQMD 'deki *Format* yerleşik biçimde MQFMT_IMS olarak ayarlanmalı ve MQIIH' deki *Format* , ileti verilerinizi tanımlayan biçimin adına ayarlanmalıdır. MQIIH yoksa, MQMD ' deki *Format* değerini biçim adınıza ayarlayın.

Verileriniz (LLZZ ' ler dışında) tüm karakter verileriyse (MQCHAR), yerleşik MQFMT_IMS_VAR_STRING biçimini biçim adınız (MQIIH ya da MQMD içinde) olarak kullanın. Tersi durumda, kendi biçim adınızı kullanın; bu durumda, biçiminiz için bir veri dönüştürme çıkışı da sağlamanız gerekir. Çıkış, verinin kendisine ek olarak, iletinizdeki LLZZ 'lerin dönüştürülmesini de işlemelidir (ancak iletinin başında herhangi bir MQIIH' yi işlemesi gerekmez).

Uygulamanız *MFSMapName*kullanıyorsa, bunun yerine MQFMT_IMS ile iletileri kullanabilir ve MQIIH ' nin MFSMapName alanında IMS hareketine geçirilen eşlem adını tanımlayabilirsiniz.

IBM MQ - IMS köprüsünden ileti alınması

IMS' e göndermekte olduğunuz özgün iletide bir MQIIH yapısı varsa, yanıt iletisinde de bir MQIIH yapısı vardır.

Yanıtınız doğru olarak dönüştürüldüğünden emin olmak için:

- Özgün iletinizde MQIIH yapısı varsa, özgün iletinin MQIIH *ReplytoFormat* alanında yanıt iletiniz için istediğiniz biçimi belirtin. Bu değer, yanıt iletisinin MQIIH *Format* alanına yerleştirilir. Bu özellikle, tüm çıkış verileriniz LLZZ < karakter verileri > biçimindeyse kullanışlıdır.
- Özgün iletinizde MQIIH yapısı yoksa, yanıt iletisi için istediğiniz biçimi IMS uygulamasının IOPCB 'ye ilişkin ISRT' de MFS MOD adı olarak belirtin.

JMS/Jakarta Messaging ve Java uygulamalarının geliştirilmesi

IBM MQ üç Java dil arabirimi sağlar: IBM MQ classes for Jakarta Messaging, IBM MQ classes for JMSve IBM MQ classes for Java.

Bu görev hakkında

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

IBM MQ classes for Jakarta Messaging , ileti sistemi olarak IBM MQ için Jakarta Messaging arabirimlerini uygulayan bir Jakarta Messaging sağlayıcısıdır. Jakarta Connectors Architecture , Jakarta EE ortamında çalışan uygulamaları IBM MQ ya da Db2gibi bir Kurumsal Bilgi Sistemine (EIS) bağlamanın standart bir yolunu sağlar.

Daha fazla bilgi için, bkz. “Neden IBM MQ classes for Jakarta Messagingkullanmalıyım?” sayfa 79 ve “IBM MQ 'e Java ' dan erişme-API Seçimi” sayfa 81.

JMS 2.0 **IBM MQ classes for JMS**

IBM MQ classes for JMS , ileti sistemi olarak IBM MQ için JMS arabirimlerini uygulayan bir JMS sağlayıcısıdır. Java Platform, Enterprise Edition Connector Architecture (JCA), Java EE ortamında çalışan uygulamaları IBM MQ ya da Db2gibi bir Kurumsal Bilgi Sistemine (EIS) bağlamanın standart bir yolunu sağlar.

Daha fazla bilgi için, bkz. “Neden IBM MQ classes for JMSkullanmalıyım?” sayfa 80 ve “IBM MQ 'e Java ' dan erişme-API Seçimi” sayfa 81.

IBM MQ classes for Java

IBM MQ classes for Java , Java ortamında IBM MQ kullanmanızı sağlar. IBM MQ classes for Java , bir Java uygulamasının IBM MQ istemcisi olarak IBM MQ ' e bağlanmasına ya da IBM MQ kuyruk yöneticisine doğrudan bağlanmasına izin verir.

IBM MQ classes for Java , İleti Kuyruğu Arabirimi 'ni (MQI), yerel IBM MQ API 'sini kapsüller ve diğer nesne yönelimli arabirimlerle aynı nesne modelini kullanırken, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging sırasıyla Oracle ve Java Community Process ' dan Java ileti alışverişi arabirimlerini gerçekleştirir.

Daha fazla bilgi için, bkz. “Neden IBM MQ classes for Javakullanmalıyım?” sayfa 335 ve “IBM MQ 'e Java ' dan erişme-API Seçimi” sayfa 81.

Not:

Stabilized IBM , IBM MQ classes for Java üzerinde başka geliştirme yapmayacaktır ve bunlar IBM MQ 8.0içinde gönderilen düzeyde işlevsel olarak sabitlenecektir. IBM MQ classes for Java kullanan var olan uygulamalar tam olarak desteklenmeye devam eder, ancak yeni özellikler eklenmez ve geliştirme istekleri reddedilir. Tam olarak desteklenen, IBM MQ Sistem Gereksinimlerinde yapılan değişikliklerle birlikte hataların düzeltileceği anlamına gelir.

IBM MQ classes for Java , IMSiçinde desteklenmez.

IBM MQ classes for Java , WebSphere Libertyiçinde desteklenmez. Bunlar IBM MQ Liberty ileti sistemi özelliğiyle ya da genel JCA desteğiyle birlikte kullanılmamalıdır. Daha fazla bilgi için bkz. WebSphere MQ Java Arabirimlerinin J2EE/JEE Ortamlarında Kullanılması.

IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , IBM MQile verilen Java ileti alışverişi sağlayıcılarıdır. JMS ve Jakarta Messaging belirtimlerinde tanımlanan arabirimleri gerçekleştirmenin yanı sıra, bu ileti alışverişi sağlayıcıları Java ileti sistemi API 'sine iki uzantı kümesi ekler.

V 9.3.0 JM 3.0 V 9.3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Aynı uygulamada hem JMS 2.0 API hem de Jakarta Messaging 3.0 API ' nin kullanılması desteklenmez.

Not: Jakarta Messaging 3.0için JMS belirtimi denetimi, Oracle öğesinden Java Community Processöğesine taşınır. Ancak Oracle , Java Topluluk Sürecine taşınmayan diğer Java teknolojilerinde kullanılan "javax" adının denetimini elinde tutar. Bu nedenle Jakarta Messaging 3.0 , JMS 2.0 ile işlevsel olarak eşdeğerdir, ancak adlandırma konusunda bazı farklılıklar vardır:

- 3.0 sürümünün resmi adı Java Message Serviceyerine Jakarta Messaging .
- Paket ve sabit adlara `javax`yerine `jakarta` öneki eklenir. Örneğin, JMS 2.0 içinde bir ileti alışverişi sağlayıcısına ilk bağlantı bir `javax.jms.Connection` nesnesidir ve Jakarta Messaging 3.0 içinde bir `jakarta.jms.Connection` nesnesidir.

JMS 2.0 javax.jms paketleri JMS arabirimlerini tanımlar ve bir JMS sağlayıcısı bu arabirimleri belirli bir ileti sistemi ürünü için uygular. IBM MQ classes for JMS , IBM MQiçin JMS arabirimlerini uygulayan bir JMS sağlayıcısıdır.

JM 3.0 jakarta.jms paketleri Jakarta Messaging arabirimlerini tanımlar ve bir Jakarta Messaging sağlayıcısı bu arabirimleri belirli bir ileti sistemi ürünü için gerçekleştirir. IBM MQ classes for Jakarta Messaging , IBM MQiçin Jakarta Messaging arabirimlerini uygulayan bir Jakarta Messaging sağlayıcısıdır.

JMS ve Jakarta Messaging belirtimleri, ConnectionFactory ve Destination nesnelerinin denetlenmesini bekler. Denetimci, yönetilen nesneleri merkezi bir havuzda yaratır ve saklar; bir JMS ya da Jakarta Messaging uygulaması, Java Naming Directory Interface (JNDI) kullanarak bu nesneleri alır.

JMS 2.0 JMS 2.0için bir yönetici, merkezi bir havuzda yönetilen nesneler oluşturmak ve bunların bakımını yapmak için IBM MQ JMS yönetim aracını **JMSAdmin**ya da IBM MQ Explorerkullanabilir.

JM 3.0 Jakarta Messaging 3.0için, IBM MQ Explorerkullanarak JNDI ' yi yönetemezsiniz. JNDI yönetimi, **JMSAdmin**' ın **JMS30Admin**olan Jakarta Messaging 3.0 çeşitlemesi tarafından desteklenir.

JMS ve Jakarta Messaging ortak çok şey paylaştığı için, bu konuda JMS ile ilgili daha fazla başvuru, her ikisine de atıfta bulunma olarak alınabilir. Tüm farklar gerektiği şekilde vurgulanır.

IBM MQ classes for JMS , JMS API 'si için iki uzantı kümesi de sağlar. Bu uzantıların ana odağı, çalıştırma zamanında bağlantı fabrikalarının ve hedeflerinin dinamik olarak oluşturulmasını ve yapılandırılmasını ilgilendirir, ancak uzantılar, sorun belirleme işlevi gibi ileti sistemiyle doğrudan ilişkili olmayan işlevler de sağlar.

IBM MQ JMS uzantıları
IBM MQ classes for JMS , MQConnectionFactory, MQQueue ve MQTopic gibi nesnelerde uygulanan uzantıları içerir. Bu nesnelerin IBM MQ' e özgü özellikleri ve yöntemleri vardır. Nesneler denetlenen nesneler olabilir ya da bir uygulama nesneleri yürütme sırasında dinamik olarak yaratabilir. Bu uzantılar, IBM MQ JMS uzantıları olarak adlandırılır.

IBM JMS uzantıları
IBM MQ classes for JMS , ileti sistemi olarak IBM MQ 'e ya da kullanılan programlama dili olarak Java ' e özgü olmayan JMS API için daha genel bir uzantı kümesi de sağlar. Bu uzantılar, IBM JMS uzantıları olarak adlandırılır ve aşağıdaki geniş hedeflere sahip olur:

- IBM JMS sağlayıcıları arasında daha yüksek bir tutarlılık düzeyi sağlamak için.
- İki IBM ileti sistemi arasında bir köprü uygulaması yazmayı kolaylaştırmak için.
- Bir uygulamayı bir IBM JMS sağlayıcısından diğerine bağlamayı kolaylaştırmak için.

Uzantılar, IBM MQ Message Service Client (XMS) for C/C++ ve IBM MQ Message Service Client (XMS) for .NETiçinde sağlanana benzer bir işlev sağlar.

İlgili kavramlar

IBM MQ Java dil arabirimleri

İlgili görevler

“IBM MQ classes for JMS/Jakarta Messaging uygulamaları yazılıyor” sayfa 134

JMS modeline kısa bir giriş yaptıktan sonra, bu bölümde IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging uygulamalarının nasıl yazılacağına ilişkin ayrıntılı yönergeler yer alır.

V 9.3.0 V 9.3.0 JM 3.0 Neden IBM MQ classes for Jakarta Messagingkullanmalıyım?

IBM MQ classes for Jakarta Messaging ' in kullanılması, kuruluşunuzda var olan Jakarta Messaging becerilerinin yeniden kullanılabilmesi ve uygulamaların Jakarta Messaging sağlayıcısından ve temel IBM MQ yapılandırmasından daha bağımsız olması gibi çeşitli avantajlara sahiptir.

IBM MQ classes for Jakarta Messaging kullanımına ilişkin avantajların özeti

IBM MQ classes for Jakarta Messaging komutunu kullanmanız, var olan Jakarta Messaging becerilerini yeniden kullanmanıza ve uygulama bağımsızlığı sağlamanıza olanak sağlar.

- Jakarta Messaging becerilerini yeniden kullanabilirsiniz.

  IBM MQ classes for Jakarta Messaging , ileti sistemi olarak IBM MQ için Jakarta Messaging arabirimlerini uygulayan bir Jakarta Messaging sağlayıcısıdır. Kuruluşunuz IBM MQuygulamasında yeniyse, ancak Jakarta Messaging (ya da JMS) uygulama geliştirme becerilerine sahipse, IBM MQile sağlanan diğer API 'lerden biri yerine IBM MQ kaynaklarına erişmek için bilinen Jakarta Messaging API' yi kullanmayı daha kolay bulabilirsiniz.
- Jakarta Messaging , Jakarta EE' in ayrılmaz bir parçasıdır.

  Jakarta Messaging , Jakarta EE platformunda ileti alışverişi için kullanılacak doğal API 'dir. Jakarta EE ile uyumlu her uygulama sunucusu bir Jakarta Messaging sağlayıcısı içermelidir. You can use Jakarta Messaging in application clients, servlets, Java Server Pages (JSPs), enterprise Java beans (EJBs), and message driven beans (MDBs). Özellikle Jakarta EE uygulamalarının iletileri zamanuyumsuz olarak işlemek için MDB 'leri kullandığını ve tüm iletilerin MDB' lere Jakarta Messaging iletileri olarak teslim edildiğini unutmayın.
- Bağlantı üreticileri ve hedefleri, bir uygulamaya sabit olarak kodlanmaktansa, merkezi bir havuzda Jakarta Messaging tarafından yönetilen nesneler olarak saklanabilir.

  Denetimci, merkezi bir havuzda Jakarta Messaging tarafından denetlenen nesneler yaratabilir ve bunların bakımını yapabilir; IBM MQ classes for Jakarta Messaging uygulamaları Java Naming Directory Interface (JNDI) kullanarak bu nesneleri alabilir. Jakarta Messaging bağlantı üreticileri ve hedefleri, IBM MQkuyruk yöneticisi adları, kanal adları, bağlantı seçenekleri, kuyruk adları ve konu adları gibi belirli bilgileri içerir. Bağlantı üreticileri ve hedefleri yönetilen nesneler olarak saklandıysa, bu bilgiler bir uygulamaya sabit olarak kodlanmaz. Bu nedenle bu düzenleme, uygulamaya temel IBM MQ yapılandırmasından bir derece bağımsızlık sağlar.
- Jakarta Messaging , uygulama taşınabilirliği sağlayabilen sektör standardında bir API 'dir.

  Bir Jakarta Messaging uygulaması, yönetilen nesneler olarak saklanan bağlantı üreticilerini ve hedefleri almak için JNDI öz etme özelliğini kullanabilir ve ileti alışverişi işlemlerini gerçekleştirmek için yalnızca `jakarta.jms` (Jakarta Messaging 3.0) paketinde tanımlanan arabirimleri kullanabilir. Daha sonra uygulama, IBM MQ classes for Jakarta Messaginggibi herhangi bir Jakarta Messaging sağlayıcısından tamamen bağımsızdır ve uygulamada herhangi bir değişiklik yapılmaksızın bir Jakarta Messaging sağlayıcısından diğerine taşınabilir.

  JNDI belirli bir uygulama ortamında yoksa, bir IBM MQ classes for Jakarta Messaging uygulaması yürütme sırasında bağlantı üreticilerini ve hedefleri dinamik olarak oluşturmak ve yapılandırmak için

Jakarta Messaging API ' nın uzantılarını kullanabilir. Daha sonra uygulama tamamen bağımsız olur, ancak Jakarta Messaging sağlayıcısı olarak IBM MQ classes for Jakarta Messaging ' e bağlanır.

- Köprü uygulamalarının Jakarta Messagingkullanılarak yazılması daha kolay olabilir.

  Köprü uygulaması, bir ileti sistemi sisteminden ileti alan ve bunları başka bir ileti sistemi sistemine gönderen bir uygulamadır. Bir köprü uygulaması yazmak, ürüne özgü API ' ler ve ileti biçimleri kullanılarak karmaşık olabilir. Bunun yerine, her ileti sistemi için bir tane olmak üzere iki Jakarta Messaging sağlayıcısı kullanarak bir köprü uygulaması yazabilirsiniz. Daha sonra uygulama yalnızca bir API, Jakarta Messaging API kullanır ve yalnızca Jakarta Messaging iletilerini işler.

Konuşlandırılabilir ortamlar

Bir Jakarta EE uygulama sunucusuyla bütünleştirme sağlamak için Jakarta EE standartları, ileti sistemi sağlayıcılarının bir kaynak bağdaştırıcısı sağlamasını gerektirir. Jakarta Connectors Architecture belirtimini izleyerek IBM MQ , herhangi bir sertifikalı Jakarta EE ortamında ileti sistemi işlevleri sağlamak için Jakarta Messaging ' i kullanan bir kaynak bağdaştırıcısı sağlar. Daha fazla bilgi için bkz “Liberty ve IBM MQ kaynak bağdaştırıcısı” sayfa 425.

Not: WebSphere Application Server traditional şu anda Jakarta EE' yi desteklemez.

Jakarta EE ortamı dışında OSGi ve JAR dosyaları sağlanır ve bu, yalnızca IBM MQ classes for Jakarta Messagingürününü edinmeniz için daha kolay olur. Bu JAR dosyaları, Maven gibi yazılım yönetimi çerçeveleri içinde ya da bağımsız olarak konuşlandırılabilir. Daha fazla bilgi için bkz. “IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging ürünlerini ayrı olarak edinme” sayfa 121.

İlgili kavramlar

IBM MQ sınıfları: genel bakış

“IBM MQ 'e Java ' dan erişme-API Seçimi” sayfa 81

IBM MQ , üç Java dil arabirimi sağlar.

JMS 2.0 Neden IBM MQ classes for JMSkullanmalıyım?

IBM MQ classes for JMS ' in kullanılması, kuruluşunuzda var olan JMS becerilerinin yeniden kullanılabilmesi ve uygulamaların JMS sağlayıcısından ve temel IBM MQ yapılandırmasından daha bağımsız olması gibi çeşitli avantajlara sahiptir.

IBM MQ classes for JMS kullanımına ilişkin avantajların özeti

IBM MQ classes for JMS komutunu kullanmanız, var olan JMS becerilerini yeniden kullanmanıza ve uygulama bağımsızlığı sağlamanıza olanak sağlar.

V 9.3.0 V 9.3.0

Not: JMS 2.0 yerine Jakarta Messaginggeçti. IBM MQ classes for JMS , JMS 2.0 standardını desteklemeye devam eder, ancak Java ileti sisteminde gelecekteki geliştirmeler yalnızca Jakarta Messagingiçinde, dolayısıyla IBM MQ classes for Jakarta Messagingiçinde ortaya çıkar. IBM MQ classes for JMS yalnızca var olan JMS 2.0 uygulamalarının bakımı ve genişletilmesi için önerilir. IBM MQ classes for Jakarta Messaging , yeni geliştirme için tercih edilen teknoloji olmalıdır.

- JMS becerilerini yeniden kullanabilirsiniz.

  IBM MQ classes for JMS , ileti sistemi olarak IBM MQ için JMS arabirimlerini uygulayan bir JMS sağlayıcısıdır. Kuruluşunuz IBM MQuygulamasında yeniyse, ancak zaten JMS uygulama geliştirme becerilerine sahipse, IBM MQile birlikte sağlanan diğer API ' lerden biri yerine IBM MQ kaynaklarına erişmek için bilinen JMS API 'sini kullanmayı daha kolay bulabilirsiniz.

- JMS , Java Platform, Enterprise Edition ' un (Java EE) ayrılmaz bir parçasıdır.

  JMS , Java EE platformunda ileti alışverişi için kullanılacak doğal API 'dir. Java EE ile uyumlu her uygulama sunucusu bir JMS sağlayıcısı içermelidir. You can use JMS in application clients, servlets, Java Server Pages (JSPs), enterprise Java beans (EJBs), and message driven beans (MDBs). Özellikle Java EE

uygulamalarının iletileri zamanuyumsuz olarak işlemek için MDB 'leri kullandığını ve tüm iletilerin MDB' lere JMS iletileri olarak teslim edildiğini unutmayın.

- Bağlantı üreticileri ve hedefleri, bir uygulamaya sabit olarak kodlanmaktansa, merkezi bir havuzda JMS tarafından yönetilen nesneler olarak saklanabilir.

  Denetimci, merkezi bir havuzda JMS tarafından denetlenen nesneler yaratabilir ve bunların bakımını yapabilir; IBM MQ classes for JMS uygulamaları Java Naming Directory Interface (JNDI) kullanarak bu nesneleri alabilir. JMS bağlantı üreticileri ve hedefleri, IBM MQkuyruk yöneticisi adları, kanal adları, bağlantı seçenekleri, kuyruk adları ve konu adları gibi belirli bilgileri içerir. Bağlantı üreticileri ve hedefleri yönetilen nesneler olarak saklandıysa, bu bilgiler bir uygulamaya sabit olarak kodlanmaz. Bu nedenle bu düzenleme, uygulamaya temel IBM MQ yapılandırmasından bir derece bağımsızlık sağlar.

- JMS , uygulama taşınabilirliği sağlayabilen sektör standardında bir API 'dir.

  Bir JMS uygulaması, yönetilen nesneler olarak saklanan bağlantı üreticilerini ve hedefleri almak için JNDI öz etme özelliğini kullanabilir ve ileti alışverişi işlemlerini gerçekleştirmek için yalnızca `javax.jms` paketinde tanımlanan arabirimleri kullanabilir. Daha sonra uygulama, IBM MQ classes for JMSgibi herhangi bir JMS sağlayıcısından tamamen bağımsızdır ve uygulamada herhangi bir değişiklik yapılmaksızın bir JMS sağlayıcısından diğerine taşınabilir.

  JNDI belirli bir uygulama ortamında yoksa, bir IBM MQ classes for JMS uygulaması, çalıştırma zamanında bağlantı üreticilerini ve hedefleri dinamik olarak oluşturmak ve yapılandırmak için JMS API ' nın uzantılarını kullanabilir. Daha sonra uygulama tamamen bağımsız olur, ancak JMS sağlayıcısı olarak IBM MQ classes for JMS ' e bağlanır.

- Köprü uygulamalarının JMSkullanılarak yazılması daha kolay olabilir.

  Köprü uygulaması, bir ileti sistemi sisteminden ileti alan ve bunları başka bir ileti sistemi sistemine gönderen bir uygulamadır. Bir köprü uygulaması yazmak, ürüne özgü API ' ler ve ileti biçimleri kullanılarak karmaşık olabilir. Bunun yerine, her ileti sistemi için bir tane olmak üzere iki JMS sağlayıcısı kullanarak bir köprü uygulaması yazabilirsiniz. Daha sonra uygulama yalnızca bir API, JMS API kullanır ve yalnızca JMS iletilerini işler.

Konuşlandırılabilir ortamlar

Bir Java EE uygulama sunucusuyla bütünleştirme sağlamak için Java EE standartları, ileti sistemi sağlayıcılarının bir kaynak bağdaştırıcısı sağlamasını gerektirir. Java EE Connector Architecture (JCA) belirtimini izleyerek IBM MQ , onaylı herhangi bir Java EE ortamında ileti alışverişi işlevleri sağlamak için JMS komutunu kullanan bir kaynak bağdaştırıcısı sağlar.

IBM MQ classes for Java ' in Java EEiçinde kullanılması mümkün olsa da, bu API bu amaçla tasarlanmaz ya da eniyilenmez. Java EEiçindeki IBM MQ classes for Java ile ilgili daha fazla bilgi için bkz. “Java EE içinde IBM MQ classes for Java uygulamalarının çalıştırılması” sayfa 336.

Java EE ortamı dışında OSGi ve JAR dosyaları sağlanır ve bu, yalnızca IBM MQ classes for JMSürününü edinmeniz için daha kolay olur. Bu JAR dosyaları, Maven gibi yazılım yönetimi çerçeveleri içinde ya da bağımsız olarak konuşlandırılabilir. Daha fazla bilgi için bkz. “IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging ürünlerini ayrı olarak edinme” sayfa 121.

İlgili kavramlar

V 9.3.0 V 9.3.0 IBM MQ sınıfları: genel bakış

“Neden IBM MQ classes for Jakarta Messagingkullanmalıyım?” sayfa 79
IBM MQ classes for Jakarta Messaging ' in kullanılması, kuruluşunuzda var olan Jakarta Messaging becerilerinin yeniden kullanılabilmesi ve uygulamaların Jakarta Messaging sağlayıcısından ve temel IBM MQ yapılandırmasından daha bağımsız olması gibi çeşitli avantajlara sahiptir.

“IBM MQ 'e Java ' dan erişme-API Seçimi” sayfa 81
IBM MQ , üç Java dil arabirimi sağlar.

IBM MQ 'e Java ' dan erişme-API Seçimi

IBM MQ , üç Java dil arabirimi sağlar.

- V 9.3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging
- IBM MQ classes for JMS
- IBM MQ classes for Java

IBM MQ classes for Jakarta Messaging

V 9.3.0 V 9.3.0 IBM MQ classes for Jakarta Messaging , Jakarta Messaging 3.0 API 'lerini kullanarak yazılan uygulamaların IBM MQ ' yi ileti alışverişi sağlayıcısı olarak kullanmasına olanak sağlar.

Jakarta Messaging , Java uygulamalarındaki ileti sistemi için stratejik yönelimdir.

Jakarta Messaging 3.0 , işlevsel olarak JMS 2.0ile eşdeğerdir, bu nedenle daha fazla bilgi için bkz. “IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma” sayfa 78.

IBM MQ classes for JMS

IBM MQ classes for JMS , JMS 2.0 API 'lerini kullanarak yazılan uygulamaların IBM MQ ' yi ileti alışverişi sağlayıcısı olarak kullanmasına olanak sağlar.

Jakarta Messaging JMS'in yerini aldıkça, IBM MQ classes for JMS ' in var olan uygulamalarda ya da Jakarta Messaging' yi desteklemeyen ortamlarda (örneğin, WebSphere Application Server) kullanılması önerilir.

Aynı uygulamada hem IBM MQ classes for Jakarta Messaging , hem de IBM MQ classes for JMS kullanılması desteklenmez.

Daha fazla bilgi için bkz. “IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma” sayfa 78.

IBM MQ classes for Java

Stabilized Java uygulamalarının IBM MQ kaynaklarına erişmek için kullanabileceği diğer API, IBM MQ ' yi ileti alışverişi sağlayıcısı olarak kullanmak üzere programlar için IBM MQodaklı bir API sağlayan IBM MQ classes for JavaAPI 'dir. Ancak IBM MQ classes for Java , IBM MQ 8.0içinde gönderilen düzeyde işlevsel olarak sabitlenir. Daha fazla bilgi için bkz “Neden IBM MQ classes for Javakullanmalıyım?” sayfa 335. IBM MQ classes for Java kullanan var olan uygulamalar tam olarak desteklenmeye devam etse de, yeni uygulamalar IBM MQ classes for Jakarta Messagingkullanmalıdır.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging ortak özellikleri

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , IBM MQ' in hem noktadan noktaya iletişim hem de yayınlama/abone olma ileti sistemi özelliklerine erişim sağlar. JMS standart ileti sistemi modeli için destek sağlayan JMS iletileri gönderirken, uygulamalar ek üstbilgiler olmadan da ileti gönderebilir ve alabilir; böylece, C MQI uygulamaları gibi diğer IBM MQ uygulamalarıyla birlikte çalışabilir. MQMD ve MQ ileti bilgi yüklerinin tam denetimi kullanılabilir.

İleti akışı, zamanuyumsuz koyma ve rapor iletileri gibi diğer IBM MQ özellikleri de mevcuttur.

Sağlanan PCF yardımcı sınıfları kullanılarak, IBM MQ PCF denetim iletileri JMS API aracılığıyla gönderilebilir ve alınabilir ve kuyruk yöneticilerini denetlemek için kullanılabilir.

IBM MQ' e zamanuyumsuz tüketim ve otomatik yeniden bağlanma gibi yakın zamanda eklenen özellikler IBM MQ classes for Javaiçinde bulunmaz, ancak IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingiçinde kullanılabilir.

Geliştirmeler isteniyor

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingaracılığıyla kullanılamayan IBM MQ özelliklerine erişmeniz gerekirse, bir fikir edinebilirsiniz.

IBM , daha sonra IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging uygulamasında uygulamanın mümkün olup olmadığını ya da izlenebilecek en iyi uygulama olup olmadığını bilebilir.

IBM açık standarda katkıda bulunan bir özellik olduğundan, ek ileti sistemi özellikleri için bu özellikler JCP sürecinin bir parçası olarak yükseltilebilir. Bunlar yalnızca Jakarta İleti Sistemi için geçerlidir.

İlgili bilgiler

IBM Ideas Portal 'a Hoş Geldiniz

JMS Java Belirtim İnceleme Süreci

PCF iletileri göndermek için JMS ' nin kullanılması

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ classes for Jakarta Messaging için önkoşullar

Bu konu, IBM MQ classes for Jakarta Messaging' yi kullanmadan önce bilmeniz gerekenleri size bildirir. IBM MQ classes for Jakarta Messaging uygulamalarını geliştirmek ve çalıştırmak için önkoşul olarak belirli yazılım bileşenlerine gereksinim duyarsınız.

IBM MQ classes for Jakarta Messagingile ilgili önkoşullar hakkında bilgi için bkz. IBM MQ için Sistem Gereksinimleri.

IBM MQ classes for Jakarta Messaging uygulamalarını geliştirmek için bir Java SE Software Development Kit (SDK) gerekir. İşletim sisteminiz tarafından desteklenen JDK ' lere ilişkin ayrıntılar için bkz. IBM MQ için Sistem Gereksinimleri.

IBM MQ classes for Jakarta Messaging uygulamalarını çalıştırmak için aşağıdaki yazılım bileşenlerine gereksinim duyarsınız:

- Bir IBM MQ kuyruk yöneticisi.
- Uygulamaları çalıştırdığınız her sistem için bir Java runtime environment (JRE).
- IBM i İşletim sisteminin 30. seçeneği olan IBM iQshell için.
- z/OS z/OSiçin, z/OS UNIX System Services (z/OS UNIX).

IBM JSSE sağlayıcısı, FIPS onaylı bir şifreleme sağlayıcısı içerir; bu nedenle, anında kullanıma hazır FIPS 140-2 uyumluluğu için programlı olarak yapılandırılabilir. Bu nedenle, FIPS 140-2 uyumluluğu doğrudan IBM MQ classes for Jakarta Messagingtarafından desteklenebilir.

OracleJSSE sağlayıcısının içinde yapılandırılmış bir FIPS sertifikalı şifreleme sağlayıcısı olabilir, ancak bu hemen kullanıma hazır değildir ve programlı yapılandırma için kullanılamaz. Bu nedenle IBM MQ classes for Jakarta Messaging , FIPS 140-2 uyumluluğunu doğrudan etkinleştiremez. Bu tür bir uyumluluğu el ile etkinleştirebilirsiniz, ancak IBM şu anda bu konuda rehberlik sağlayamaz.

IPv6 adresleri Java sanal makineniz (JVM) ve işletim sisteminizdeki TCP/IP uygulaması tarafından destekleniyorsa, IBM MQ classes for Jakarta Messaging uygulamalarınızda Internet Protocol sürüm 6 (IPv6) adreslerini kullanabilirsiniz. IBM MQ Jakarta Messaging yönetim aracı **JMS30Admin**, IPv6 adreslerini de kabul eder. Bu araçla ilgili daha fazla bilgi için Yönetim araçlarını kullanarak JMS ve Jakarta Messaging nesnelerini yapılandırmabaşlıklı konuya bakın.

IBM MQ JMS yönetim aracı ve IBM MQ Explorer , yönetilen nesneleri depolayan bir dizin hizmetine erişmek için Java Naming Directory Interface (JNDI) olanağını kullanır. IBM MQ classes for Jakarta Messaging uygulamaları, bir dizin hizmetinden yönetilen nesneleri almak için JNDI ' yi de kullanabilir.

Not: Jakarta Messaging 3.0için, IBM MQ Explorerkullanarak JNDI ' yi yönetemezsiniz. JNDI yönetimi, **JMSAdmin**' ın **JMS30Admin**olan Jakarta Messaging 3.0 çeşitlemesi tarafından desteklenir.

Hizmet sağlayıcı, dizin hizmetiyle JNDI çağrılarını eşleyerek bir dizin hizmetine erişim sağlayan koddur. `fscontext.jar` ve `providerutil.jar` dosyalarındaki bir dosya sistemi hizmet sağlayıcısı IBM MQ classes for Jakarta Messagingile birlikte sağlanır. Dosya sistemi hizmet sağlayıcısı, yerel dosya sistemine dayalı olarak bir dizin hizmetine erişim sağlar.

LDAP sunucusuna dayalı bir dizin hizmeti kullanmayı planlıyorsanız, bir LDAP sunucusu kurmanız ve yapılandırmanız ya da var olan bir LDAP sunucusuna erişiminiz olması gerekir. Özellikle, Java nesnelerini depolamak için LDAP sunucusunu yapılandırmanız gerekir. LDAP sunucunuzu kurma ve yapılandırma hakkında bilgi için sunucuyla birlikte sağlanan belgelere bakın.

JMS 2.0 IBM MQ classes for JMS için önkoşullar

Bu konu, IBM MQ classes for JMS' yi kullanmadan önce bilmeniz gerekenleri size bildirir. IBM MQ classes for JMS uygulamalarını geliştirmek ve çalıştırmak için önkoşul olarak belirli yazılım bileşenlerine gereksinim duyarsınız.

IBM MQ classes for JMSile ilgili önkoşullar hakkında bilgi için bkz. IBM MQ için Sistem Gereksinimleri.

IBM MQ classes for JMS uygulamalarını geliştirmek için bir Java SE Software Development Kit (SDK) gerekir. İşletim sisteminiz tarafından desteklenen JDK ' lere ilişkin ayrıntılar için bkz. IBM MQ için Sistem Gereksinimleri.

IBM MQ classes for JMS uygulamalarını çalıştırmak için aşağıdaki yazılım bileşenlerine gereksinim duyarsınız:

- Bir IBM MQ kuyruk yöneticisi.
- Uygulamaları çalıştırdığınız her sistem için bir Java runtime environment (JRE).
- IBM i İşletim sisteminin 30. seçeneği olan IBM iQshell için.
- z/OS z/OSiçin, z/OS UNIX System Services (z/OS UNIX).

IBM JSSE sağlayıcısı, FIPS onaylı bir şifreleme sağlayıcısı içerir; bu nedenle, anında kullanıma hazır FIPS 140-2 uyumluluğu için programlı olarak yapılandırılabilir. Bu nedenle, FIPS 140-2 uyumluluğu doğrudan IBM MQ classes for Java ve IBM MQ classes for JMStarafından desteklenebilir.

OracleJSSE sağlayıcısının içinde yapılandırılmış bir FIPS sertifikalı şifreleme sağlayıcısı olabilir, ancak bu hemen kullanıma hazır değildir ve programlı yapılandırma için kullanılamaz. Bu nedenle, bu durumda IBM MQ classes for Java ve IBM MQ classes for JMS , FIPS 140-2 uyumluluğunu doğrudan etkinleştiremez. Bu tür bir uyumluluğu el ile etkinleştirebilirsiniz, ancak IBM şu anda bu konuda rehberlik sağlayamaz.

IPv6 adresleri Java sanal makineniz (JVM) ve işletim sisteminizdeki TCP/IP uygulaması tarafından destekleniyorsa, IBM MQ classes for JMS uygulamalarınızda Internet Protocol sürüm 6 (IPv6) adreslerini kullanabilirsiniz. IBM MQ JMS yönetim aracı (bkz. Yönetim aracını kullanarak JMS nesnelerini yapılandırma) IPv6 adreslerini de kabul eder.

IBM MQ JMS yönetim aracı ve IBM MQ Explorer , yönetilen nesneleri depolayan bir dizin hizmetine erişmek için Java Naming Directory Interface (JNDI) olanağını kullanır. IBM MQ classes for JMS uygulamaları, bir dizin hizmetinden yönetilen nesneleri almak için JNDI ' yi de kullanabilir. Hizmet sağlayıcı, dizin hizmetiyle JNDI çağrılarını eşleyerek bir dizin hizmetine erişim sağlayan koddur. `fscontext.jar` ve `providerutil.jar` dosyalarındaki bir dosya sistemi hizmet sağlayıcısı IBM MQ classes for JMSile birlikte sağlanır. Dosya sistemi hizmet sağlayıcısı, yerel dosya sistemine dayalı olarak bir dizin hizmetine erişim sağlar.

LDAP sunucusuna dayalı bir dizin hizmeti kullanmayı planlıyorsanız, bir LDAP sunucusu kurmanız ve yapılandırmanız ya da var olan bir LDAP sunucusuna erişiminiz olması gerekir. Özellikle, Java nesnelerini depolamak için LDAP sunucusunu yapılandırmanız gerekir. LDAP sunucunuzu kurma ve yapılandırma hakkında bilgi için sunucuyla birlikte sağlanan belgelere bakın.

IBM MQ classes for JMS/Jakarta Messaging ürününü kurma ve yapılandırma

Bu bölümde, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingürününü kurduğunuzda oluşturulan dizinler ve dosyalar açıklanır ve kuruluştan sonra IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging ' in nasıl yapılandırılacağı anlatılır.

İlgili kavramlar

“IBM MQ kaynak bağdaştırıcısının kullanılması” sayfa 419

Kaynak bağdaştırıcısı, bir uygulama sunucusunda çalışan uygulamaların IBM MQ kaynaklarına erişmesine izin verir. Gelen ve giden iletişimi destekler.

IBM MQ classes for JMS için kurulu olan

IBM MQ classes for JMSkurulurken bir dizi dosya ve dizin oluşturulur. Windows' ta, ortam değişkenleri otomatik olarak ayarlanarak kuruluş sırasında bazı yapılandırma gerçekleştirilir. Diğer platformlarda

ve belirli Windows ortamlarında IBM MQ classes for JMS uygulamalarını çalıştırmadan önce ortam değişkenlerini ayarlamanız gerekir.

Çoğu işletim sistemi için, IBM MQ classes for JMS , IBM MQürününü kurduğunuzda isteğe bağlı bir bileşen olarak kurulur.

IBM MQkuruluşu hakkında daha fazla bilgi için bkz:

Multi Kuruluş IBM MQ

z/OS Kuruluş IBM MQ for z/OS

Önemli: "IBM MQ classes for JMS/Jakarta Messaging yeniden dizinlenebilir JAR dosyaları" sayfa 86içinde açıklanan yeniden taşınabilir JAR dosyalarının yanı sıra, IBM MQ classes for JMS JAR dosyalarının ya da yerel kitaplıkların başka makinelere kopyalanması ya da IBM MQ classes for JMS ' in kurulu olduğu bir makinede farklı bir yere kopyalanması desteklenmez.

Kuruluş dizinleri

Çizelge 5 sayfa 85 , IBM MQ classes for JMS dosyalarının her platformda nereye kurulduğunu gösterir.

Çizelge 5. IBM MQ classes for JMS kuruluş dizinleri

Hizmet olarak sunulan	**Dizin**
Linux AIX AIX and Linux	*MQ_INSTALLATION_PATH*`/java`
Windows Windows	*MQ_INSTALLATION_PATH*`\java`
IBM i IBM i	`/QIBM/ProdData/mqm/java`
z/OS z/OS	*MQ_INSTALLATION_PATH*`/java`

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Kuruluş dizini aşağıdaki içeriği içerir:

- *MQ_INSTALLATION_PATH*`\java\lib` dizinindeki yeniden yerleştirilebilir JAR dosyaları da içinde olmak üzere IBM MQ classes for JMS JAR dosyaları.
- Java Yerel Arabirimi 'ni kullanan uygulamalar tarafından kullanılan IBM MQ yerel kitaplıkları.

  32 bit yerel kitaplıklar *MQ_INSTALLATION_PATH*`\java\lib` dizinine kurulur ve 64 bit yerel kitaplıklar *MQ_INSTALLATION_PATH*\java\lib64 dizininde bulunabilir.

  IBM MQ yerel kitaplıklarıyla ilgili daha fazla bilgi için bkz. "Java Native Interface (JNI) kitaplıklarının yapılandırılması" sayfa 92.
- "IBM MQ classes for JMS/Jakarta Messaging ile sağlanan komut dosyaları" sayfa 118içinde açıklanan ek komut dosyaları. Bu komut dosyaları *MQ_INSTALLATION_PATH*`\java\bin` dizininde bulunur.
- IBM MQ classes for JMS API belirtimleri. Javadoc aracı, API belirtimlerini içeren HTML sayfalarını oluşturmak için kullanılır.

  HTML sayfaları *MQ_INSTALLATION_PATH*`\java\doc\WMQJMSClasses` dizininde bulunur:

  – ALW AIX, Linux, and Windowsüzerinde, bu alt dizin tek tek HTML sayfalarını içerir.

  – IBM i IBM i' da, HTML sayfaları `wmqjms_javadoc.jar`adlı bir dosyada bulunur.

  – z/OS z/OS' da, HTML sayfaları `wmqjms_javadoc.jar`adlı bir dosyada bulunur.
- OGSi için destek. OSGi paketleri `java\lib\OSGi` dizinine kurulur ve "IBM MQ classes for JMS ile OSGi desteği" sayfa 119içinde açıklanır.

- Herhangi bir Java Platform, Enterprise Edition 7 (Java EE 7) ya da Jakarta EE uyumlu uygulama sunucusuna konuşlandırılabilecek IBM MQ kaynak bağdaştırıcısı.

  IBM MQ kaynak bağdaştırıcısı *MQ_INSTALLATION_PATH*`\java\lib\jca` dizininde bulunur; daha fazla bilgi için bkz. “IBM MQ kaynak bağdaştırıcısının kullanılması” sayfa 419

- Windows Windowsişletim sistemlerinde, hata ayıklamak için kullanılabilecek simgeler *MQ_INSTALLATION_PATH*`\java\lib\symbols` dizinine kurulur.

Kuruluş dizini, diğer IBM MQ bileşenlerine ait bazı dosyaları da içerir.

Örnek Uygulamalar

JMS 2.0 Bazı örnek uygulamalar IBM MQ classes for JMSile sağlanır. Çizelge 6 sayfa 86 , örnek uygulamaların her altyapıda nereye kurulduğunu gösterir.

JM 3.0 IBM MQ classes for Jakarta Messagingiçin yeni örnekler hazırlanıyor.

JMS 2.0

Çizelge 6. IBM MQ classes for JMS için örnek dizinleri

Hizmet olarak sunulan	**Dizin**
Linux AIX AIX and Linux	*MQ_INSTALLATION_PATH*`/samp/jms`
Windows Windows	*MQ_INSTALLATION_PATH*`\tools\jms`
IBM i IBM i	`/QIBM/ProdData/mqm/java/samples/jms`
z/OS z/OS	*MQ_INSTALLATION_PATH*`/java/samples/jms`

Bu çizelgede *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Kuruluştan sonra, uygulamaları derlemek ve çalıştırmak için bazı yapılandırma görevlerini gerçekleştirmeniz gerekebilir.

“IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama” sayfa 89 , örnek IBM MQ classes for JMS uygulamalarını çalıştırmak için gereken sınıf yolunu açıklar. Bu konuda, özel durumlarda başvurulması gereken ek JAR dosyaları ve IBM MQ classes for JMSile birlikte verilen komut dosyalarını çalıştırmak için ayarlamanız gereken ortam değişkenleri de açıklanmaktadır.

Bir uygulamanın izlenmesi ve günlüğe kaydedilmesi gibi özellikleri denetlemek için bir yapılandırma özellikleri dosyası sağlamanız gerekir. IBM MQ classes for JMS yapılandırma özellikleri dosyası, “IBM MQ classes for JMS/Jakarta Messaging yapılandırma dosyası” sayfa 94içinde açıklanmıştır.

İlgili kavramlar

Kaynak bağdaştırıcısı konuşlandırılırken sorunlar oluştu

İlgili görevler

“IBM MQ classes for JMS örnek uygulamalarının kullanılması” sayfa 114
IBM MQ classes for JMS örnek uygulamaları, JMS API ' nin ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluş ve ileti sistemi sunucunuzun kurulumunu doğrulamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

IBM MQ classes for JMS/Jakarta Messaging yeniden dizinlenebilir JAR dosyaları
Yeniden taşınabilir JAR dosyaları, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingkomutunu çalıştırması gereken sistemlere taşınabilir.

Önemli:

- Relocatable JAR dosyalarındaaçıklanan taşınabilir JAR dosyalarının yanı sıra, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging JAR dosyalarının ya da yerel kitaplıklarının başka makinelere kopyalanması ya da IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging ' in kurulu olduğu bir makinede farklı bir yere kopyalanması desteklenmez.
- Yeniden yerleştirilebilir JAR dosyalarını, WebSphere Application Server ya da WebSphere Libertygibi Java EE uygulama sunucularında konuşlandırılan uygulamalara eklemeyin. Bu ortamlarda, IBM MQ kaynak bağdaştırıcısı konuşlandırılmalı ve kullanılmalıdır. WebSphere Application Server ' in IBM MQ kaynak bağdaştırıcısını yerleştirdiğini, bu nedenle bu ortama el ile yerleştirmeye gerek olmadığını unutmayın.
- Sınıf yükleyici çakışmalarını önlemek için, relocatable JAR dosyalarının aynı Java yürütme ortamı içindeki birden çok uygulama içinde paketlenmesi önerilmez. Bu senaryoda, IBM MQ yeniden dizinlenebilir JAR dosyalarını Java yürütme ortamının sınıf yolunda (classpath) kullanılır kılın.
- Yeniden dizinlenebilir JAR dosyalarını uygulamalarınızda paketliyorsanız, tüm önkoşul JAR dosyalarını Relocatable JAR filesbaşlıklı konuda açıklandığı gibi eklemeyi doğrulayın. IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging ' nin güncel ve bilinen sorunların yeniden aracılık edildiğinden emin olmak için, paket JAR dosyalarını uygulama bakımının bir parçası olarak güncellemek için uygun yordamlara sahip olduğunuzdan da emin olmanız gerekir.

Yeniden dizinlenebilir JAR dosyaları

Bir kuruluş içinde, aşağıdaki dosyalar IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingişletim sistemini çalıştırması gereken sistemlere taşınabilir:

- V 9.3.0 V 9.3.0 `bcpkix-jdk15to18.jar` "4" sayfa 87
- V 9.3.5 `bcpkix-jdk18on.jar` "3" sayfa 87
- V 9.3.0 V 9.3.0 `bcprov-jdk15to18.jar` "4" sayfa 87
- V 9.3.5 `bcprov-jdk18on.jar` "3" sayfa 87
- V 9.3.0 V 9.3.0 `bcutil-jdk15to18.jar` "4" sayfa 87
- V 9.3.5 `bcutil-jdk18on.jar` "3" sayfa 87
- JMS 2.0 `com.ibm.mq.allclient.jar` "1" sayfa 87
- V 9.3.0 V 9.3.0 JM 3.0 `com.ibm.mq.jakarta.client.jar` "2" sayfa 87
- Removed V 9.3.3 `com.ibm.mq.traceControl.jar`
- `fscontext.jar`
- V 9.3.3 `jackson-annotations.jar`
- V 9.3.3 `jackson-core.jar`
- V 9.3.3 `jackson-databind.jar`
- `jakarta.jms-api.jar`
- `jms.jar`
- `org.json.jar`
- `providerutil.jar`

Notlar:

1. JMS 2.0 ve JMS 1.1
2. Jakarta Messaging 3.0
3. Continuous Delivery Kaynak: IBM MQ 9.3.5
4. Long Term Support ve Continuous Delivery önce IBM MQ 9.3.5

JMS JAR dosyaları

V 9.3.0 V 9.3.0 `jms.jar` , JMS 1.1 ve JMS 2.0 arabirimlerini içerir-bunlar `javax.jms.*`olarak adlandırılır.

V 9.3.0 V 9.3.0 JM 3.0 `jakarta.jms-api.jar` , Jakarta Messaging 3.0 arabirimlerini içerir-bunlar `jakarta.jms.*`olarak adlandırılır.

fscontext.jar ve providerutil.jar

Uygulamanız bir dosya sistemi bağlamı kullanarak JNDI aramaları gerçekleştiriyorsa, `fscontext.jar` ve `providerutil.jar` dosyaları gereklidir.

Bouncy Castle güvenlik sağlayıcısı ve CMS destek JAR dosyaları

Bouncy Castle güvenlik sağlayıcısı ve CMS destek JAR dosyaları gereklidir. Daha fazla bilgi için bkz. Support for non-IBM JRE with AMS.

V 9.3.5 Continuous Delivery from IBM MQ 9.3.5için aşağıdaki JAR dosyaları gereklidir:

- `bcpkix-jdk18on.jar`
- `bcprov-jdk18on.jar`
- `bcutil-jdk18on.jar`

LTS IBM MQ 9.3.5öncesinde Long Term Support ve Continuous Delivery için aşağıdaki JAR dosyaları gereklidir:

- `bcpkix-jdk15to18.jar`
- `bcprov-jdk15to18.jar`
- `bcutil-jdk15to18.jar`

org.json.jar

IBM MQ classes for JMS uygulamanız JSON biçiminde bir CCDT kullanıyorsa `org.json.jar` dosyası gereklidir.

com.ibm.mq.allclient.jar ve com.ibm.mq.jakarta.client.jar

`com.ibm.mq.allclient.jar` ve `com.ibm.mq.jakarta.client.jar` dosyaları, IBM MQ classes for JMS, IBM MQ classes for Jakarta Messaging, IBM MQ classes for Javave PCF ve Üstbilgi Sınıflarını içerir. Bu JAR dosyasını yeni bir konuma taşırsanız, yeni IBM MQ düzeltme paketleriyle bu yeni konumu korumak için gereken adımları gerçekleştirdiğinizden emin olun. Ayrıca, geçici bir düzeltme alıyorsanız, dosyaların kullanılmasının IBM Desteği tarafından bilindiğinden emin olun.

`com.ibm.mq.allclient.jar` ve `com.ibm.mq.jakarta.client.jar`dosyalarının sürümünü belirlemek için aşağıdaki komutu kullanın:

V 9.3.0 V 9.3.0 JM 3.0

```
java -jar com.ibm.mq.jakarta.client.jar
```

JMS 2.0

```
java -jar com.ibm.mq.allclient.jar
```

Aşağıdaki örnek, bu komutun bazı örnek çıkışlarını göstermektedir:

```
C:\Program Files\IBM\MQ_1\java\lib>java -jar com.ibm.mq.allclient.jar
Name:       Java Message Service Client
Version:    9.3.0.0
```

```
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       WebSphere MQ classes for Java Message Service
Version:    9.3.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       WebSphere MQ JMS Provider
Version:    9.3.0.0
Level:      p000-L140428.1 mqjbnd=p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       Common Services for Java Platform, Standard Edition
Version:    9.3.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```

jackson-annotations.jar, jackson-core.jar ve jackson-databind.jar

V 9.3.3

IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging uygulamanız bir kuyruk yöneticisine güvenli TLS bağlantıları oluşturursa, üç Jackson JAR dosyası gereklidir.

IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama
IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging uygulamalarını derlemeden ve çalıştırmadan önce, **CLASSPATH** ortam değişkeninizin ayarı IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging Java arşiv (JAR) dosyasını içermelidir. Gereksinimlerinize bağlı olarak, diğer JAR dosyalarını sınıf yolunuza eklemeniz gerekebilir. IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingile sağlanan komut dosyalarını çalıştırmak için diğer ortam değişkenleri ayarlanmalıdır.

Başlamadan önce

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Aynı uygulamada hem Jakarta Messaging 3.0 API hem de JMS 2.0 API ' nin kullanılması desteklenmez. Daha fazla bilgi için, bkz. JMS/Jakarta İleti Sistemi için IBM MQ sınıflarının kullanılması.

Önemli: Java seçenek *-Xbootclasspath* 'in IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging ' i içerecek şekilde ayarlanması desteklenmez.

Bu görev hakkında

IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging uygulamalarını derlemek ve çalıştırmak için, aşağıdaki çizelgelerde gösterildiği gibi, altyapınızın ve Java ileti sistemi sürümünüze ilişkin **CLASSPATH** ayarını kullanın. Alternatif olarak, ortam değişkenini kullanmak yerine **java** komutunda sınıf yolunu belirtebilirsiniz.

JMS 2.0 IBM MQ classes for JMSiçin ayar, IBM MQ classes for JMS örnek uygulamalarını derleyebilmeniz ve çalıştırabilmeniz için örnekler dizinini içerir.

JM 3.0 IBM MQ classes for Jakarta Messagingiçin yeni örnekler hazırlanıyor.

JM 3.0

*Çizelge 7. Jakarta Messaging 3.0 uygulamalarının derlenmesi ve çalıştırılması IBM MQ classes for Jakarta Messaging için **CLASSPATH** ayarları*	
Hizmet olarak sunulan	**CLASSPATH ayar**
AIX	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jakarta.client.jar:
Linux	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jakarta.client.jar:
IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jakarta.client.jar:
Windows	CLASSPATH= *MQ_INSTALLATION_PATH*\java\lib\com.ibm.mq.jakarta.client.jar;
z/OS	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jakarta.client.jar;

JMS 2.0

*Çizelge 8. Örnek uygulamalar da içinde olmak üzere IBM MQ classes for JMS uygulamalarını derlemek ve çalıştırmak için JMS 2.0 ile ilgili **CLASSPATH** ayarları*	
Hizmet olarak sunulan	**CLASSPATH ayarı**
AIX	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.allclient.jar: *MQ_INSTALLATION_PATH*/samp/jms/samples:
Linux	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.allclient.jar: *MQ_INSTALLATION_PATH*/samp/jms/samples:
IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mq.allclient.jar: /QIBM/ProdData/mqm/java/samples/jms/samples:
Windows	CLASSPATH= *MQ_INSTALLATION_PATH*\java\lib\com.ibm.mq.allclient.jar; *MQ_INSTALLATION_PATH*\tools\jms\samples;
z/OS	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.allclient.jar: *MQ_INSTALLATION_PATH*/java/samples/jms/samples:

Bu çizelgelerde *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

`com.ibm.mq.jakarta.client.jar` ya da `com.ibm.mq.allclient.jar` JAR dosyasının bildirgesi, IBM MQ classes for JMS uygulamalarının gerektirdiği diğer JAR dosyalarının çoğunun başvurularını içerir; bu nedenle, bu JAR dosyalarını sınıf yolunuza eklemenize gerek yoktur. Bu JAR dosyaları, bir dizin hizmetinden ve Java Transaction API (JTA) kullanan uygulamalardan yönetilen nesneleri almak için Java Naming Directory Interface (JNDI) kullanan uygulamaların gerektirdiği dosyaları içerir.

Ancak, aşağıdaki durumlarda sınıf yolunuza ek JAR dosyaları eklemeniz gerekir:

- `com.ibm.mq` paketinde tanımlanan kanal çıkış arabirimlerini uygulayan kanal çıkış sınıflarını kullanıyorsanız, `com.ibm.mq.exits` paketinde tanımlananlar yerine, sınıf yolunuza IBM MQ classes for Java JAR dosyasını `com.ibm.mq.jar`eklemeniz gerekir.
- Uygulamanız bir dizin hizmetinden denetlenen nesneleri almak için JNDI kullanıyorsa, sınıf yolunuza aşağıdaki JAR dosyalarını da eklemelisiniz:

  - `fscontext.jar`
  - `providerutil.jar`
- Uygulamanız JTA kullanıyorsa, sınıf yolunuza `jta.jar` eklemelisiniz.

Not: Bu ek JAR dosyaları yalnızca uygulamalarınızı derlemek için gereklidir, çalıştırmak için gerekli değildir.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging ile sağlanan komut dosyaları aşağıdaki ortam değişkenlerini kullanır:

MQ_JAVA_DATA_PATH
Bu ortam değişkeni, günlük ve izleme çıkışına ilişkin dizini belirtir.

MQ_JAVA_KURULUŞ_YOLU
Bu ortam değişkeni, IBM MQ classes for JMS ' in kurulu olduğu dizini belirtir.

MQ_JAVA_LIB_YOLU
Bu ortam değişkeni, önceki çizelgelerde gösterildiği gibi IBM MQ classes for JMS kitaplıklarının saklandığı dizini belirtir.

Yordam

- Windows

  Windowsişletim sistemlerinde, IBM MQkurulduktan sonra **setmqenv**komutunu çalıştırın.

  Önce bu komutu çalıştırmazsanız, bir **dspmqver** komutu verdiğinizde aşağıdaki hata iletisi görüntülenebilir:

  ```
  AMQ8351: IBM MQ Java ortamı yapılandırılmadı
  doğru ya da IBM MQ JRE özelliği kurulmadı.
  ```

  Not: IBM MQ Java runtime environment (JRE) programını kurmadıysanız bu ileti beklenir (Ek Windows özellikleri önkoşul denetimikonusuna bakın).

- Linux AIX

  AIX and Linux sistemlerinde ortam değişkenlerini kendiniz ayarlayın:

  JMS 2.0 JMS 2.0için, ortam değişkenlerini ayarlamak üzere aşağıdaki komut dosyalarından birini kullanın:

  - 32 bit JVM kullanıyorsanız, `setjmsenv`komut dosyasını kullanın.
  - Bir AIX ya da Linux sisteminde 64 bit JVM kullanıyorsanız, `setjmsenv64`komut dosyasını kullanın.

  JM 3.0 Jakarta Messaging 3.0için, ortam değişkenlerini ayarlamak üzere aşağıdaki komut dosyalarından birini kullanın:

  - 32 bit JVM kullanıyorsanız, `setjms30env`komut dosyasını kullanın.
  - 64 bit JVM kullanıyorsanız, `setjms30env64`komut dosyasını kullanın.

  Bu komut dosyaları *MQ_INSTALLATION_PATH*`/java/bin` dizininde bulunur; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

  Bu komut dosyalarını çeşitli şekillerde kullanabilirsiniz. Komut dosyasını, tabloda gösterildiği gibi gerekli ortam değişkenlerini ayarlamak için temel olarak kullanabilir ya da bunları bir metin düzenleyicisi kullanarak `.profile` ' e ekleyebilirsiniz. Tipik olmayan bir kurulumunuz varsa, komut dosyası içeriğini gerektiği şekilde düzenleyin. Diğer bir seçenek olarak, komut dosyasını JMS başlatma komut dosyalarının çalıştırılacağı her oturumda çalıştırabilirsiniz. Bu seçeneği belirlerseniz, komut dosyasını JMS doğrulama işlemi sırasında başlatmış olduğunuz her kabuk penceresinde çalıştırmanız gerekir:

  - JMS 2.0 JMS 2.0için `. ./setjmsenv` ya da `../setjmsenv64`yazın.
  - JM 3.0 Jakarta Messaging 3.0için `. ./setjms30env` ya da `. ./setjms30env64`yazın.

IBM i IBM iişletim sistemlerinde **QIBM_MULTI_THREADED** ortam değişkenini Yolarak ayarlamanız gerekir. Birden çok iş parçacıklı uygulamaları, tek iş parçacıklı uygulamaları çalıştırdığınız şekilde çalıştırabilirsiniz. Daha fazla bilgi için bkz. IBM MQ ' nun Java ve JMS ile ayarlanması.

İlgili görevler

"IBM MQ classes for JMS örnek uygulamalarının kullanılması" sayfa 114
IBM MQ classes for JMS örnek uygulamaları, JMS API ' nin ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluş ve ileti sistemi sunucunuzun kurulumunu doğrulamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

İlgili başvurular

"IBM MQ classes for JMS/Jakarta Messaging ile sağlanan komut dosyaları" sayfa 118
IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingkullanılırken gerçekleştirilmesi gereken ortak görevlere yardımcı olmak için bir dizi komut dosyası sağlanır.

Java Native Interface (JNI) kitaplıklarının yapılandırılması

Bağ tanımları iletimini kullanarak bir kuyruk yöneticisine bağlanan ya da istemci iletimini kullanarak bir kuyruk yöneticisine bağlanan ve Javadışındaki dillerde yazılmış kanal çıkış programlarını kullanan IBM MQ classes for JMS uygulamalarının, Java Native Interface (JNI) kitaplıklarına erişime izin veren bir ortamda çalıştırılması gerekir.

Başlamadan önce

WebSphere Application Server ortamının kullanılmasına ilişkin ek bilgi için IBM MQ ileti alışverişi sağlayıcısını yerli kitaplık bilgileriyle yapılandırma başlıklı konuya bakın.

Bu görev hakkında

Bu ortamı ayarlamak için, Java Virtual Machine (JVM) olanağının IBM MQ classes for JMS uygulamasını başlatmadan önce mqjbnd kitaplığını yükleyebilmesi için ortamın kitaplık yolunu yapılandırmanız gerekir.

IBM MQ , iki Java Native Interface (JNI) kitaplığı sağlar:

mqjbnd
Bu kitaplık, bağ tanımları iletimini kullanarak kuyruk yöneticisine bağlanan uygulamalar tarafından kullanılır. IBM MQ classes for JMS ile kuyruk yöneticisi arasındaki arabirimi sağlar. IBM MQ 9.3 ile kurulan mqjbnd kitaplığı, herhangi bir IBM MQ 9.3 (ya da daha önceki) kuyruk yöneticisine bağlanmak için kullanılabilir.

mqjexitstub02
Bir uygulama istemci iletimini kullanarak bir kuyruk yöneticisine bağlandığında ve Javadışında bir dilde yazılmış bir kanal çıkış programını kullandığında, mqjexitstub02 kitaplığı IBM MQ classes for JMS tarafından yüklenir.

Belirli platformlarda IBM MQ , bu JNI kitaplıklarının 32 bit ve 64 bit sürümlerini kurar. Her bir platforma ilişkin kitaplıkların yeri Tablo 1içinde gösterilir.

Çizelge 9. Her platform için IBM MQ classes for JMS kitaplıklarının konumu

Hizmet olarak sunulan	IBM MQ classes for JMS kitaplıklarını içeren dizin
AIX AIX Linux Linux (POWER, x86-64 ve zSeries s390x platformları)	*MQ_INSTALLATION_PATH*/java/lib (32 bit kitaplık) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitlik kitaplıklar)

Çizelge 9. Her platform için IBM MQ classes for JMS kitaplıklarının konumu (devamı var)	
Hizmet olarak sunulan	**IBM MQ classes for JMS kitaplıklarını içeren dizin**
Windows Windows	*MQ_INSTALLATION_PATH*\java\lib (32 bitlik kitaplıklar) *MQ_INSTALLATION_PATH*\java\lib64 (64 bitlik kitaplıklar)
z/OS z/OS	*MQ_INSTALLATION_PATH*/java/lib (31 bit ve 64 bit kitaplıklar)

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Not: z/OS z/OSişletim sistemlerinde 31 bit ya da 64 bit Java Virtual Machine (JVM) kullanabilirsiniz. Hangi JNI kitaplıklarının kullanılacağını belirtmeniz gerekmez; IBM MQ classes for JMS kendisi için yüklenecek JNI kitaplıklarını saptayabilir.

Yordam

1. JVM ' nin **java.library.path** özelliğini iki şekilde yapılandırın:
   - JVM bağımsız değişkenini aşağıdaki örnekte gösterildiği gibi belirterek:

     ```
     -Djava.library.path=path_to_library_directory
     ```

     Linux Örneğin, varsayılan bir konum kuruluşu için Linux üzerinde 64 bit JVM için şunu belirtin:

     ```
     -Djava.library.path=/opt/mqm/java/lib64
     ```

   - JVM 'nin kendi `java.library.path`' ini kuracağı şekilde kabuk ortamını yapılandırarak. Bu yol, platforma ve IBM MQ' i kurduğunuz konuma göre değişir. Örneğin, 64 bitlik JVM ve varsayılan IBM MQ kuruluş konumu için aşağıdaki ayarları kullanabilirsiniz:

     AIX
     ```
     export LIBPATH=/usr/mqm/java/lib64:$LIBPATH
     ```

     Linux
     ```
     export LD_LIBRARY_PATH=/opt/mqm/java/lib64:$LD_LIBRARY_PATH
     ```

     Windows
     ```
     set PATH=C:\Program Files\IBM\MQ\java\lib64;%PATH%
     ```

   Ortam doğru yapılandırılmadığında gördüğünüz kural dışı durum yığınına örnek olarak şunlar verilebilir:

```
Nedeni: com.ibm.mq.jmqi.local.LocalMQ$4: CC=2;RC=2495;
AMQ8598: WebSphere MQ yerel JNI kitaplığı yüklenemedi: 'mqjbnd'.
    com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1268)
    com.ibm.mq.jmqi.local.LocalMQ$1.run(LocalMQ.java:309)
    java.security.AccessController.doPrivileged(AccessController.java:400)
    com.ibm.mq.jmqi.local.LocalMQ.initialise_inner(LocalMQ.java:259)
    com.ibm.mq.jmqi.local.LocalMQ.initialise(LocalMQ.java:221)
    com.ibm.mq.jmqi.local.LocalMQ. < init> (LocalMQ.java:1350)
    com.ibm.mq.jmqi.local.LocalServer. < init> (LocalServer.java:230)
    sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)

sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:86)


sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.jav
a:58)
    java.lang.reflect.Constructor.newInstance(Constructor.java:542)
    com.ibm.mq.jmqi.JmqiEnvironment.getInstance(JmqiEnvironment.java:706)
    com.ibm.mq.jmqi.JmqiEnvironment.getMQI(JmqiEnvironment.java:640)

com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection(WMQConnectionF
```

```
actory.java:8437)
    ... 7 tane daha
Nedeni: java.lang.UnsatisfiedLinkError: mqjbnd ( java.library.pathiçinde bulunamadı)
    java.lang.ClassLoader.loadLibraryWithPath(ClassLoader.java:1235)
    java.lang.ClassLoader.loadLibraryWithClassLoader(ClassLoader.java:1205)
    java.lang.System.loadLibrary(System.java:534)
    com.ibm.mq.jmqi.local.LocalMQ.loadLib(LocalMQ.java:1240)
    ... 20 tane daha.
```

2. 32 bit ya da 64 bit ortam ayarlandıktan sonra şu komutu kullanarak IBM MQ classes for JMS uygulamasını başlatın:

   ```
   java application-name
   ```

   Burada *uygula-adı* , çalıştırılacak IBM MQ classes for JMS uygulamasının adıdır.

   Aşağıdaki IBM MQ neden kodu 2495 'i (MQRC_MODULE_NOT_FOUND) içeren bir kural dışı durum IBM MQ classes for JMS tarafından yayınlandı:

   - IBM MQ classes for JMS uygulaması 32 bit Java runtime environmentiçinde çalıştırılır ve 32 bit Java runtime environment , 64 bit Java Native Library ürününü yükleyemediğinden IBM MQ classes for JMSiçin 64 bit ortam ayarlanmıştır.
   - IBM MQ classes for JMS uygulaması 64 bit Java runtime environmentiçinde çalıştırılır ve 64 bit Java runtime environment , 32 bit Java Native Library ürününü yükleyemediğinden IBM MQ classes for JMSiçin 32 bit ortam ayarlandı.

IBM MQ classes for JMS/Jakarta Messaging yapılandırma dosyası
IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging yapılandırma dosyaları, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging' yi yapılandırmak için kullanılan özellikleri belirtir.

Not: Yapılandırma dosyasında tanımlanan özellikler, JVM sistem özellikleri olarak da ayarlanabilir. Bir özellik hem yapılandırma dosyasında hem de sistem özelliği olarak ayarlanırsa, sistem özelliği öncelikli olur. Bu nedenle, gerekirse, yapılandırma dosyasındaki herhangi bir özelliği **java** komutunda sistem özelliği olarak belirterek geçersiz kılabilirsiniz.

IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılandırma dosyasının biçimi, standart bir Java özellikler dosyasının biçimidir. IBM MQ classes for JMS kuruluş dizininin `bin` altdizininde `jms.config` adlı örnek bir yapılanış kütüğü bulunur. Bu dosya, desteklenen tüm özellikleri ve varsayılan değerlerini belgeler.

Bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılandırma dosyasının adını ve konumunu seçebilirsiniz. Uygulamanızı başlattığınızda, aşağıdaki biçimi kullanan bir **java** komutu kullanın:

```
java -Dcom.ibm.msg.client.config.location= config_file_url application_name
```

Komutta *config_file_url* , IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılandırma dosyasının adını ve konumunu belirten tek tip bir kaynak konum belirleyicidir (URL). Şu tiplerin URL ' leri desteklenir: http, file, ftp ve jar.

Aşağıda bir **java** komutu örneği verilmiştir:

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config MyAppClass
```

Bu komut, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılandırma dosyasını yerel Windows sisteminde `D:\mydir\mjms.config` dosyası olarak tanımlar.

Bir uygulama başlatıldığında, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılandırma dosyasının içeriğini okur ve belirtilen özellikleri bir iç özellik deposunda saklar. **java** komutu bir yapılandırma dosyasını belirtmezse ya da yapılandırma dosyası bulunamazsa, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging tüm özellikler için varsayılan değerleri kullanır.

Bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılanış dosyası, bir uygulama ile bir kuyruk yöneticisi ya da aracı arasında desteklenen iletişimle birlikte kullanılabilir.

IBM MQ MQI client yapılandırma dosyasında belirtilen özelliklerin geçersiz kılınması

IBM MQ MQI client yapılandırma dosyası, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingürününü yapılandırmak için kullanılan özellikleri de belirtebilir. Ancak, IBM MQ MQI client yapılanış dosyasında belirtilen özellikler yalnızca, bir uygulama istemci kipinde bir kuyruk yöneticisine bağlandığında geçerlidir.

Gerekirse, bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılandırma dosyasında özellik olarak belirterek IBM MQ MQI client yapılandırma dosyasındaki herhangi bir özniteliği geçersiz kılabilirsiniz. IBM MQ MQI client yapılandırma dosyasındaki bir özniteliği geçersiz kılmak için, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılandırma dosyasında aşağıdaki biçime sahip bir giriş kullanın:

```
com.ibm.mq.cfg. stanza. propName = propValue
```

Girdideki değişkenler aşağıdaki anlamlara sahiptir:

kıta
Özniteliği içeren IBM MQ MQI client yapılandırma dosyasındaki dörtlünün adı

propName
IBM MQ MQI client yapılandırma dosyasında belirtilen özniteliğin adı

propValue
IBM MQ MQI client yapılandırma dosyasında belirtilen özniteliğin değerini geçersiz kılan özelliğin değeri

Alternatif olarak, özelliği **java** komutunda sistem özelliği olarak belirterek IBM MQ MQI client yapılandırma dosyasındaki bir özniteliği geçersiz kılabilirsiniz. Özelliği bir sistem özelliği olarak belirtmek için önceki biçimi kullanın.

Bir IBM MQ MQI client yapılandırma dosyasında yalnızca aşağıdaki öznitelikler IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingile ilgilidir. Diğer öznitelikleri belirtirseniz ya da geçersiz kılarsanız, bunun bir etkisi olmaz. Özellikle, istemci yapılandırma dosyasının KANAL kısmı içindeki `ChannelDefinitionFile` ve `ChannelDefinitionDirectory` ' nin kullanılmadığına dikkat edin. CCDT ' nin IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingile nasıl kullanılacağına ilişkin ayrıntılar için bkz. “IBM MQ classes for JMS ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 273 .

Çizelge 10. İstemci yapılanış dosyasının hangi kısmı hangi özniteliği içeriyor?

Kıta	**Öznitelik**
İstemci yapılanış kütüğünün KANAL kısmı	Put1DefaultAlwaysSync
İstemci yapılanış kütüğünün KANAL kısmı	DefRecon
İstemci yapılanış kütüğünün KANAL kısmı	ReconDelay
İstemci yapılanış kütüğünün KANAL kısmı	PasswordProtection
ClientExitİstemci yapılanış kütüğünün yol kısmı	ExitsDefaultYolu
ClientExitİstemci yapılanış kütüğünün yol kısmı	ExitsDefaultPath64
ClientExitİstemci yapılanış kütüğünün yol kısmı	JavaExitsClasspath
İstemci yapılanış kütüğünün JMQI kısmı	useMQCSPauthentication
İstemci yapılanış kütüğünün MessageBuffer kısmı	MaximumSize
İstemci yapılanış kütüğünün MessageBuffer kısmı	PurgeTime

Çizelge 10. İstemci yapılanış dosyasının hangi kısmı hangi özniteliği içeriyor? (devamı var)	
Kıta	**Öznitelik**
İstemci yapılanış kütüğünün MessageBuffer kısmı	UpdatePercentage
İstemci yapılanış kütüğünün TCP kısmı	ClntRcvBufSize
İstemci yapılanış kütüğünün TCP kısmı	ClntSndBufSize
İstemci yapılanış kütüğünün TCP kısmı	Connect_Timeout
İstemci yapılanış kütüğünün TCP kısmı	KeepAlive

IBM MQ MQI client yapılandırmasıyla ilgili daha fazla ayrıntı için bkz. IBM MQ MQI client yapılandırma dosyası, `mqclient.ini`

JMS izlemesini yapılandırmak için Java Standard Environment Trace (Standart Ortam İzleme) olanağının kullanılması
IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging izleme olanağını yapılandırmak için Java Standard Environment Trace Settings (Standart Ortam İzleme Ayarları) kısmına bakın.

com.ibm.msg.client.commonservices.trace.outputName = *traceOutputAd*
traceOutputName , izleme çıkışının gönderildiği dizin ve dosya adıdır.

Varsayılan olarak, izleme bilgileri uygulamanın yürürlükteki çalışma dizinindeki bir izleme dosyasına yazılır. İzleme kütüğünün adı, uygulamanın çalıştığı ortama bağlıdır:

- V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0' den, uygulama IBM MQ classes for Jakarta Messaging dosyasını `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0) ya da IBM MQ classes for JMS dosyasını `com.ibm.mq.allclient.jar` (JMS 2.0) yeniden yüklenebilir JAR dosyasından yüklüyorsa, izleme `mqjavaclient_%PID%.cl%u.trc`adlı bir dosyaya yazılır.
- IBM MQ 9.1.5 ve IBM MQ 9.1.0 Fix Pack 5' den:
  - Uygulama IBM MQ classes for JMS dosyasını yeniden yüklenebilir JAR dosyasından `com.ibm.mq.allclient.jar`yüklediyse, izleme `mqjavaclient_%PID%.cl%u.trc`adlı bir dosyaya yazılır.
  - Uygulama IBM MQ classes for JMS dosyasını `com.ibm.mqjms.jar`JAR dosyasından yüklüyorsa, izleme `mqjava_%PID%.cl%u.trc`adlı bir dosyaya yazılır.
- IBM MQ 9.0.0 Fix Pack 2' dan:
  - Uygulama IBM MQ classes for JMS dosyasını yeniden yüklenebilir JAR dosyasından `com.ibm.mq.allclient.jar`yüklediyse, izleme `mqjavaclient_%PID%.trc`adlı bir dosyaya yazılır.
  - Uygulama IBM MQ classes for JMS dosyasını `com.ibm.mqjms.jar`JAR dosyasından yüklüyorsa, izleme `mqjava_%PID%.trc`adlı bir dosyaya yazılır.
- For IBM MQ classes for JMS for IBM MQ 9.0.0 Fix Pack 1 or earlier, trace is written to a file called `mqjms_%PID%.trc`.

Burada *%PID%* , izlenmekte olan uygulamanın işlem tanıtıcısıdır ve *%u* , farklı Java sınıf yükleyicileri altında izleme çalıştıran iş parçacıkları arasında ayrım yapmak için benzersiz bir sayıdır.

Bir işlem tanıtıcısı yoksa, rasgele bir sayı oluşturulur ve başına `f`harfi eklenir. İşlem tanıtıcısını belirttiğiniz bir dosya adına eklemek için `%PID%`dizgisini kullanın.

Alternatif bir dizin belirtirseniz, dizin varolmalıdır ve bu dizin için yazma izniniz olmalıdır. Yazma izniniz yoksa, izleme çıkışı `System.err`' e yazılır.

com.ibm.msg.client.commonservices.trace.include = *includeList*
includeList , takip edilen paketlerin ve sınıfların ya da `ALL` ya da `NONE`özel değerlerinin bir listesidir.

Paket ya da sınıf adlarını noktalı virgülle (;) ayırın. *includeList* , varsayılan olarak ALLdeğerini alır ve IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingiçindeki tüm paketleri ve sınıfları izler.

Not: Bir paket ekleyebilir, ancak daha sonra bu paketin alt paketlerini dışlayabilirsiniz. Örneğin, a.b ve dışlama paketini a.b.xeklerseniz, izleme a.b.x ya da a.b.x.1değil, a.b.y ve a.b.ziçindeki her şeyi içerir.

com.ibm.msg.client.commonservices.trace.exclude = *excludeList*
excludeList , izlenmeyen paketlerin ve sınıfların ya da özel değerlerin ALL ya da NONElistesidir.

Paket ya da sınıf adlarını noktalı virgülle (;) ayırın. *excludeList* varsayılan olarak NONEdeğerine ayarlanır ve bu nedenle, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging içindeki hiçbir paket ve sınıf izlenmez.

Not: Bir paketi dışlayabilir, ancak daha sonra bu paketin alt paketlerini ekleyebilirsiniz. Örneğin, a.b paketini dışlar ve a.b.xpaketini dahil ederseniz, izleme a.b.y ya da a.b.zdeğil, a.b.x ve a.b.x.1içindeki her şeyi içerir.

Hem içerilen hem de dışlanan olarak, aynı düzeyde belirtilen herhangi bir paket ya da sınıf içerilir.

com.ibm.msg.client.commonservices.trace.maxBytes = *maxArrayByte*
maxArrayBytes , herhangi bir bayt dizisinden izlenecek bayt sayısı üst sınırıdır.

maxArrayBytes pozitif bir tamsayıya ayarlanırsa, izleme dosyasına yazılan bayt dizisindeki bayt sayısını sınırlar. *maxArrayBytes* yazdıktan sonra bayt dizisini keser. *maxArrayBytes* ayarı, sonuçtaki izleme kütüğünün büyüklüğünü azaltır ve izlemenin uygulamanın başarımı üzerindeki etkisini azaltır.

Bu "zelli§e iliŸkin 0 de§eri, izleme kt § ne bayt dizilerinin i‡eriklerinin g" nderilmediğini g " rntler.

Varsayılan değer, izleme dosyasına gönderilen bayt dizisindeki bayt sayısı sınırını kaldıran -1değeridir.

com.ibm.msg.client.commonservices.trace.limit = *maxTraceByte*
maxTraceBytes , bir izleme çıkış dosyasına yazılan byte sayısı üst sınırıdır.

maxTraceBytes , *traceCycles*ile çalışır. Yazılan izleme baytı sayısı sınıra yakınsa, dosya kapatılır ve yeni bir izleme çıkış dosyası başlatılır.

0 değeri, izleme çıkış kütüğünün uzunluğunun sıfır olduğu anlamına gelir. Varsayılan değer -1olup bu, bir izleme çıkış dosyasına yazılacak veri miktarının sınırsız olduğu anlamına gelir.

com.ibm.msg.client.commonservices.trace.count = *traceCycles*
traceCycles , geçiş için geçilecek izleme çıkış dosyalarının sayısıdır.

Yürürlükteki izleme çıkışı dosyası *maxTraceBytes*ile belirtilen sınıra ulaşırsa, dosya kapatılır. Sonraki izleme çıkışı, sıralı olarak sonraki izleme çıkışı kütüğüne yazılır. Her izleme çıkış dosyası, dosya adının sonuna eklenen sayısal bir sonekle ayırt edilir. Yürürlükteki ya da en son izleme çıkışı kütüğü mqjms.trc.0, sonraki izleme çıkışı kütüğü mqjms.trc.1. Daha eski izleme dosyaları, sınıra kadar aynı numaralandırma kalıbını izler.

traceCycles varsayılan değeri 1 'dir. *traceCycles* değeri 1 ise, yürürlükteki izleme çıkış dosyası büyüklük üst sınırına ulaştığında dosya kapatılır ve silinir. Aynı adı taşıyan yeni bir izleme çıkış dosyası başlatıldı. Bu nedenle, bir kerede tek bir izleme çıkış dosyası vardır.

com.ibm.msg.client.commonservices.trace.parameter = *traceParameters*
traceParameters , yöntem değiştirgelerinin ve dönüş değerlerinin izleme kapsamına alınıp alınmayacağını denetler.

traceParameters , varsayılan olarak TRUEdeğerine ayarlanır. *traceParameters* FALSEolarak ayarlanırsa, yalnızca yöntem imzaları izlenir.

com.ibm.msg.client.commonservices.trace.startup = *başlatma*
Kaynakların ayrıldığı IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging başlatma aşaması vardır. Ana izleme olanağı, kaynak ayırma aşamasında kullanıma hazırlandı.

startup TRUEolarak ayarlanırsa, başlatma izlemesi kullanılır. İzleme bilgileri hemen üretilir ve izleme olanağının kendisi de içinde olmak üzere tüm bileşenlerin kurulumunu içerir. Yapılandırma sorunlarını tanılamak için başlatma izleme bilgileri kullanılabilir. Başlatma izleme bilgileri her zaman `System.err`' e yazılır.

startup , varsayılan olarak FALSEdeğerine ayarlanır.

Kullanıma hazırlama tamamlanmadan önce *startup* işaretlenir. Bu nedenle, yalnızca komut satırında özelliği Java sistem özelliği olarak belirtin. Bunu IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yapılandırma dosyasında belirtmeyin.

com.ibm.msg.client.commonservices.trace.compress = *compressedTrace*
İzleme çıkışını sıkıştırmak için *compressedTrace* değerini TRUE olarak ayarlayın.

compressedTrace varsayılan değeri FALSEdeğeridir.

compressedTrace TRUEolarak ayarlanırsa, izleme çıkışı sıkıştırılır. Varsayılan izleme çıkış dosyası adı `.trz`uzantısına sahiptir. Sıkıştırma FALSEolarak ayarlanırsa, varsayılan değer, dosyanın sıkıştırılmadığını belirtmek için `.trc` uzantısına sahiptir. Ancak, *traceOutputName* dosyasında izleme çıkışına ilişkin dosya adı belirtildiyse, bu ad kullanılır; dosyaya sonek uygulanmaz.

Sıkıştırılmış izleme çıkışı sıkıştırılmamış değerinden küçük. Daha az G/Ç olduğundan, sıkıştırılmamış izlemeden daha hızlı yazılabilir. Sıkıştırılmış izleme, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging performansı üzerinde sıkıştırılmamış izlemeden daha az etkiye sahiptir.

maxTraceBytes ve *traceCycles* ayarlanırsa, birden çok düz dosya yerine birden çok sıkıştırılmış izleme dosyası yaratılır.

IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging denetimsiz bir şekilde sona ererse, sıkıştırılmış bir izleme dosyası geçerli olmayabilir. Bu nedenle, izleme sıkıştırması yalnızca IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging denetimli bir şekilde kapandığında kullanılmalıdır. Yalnızca incelenmekte olan sorunlar JVM ' nin beklenmedik bir şekilde durmasına neden olmazsa izleme sıkıştırmasını kullanın. `System.Halt()` ' in kapanmasına ya da olağandışı, denetimsiz JVM sonlandırmasına neden olabilecek sorunları tanılarken izleme sıkıştırmasını kullanmayın.

com.ibm.msg.client.commonservices.trace.level = *traceLevel*
traceLevel , izleme için bir süzgeç düzeyi belirtir. Tanımlanan izleme düzeyleri aşağıdaki gibidir:

- `TRACE_NONE`: `0`
- `TRACE_EXCEPTION`: `1`
- `TRACE_WARNING`: `3`
- `TRACE_INFO`: `6`
- `TRACE_ENTRYEXIT`: `8`
- `TRACE_DATA`: `9`
- `TRACE_ALL`: `Integer.MAX_VALUE`

Her izleme düzeyi tüm alt düzeyleri içerir. Örneğin, izleme düzeyi `TRACE_INFO`olarak ayarlanırsa, tanımlı düzeyi `TRACE_EXCEPTION`, `TRACE_WARNING`ya da `TRACE_INFO` olan herhangi bir izleme noktası izlemeye yazılır. Diğer tüm izleme noktaları dışlanır.

com.ibm.msg.client.commonservices.trace.standalone = *standaloneTrace*

standaloneTrace , IBM MQ JMS istemci izleme hizmetinin WebSphere Application Server ortamında kullanılıp kullanılmayacağını denetler.

standaloneTrace TRUEolarak ayarlanırsa, izleme yapılandırmasını belirlemek için IBM MQ JMS istemcisi izleme özellikleri kullanılır.

standaloneTrace , FALSEolarak ayarlanırsa ve IBM MQ JMS istemcisi bir WebSphere Application Server kapsayıcısında çalışıyorsa, WebSphere Application Server izleme hizmeti kullanılır. Oluşturulan izleme bilgileri, uygulama sunucusunun izleme ayarlarına bağlıdır.

standaloneTrace varsayılan değeri FALSEdeğeridir.

Günlüğe kaydetme kısmı
IBM MQ classes for JMS günlük olanağını yapılandırmak için Günlük Kaydı (Logging) özelliğini kullanın.

Günlüğe kaydetme kısmı aşağıdaki özellikleri içerir:

com.ibm.msg.client.commonservices.log.outputName = yol
IBM MQ classes for JMS günlük olanağı tarafından kullanılan günlük dosyasının adı. Varsayılan değer, IBM MQ classes for JMS ' in çalıştığı Java Runtime Environment için yürürlükteki çalışma dizinine yazılan `mqjms.log`değeridir.

Özellik aşağıdaki değerlerden birini alabilir:

- tek yol adı
- yol adlarının virgülle ayrılmış listesi (tüm veriler tüm dosyalara kaydedilir)

Her yol adı mutlak ya da göreli bir yol adı olabilir ya da:

"stderr" ya da "System.err"
Standart hata akışını temsil eder.

"stdout" ya da "System.out"
Standart çıkış akımını gösterir.

com.ibm.msg.client.commonservices.log.maxBytes
İleti verilerini günlüğe kaydetmek için herhangi bir çağrıdan günlüğe kaydedilen bayt sayısı üst sınırı.

Pozitif tamsayı
Veriler, günlük çağrısı başına bu bayt değerine kadar yazılır.

0
Veri yazılmaz.

-1
Sınırsız veri yazılır (varsayılan).

com.ibm.msg.client.commonservices.log.limit
Herhangi bir 1 günlük kütüğüne yazılan byte sayısı üst sınırı (varsayılan değer 262144 'tür).

Pozitif tamsayı
Veriler, günlük dosyası başına bu bayt değerine kadar yazılır.

0
Veri yazılmaz.

-1
Sınırsız veri yazılır.

com.ibm.msg.client.commonservices.log.count
Geçilecek günlük dosyalarının sayısı. Her kütük `com.ibm.msg.client.commonservices.trace.limit` kütüğüne ulaştığında, sonraki kütükte izleme başlar; varsayılan değer 3 'tür.

Pozitif tamsayı
Üzerinden geçilecek dosya sayısı.

0
Tek bir dosya.

Java SE Specifics kısmı
IBM MQ classes for JMS bir Java Standard Edition ortamında kullanıldığında kullanılan özellikleri yapılandırmak için Java SE Specifics kısmına bakın.

com.ibm.msg.client.commonservices.j2se.produceJavaCore = TRUE|FALSE
IBM MQ classes for JMS bir FDC dosyası oluşturduktan hemen sonra JavaCore dosyasının yazılıp yazılmayacağını belirler. Bu seçenek TRUE olarak ayarlanırsa, IBM MQ classes for JMS ' nin çalıştığı Java Runtime Environment 'ın çalışma dizininde bir JavaCore dosyası üretilir.

DOĞRU
Java Runtime Environment 'ın bunu yapabilme yeteneğine bağlı olarak JavaCore oluşturun.

YANLIŞ
JavaCore oluşturmayın; bu varsayılan değerdir.

IBM MQ Özellikler kısmı
IBM MQ classes for JMS Etkileşimini IBM MQetkileyen özellikleri ayarlamak için IBM MQ Özellikler kısmına bakın.

com.ibm.msg.client.wmq.compat.base.internal.MQQueue.smallMsgsBufferReductionThreshold
IBM MQ classes for JMS kullanan bir uygulama, IBM MQ ileti alışverişi sağlayıcısı geçiş kipini kullanarak bir IBM MQ kuyruk yöneticisine bağlanırken IBM MQ classes for JMS , ileti aldığında varsayılan arabellek boyutunu 4 KB kullanır. Uygulamanın almaya çalıştığı ileti 4 KB ' den büyükse, IBM MQ classes for JMS arabelleği iletiyi alacak büyüklükte olacak şekilde yeniden boyutlandırır. Daha sonra, daha büyük arabellek boyutu, sonraki iletiler alındığında kullanılır.

Bu özellik, arabellek büyüklüğünün ne zaman 4 KB ' ye indirileceğini denetler. Varsayılan olarak, arabellek büyüklüğünden daha küçük on ardışık ileti alındığında, arabellek büyüklüğü 4 KB ' ye düşürülür. Bir ileti her alındığında arabellek büyüklüğünü 4 KB ' ye geri döndürmek için özelliği 0 değerine ayarlayın.

0

Arabellek her zaman varsayılan büyüklüğe sıfırlanır.

10

Bu varsayılan değerdir. Arabellek, onuncu mesajdan sonra yeniden boyutlandırılacak.

com.ibm.msg.client.wmq.receiveConversionCCSID
IBM MQ classes for JMS kullanan bir uygulama IBM MQ ileti alışverişi sağlayıcısı normal kipini kullanarak bir IBM MQ kuyruk yöneticisine bağlanırken `receiveConversionCCSID` özelliği, kuyruk yöneticisinden ileti almak için kullanılan MQMD yapısındaki varsayılan CCSID değerini geçersiz kılacak şekilde ayarlanabilir. Varsayılan olarak MQMD, 1208 olarak ayarlanmış bir CCSID alanı içerir, ancak kuyruk yöneticisi iletileri bu kod sayfasına dönüştüremezse bu değiştirilebilir.

Geçerli değerler, herhangi bir geçerli CCSID numarası ya da aşağıdaki değerlerden biridir:

-1
Altyapının varsayılan değerini kullanın.

1208
Bu varsayılan değerdir.

İstemci kipi özel kısmı
IBM MQ classes for JMS CLIENT iletimini kullanan bir kuyruk yöneticisine bağlandığında kullanılacak özellikleri belirtmek için Client-mode özel özelliklerini kullanın.

com.ibm.mq.polling.RemoteRequestEntry
IBM MQ classes for JMS ' in bir kuyruk yöneticisinden yanıt beklerken bozuk bağlantıları denetlemek için kullandığı yoklama aralığını belirtir.

Pozitif tamsayı
Denetlemeden önce beklenecek milisaniye sayısı. Varsayılan değer 10000 ya da 10 saniyedir. Alt sınır değeri 3000 'dir ve alt değerler bu alt sınır değeriyle aynı şekilde işlenir.

JMS istemci davranışını yapılandırmak için kullanılan özellikler
JMS istemcisinin davranışını yapılandırmak için bu özellikleri kullanın.

com.ibm.mq.jms.SupportMQExtensions TRUE|FALSE
JMS 2.0 belirtimi, belirli davranışların çalışma şeklini değiştirir. IBM MQ 8.0 ,
değiştirilen bu davranışları önceki uygulamalara geri döndürmek için *TRUE*olarak ayarlanabilen `com.ibm.mq.jms.SupportMQExtensions`özelliğini içerir. Değiştirilen davranışların geri çevrilmesi, JMS 2.0 uygulamaları için ve ayrıca JMS 1.1 API 'sini kullanan, ancak IBM MQ 8.0 IBM MQ classes for JMSürününe karşı çalışan bazı uygulamalar için gerekli olabilir.

DOĞRU
SupportMQExtensions ayarı TRUEolarak ayarlanarak aşağıdaki üç işlev alanı geri çevrilir:

İleti önceliği
İletilere bir öncelik atanabilir: *0* - *9*. JMS 2.0' den önce iletiler, kuyruğun varsayılan önceliğinin kullanıldığını gösteren *-1*değerini de kullanabilir. JMS 2.0 , *-1* ileti önceliğinin ayarlanmasına izin vermez. `SupportMQExtensions` ' un açılması, *-1* değerinin kullanılmasını sağlar.

İstemci Tanıtıcısı
JMS 2.0 belirtimi, bağlantı kurulurken boş olmayan istemci tanıtıcılarının benzersiz olup olmadıklarının denetlenmesini gerektirir. `SupportMQExtensions`' un açılması, bu gereksinimin göz ardı edileceği ve bir istemci tanıtıcısının yeniden kullanılabileceği anlamına gelir.

`NoLocal`
JMS 2.0 belirtimi, bu değişmez açıldığında bir tüketicinin aynı istemci tanıtıcısı tarafından yayınlanan iletileri alamamasını gerektirir. JMS 2.0öncesinde bu öznitelik, kendi bağlantısıyla yayınlanan iletileri almasını önlemek için bir abonede ayarlanmıştı. `SupportMQExtensions` ' un açılması bu davranışı önceki uygulamasına geri çevirir.

YANLIŞ
Davranış değişiklikleri korunur.

`com.ibm.msg.client.jms.ByteStreamReadOnlyAfterSend= TRUE|FALSE`
IBM MQ 8.0.0 Fix Pack 2' den bir uygulama Bytes ya da Stream iletisi gönderdikten sonra IBM MQ classes for JMS , yalnızca okunur ya da salt okunur olarak gönderilen iletinin durumunu ayarlayabilir.

DOĞRU
Nesneler gönderildikten sonra salt okunur olarak ayarlanır. Bu değerin ayarlanması, JMS 2.0 belirtimiyle uyumluluğu korur

YANLIŞ
Nesneler yalnızca gönderildikten sonra yazmak üzere ayarlanır. Bu varsayılan değerdir.

İlgili kavramlar
“SupportMQExtensions özelliği” sayfa 315
JMS 2.0 belirtimi, belirli davranışların çalışma şeklini değiştirdi. IBM MQ 8.0 ve daha sonra, değiştirilen bu davranışları önceki uygulamalara geri döndürmek için TRUE olarak ayarlanabilen **`com.ibm.mq.jms.SupportMQExtensions`**özelliğini içerir.

z/OS üzerinde IBM MQ classes for JMS için STEPLIB yapılandırması
z/OSüzerinde, yürütme sırasında kullanılan STEPLIB, IBM MQ SCSQAUTH ve SCSQANLE kitaplıklarını içermelidir. Bu kitaplıkları başlatma JCL dosyasında ya da `.profile` dosyasını kullanarak belirtin.

z/OS UNIX System Services'den, aşağıdaki kod parçacığında gösterildiği gibi `.profile` içindeki bir satırı kullanarak bunları ekleyebilirsiniz; `thlqual` yerine, IBM MQ' u kurarken seçtiğiniz üst düzey veri kümesi niteleyicisini kullanabilirsiniz:

```
export STEPLIB=thlqual.SCSQAUTH:thlqual.SCSQANLE:$STEPLIB
```

Diğer ortamlarda genellikle, STEPLIB birleşiminde SCSQAUTH ve SCSQANLE 'yi içerecek şekilde başlatma JCL' sini düzenlemeniz gerekir:

```
STEPLIB DD DSN=thlqual.SCSQAUTH,DISP=SHR
        DD DSN=thlqual.SCSQANLE,DISP=SHR
```

IBM MQ classes for JMS ve yazılım yönetimi araçları
Apache Maven gibi yazılım yönetimi araçları, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingile birlikte kullanılabilir.

Birçok büyük geliştirme kuruluşu, üçüncü kişi kitaplıklarının havuzlarını merkezi olarak yönetmek için bu araçları kullanır.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , bir dizi JAR dosyasından oluşur. Java dil uygulamalarını bu API ' yi kullanarak geliştirirken, uygulamanın geliştirilmekte olduğu makinede bir IBM MQ Server, IBM MQ Client ya da IBM MQ Client SupportPac kuruluşu gerekir.

Böyle bir aracı kullanmak ve IBM MQ classes for JMS ' u oluşturan JAR dosyalarını merkezi olarak yönetilen bir havuza eklemek istiyorsanız, aşağıdaki noktalara dikkat edilmelidir:

- Bir havuz ya da kapsayıcı yalnızca kuruluşunuz içindeki geliştiriciler tarafından kullanılabilir kılınmalıdır. Kuruluş dışında dağıtıma izin verilmez.
- Havuzun, tek bir IBM MQ yayın düzeyindeki ya da düzeltme paketindeki JAR dosyalarının eksiksiz ve tutarlı bir kümesini içermesi gerekir.
- Havuzu, IBM Destek tarafından sağlanan herhangi bir bakımla güncellemekten siz sorumlu olursunuz.

Aşağıdaki JAR dosyalarının havuza kurulması gerekir:

- JMS 2.0 IBM MQ classes for JMSkullanıyorsanız `com.ibm.mq.allclient.jar` ve `jms.jar` gereklidir.
- JM 3.0 IBM MQ classes for Jakarta Messagingkullanıyorsanız, `com.ibm.mq.jakarta.client.jar` ve `jakarta.jms-api.jar` gereklidir.
- IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging kullanıyorsanız ve bir dosya sistemi JNDI bağlamında saklanan JMS denetimli nesnelere erişiyorsanız `fscontext.jar` gereklidir.
- IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging kullanıyorsanız ve bir dosya sistemi JNDI bağlamında saklanan JMS denetimli nesnelere erişiyorsanız, `providerutil.jar` gereklidir.
- IBM dışı JRE ' ler için destek için Bouncy Castle güvenlik sağlayıcısı ve CMS destek JAR dosyaları gereklidir. Daha fazla bilgi için bkz. Support for non-IBM JRE.

IBM MQ classes for JMS uygulamalarının Java security manager altında çalıştırılması

IBM MQ classes for JMS , Java security manager etkinken çalışabilir. Java security manager etkin durumdayken uygulamaları başarıyla çalıştırmak için Java Virtual Machine (JVM) ürününüzü uygun bir ilke yapılandırma dosyasıyla yapılandırmanız gerekir.

Uygun bir ilke tanımlama dosyası yaratmanın en basit yolu, Java runtime environment (JRE) ile sağlanan ilke yapılandırma dosyasını değiştirmektir. Çoğu sistemde, bu dosya JRE dizininizle göreli olarak `lib/security/java.policy` dizininde bulunur. Tercih ettiğiniz düzenleyiciyi kullanarak ya da JRE ile verilen ilke aracı programını kullanarak ilke yapılandırma dosyasını düzenleyebilirsiniz.

Örnek ilke yapılandırma dosyası

Aşağıda, IBM MQ classes for JMS ' in varsayılan güvenlik yöneticisi altında başarıyla çalışmasına olanak sağlayan bir ilke yapılandırma dosyası örneği verilmiştir. Bu dosyanın, belirli dosyaların ve dizinlerin yerlerini belirtmek için uyarlanması gerekir: *MQ_INSTALLATION_PATH* , IBM MQ ' un kurulu olduğu üst düzey dizini, *MQ_DATA_DIRECTORY* MQ veri dizininin konumunu ve *QM_NAME* , erişimin yapılandırıldığı kuyruk yöneticisinin adıdır.

```
grant codeBase "file:MQ_INSTALLATION_PATH/java/lib/*" {
  //We need access to these properties, mainly for tracing
  permission java.util.PropertyPermission "user.name","read";
  permission java.util.PropertyPermission "os.name","read";
  permission java.util.PropertyPermission "user.dir","read";
  permission java.util.PropertyPermission "line.separator","read";
  permission java.util.PropertyPermission "path.separator","read";
  permission java.util.PropertyPermission "file.separator","read";
  permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.log.*","read";
  permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.trace.*","read";
  permission java.util.PropertyPermission "Diagnostics.Java.Errors.Destination.Filename","read";
  permission java.util.PropertyPermission "com.ibm.mq.commonservices","read";
  permission java.util.PropertyPermission "com.ibm.mq.cfg.*","read";

  //Tracing - we need the ability to control java.util.logging
  permission java.util.logging.LoggingPermission "control";
```

```
  // And access to create the trace file and read the log file - assumed to be in the current
directory
  permission java.io.FilePermission "*","read,write";

  // We'd like to set up an mBean to control trace
  permission javax.management.MBeanServerPermission "createMBeanServer";
  permission javax.management.MBeanPermission "*","*";

  // We need to be able to read manifests etc from the jar files in the installation directory
  permission java.io.FilePermission "MQ_INSTALLATION_PATH/java/lib/-","read";

  //Required if mqclient.ini/mqs.ini configuration files are used
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/mqclient.ini","read";
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/mqs.ini","read";

  //For the client transport type.
  permission java.net.SocketPermission "*","connect,resolve";

  //For the bindings transport type.
  permission java.lang.RuntimePermission "loadLibrary.*";

  //For applications that use CCDT tables (access to the CCDT AMQCLCHL.TAB)
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/qmgrs/QM_NAME/@ipcc/AMQCLCHL.TAB","read";

  //For applications that use User Exits
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/exits/*","read";
  permission java.io.FilePermission "MQ_DATA_DIRECTORY/exits64/*","read";
  permission java.lang.RuntimePermission "createClassLoader";

  //Required for the z/OS platform
  permission java.util.PropertyPermission "com.ibm.vm.bitmode","read";

  // Used by the internal ConnectionFactory implementation
  permission java.lang.reflect.ReflectPermission "suppressAccessChecks";

  // Used for controlled class loading
  permission java.lang.RuntimePermission "setContextClassLoader";

  // Used to default the Application name in Client mode connections
  permission java.util.PropertyPermission "sun.java.command","read";

  // Used by the IBM JSSE classes
  permission java.util.PropertyPermission "com.ibm.crypto.provider.AESNITrace","read";

  //Required to determine if an IBM Java Runtime is running in FIPS mode,
  //and to modify the property values status as required.
  permission java.util.PropertyPermission "com.ibm.jsse2.usefipsprovider","read,write";
  permission java.util.PropertyPermission "com.ibm.jsse2.JSSEFIPS","read,write";
  //Required if an IBM FIPS provider is to be used for SSL communication.
  permission java.security.SecurityPermission "insertProvider.IBMJCEFIPS";

  // Required for non-IBM Java Runtimes that establish secure client
  // transport mode connections using mutual TLS authentication
  permission java.util.PropertyPermission "javax.net.ssl.keyStore","read";
  permission java.util.PropertyPermission "javax.net.ssl.keyStorePassword","read";
};
```

Örnekte, `grant` deyimi IBM MQ classes for JMSiçin gereken izinleri içerir. İlke yapılanış kütüğünüzdeki bu izin verme deyimlerini kullanmak için, yol adlarını IBM MQ classes for JMS ' i nereye kurduğunuza ve uygulamalarınızı nereye sakladığınıza bağlı olarak değiştirmeniz gerekebilir.

IBM MQ classes for JMSile sağlanan örnek uygulamalar ve bunları çalıştıracak komut dosyaları, güvenlik yöneticisini etkinleştirmez.

Önemli:

IBM MQ classes for JMS izleme olanağı, ek sistem özelliklerini sorgulama ve diğer dosya sistemi işlemlerini gerçekleştirirken ek izinler gerektirir.

İzlemenin etkinleştirildiği bir güvenlik yöneticisi altında çalışmaya uygun bir şablon güvenlik ilkesi dosyası, IBM MQ kuruluşunun `samples/wmqjava` dizininde `example.security.policy`olarak sağlanır.

IBM MQ classes for JMS uygulamaları için kuruluş sonrası kuruluş

Bu konuda, bir kuyruk yöneticisinin kaynaklarına erişmek için IBM MQ classes for JMS uygulamalarının hangi yetkilere gereksinim duyması gerektiği açıklanmaktadır. Ayrıca, bağlantı kiplerini tanıtır ve

uygulamaların istemci kipinde bağlanabilmeleri için bir kuyruk yöneticisinin nasıl yapılandırılacağını açıklar.

IBM MQ readme (benioku) dosyasını denetmeyi unutmayın. Bu konuda, bu konudaki bilgilerin yerine geçen bilgileri içerebilir.

Ayrıcalıklı olmayan kullanıcılar için yetki gerektiren JMS tarafından kullanılan nesneler
Ayrıcalıklı olmayan kullanıcıların, JMStarafından kullanılan kuyruklara erişmek için yetkilendirilmesi gerekir. Her JMS uygulamasının, çalıştığı kuyruk yöneticisi için yetki edinmesi gerekir.

IBM MQiçinde erişim denetimine ilişkin ayrıntılar için Güvenliği ayarlamabaşlıklı konuya bakın.

IBM MQ classes for JMS uygulamaları için kuyruk yöneticisine `bağlanma` ve `inq` yetkisi gerekir. **`setmqaut`** denetim komutunu kullanarak uygun yetkileri ayarlayabilirsiniz, örneğin:

```
setmqaut -m QM1 -t qmgr -g jmsappsgroup +connect +inq
```

Noktadan noktaya iletişim etki alanı için aşağıdaki yetkiler gereklidir:

- MessageProducer nesneleri tarafından kullanılan kuyruklar için `koyma` yetkisi gerekir.
- MessageConsumer ve QueueBrowser nesneleri tarafından kullanılan kuyruklar için `get`, `inq`ve `browse` yetkileri gerekir.
- QueueSession.createTemporaryQueue () yönteminin, QueueConnectionFactory nesnesinin TEMPMODEL özelliğiyle belirtilen model kuyruğuna erişmesi gerekir. Varsayılan olarak bu model kuyruğu SYSTEM.TEMP.MODEL.QUEUE.

Bu kuyruklardan herhangi biri diğer ad kuyruğuysa, hedef kuyrukları için sorma yetkisi gerekir. Hedef kuyruk bir küme kuyruğuysa, göz atma yetkisi de gerekir.

Yayınlama/abone olma etki alanı için, IBM MQ classes for JMS IBM MQ ileti alışverişi sağlayıcısı geçiş kipinde bir IBM MQ kuyruk yöneticisine bağlanıyorsa aşağıdaki kuyruklar kullanılır:

- 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

IBM MQ ileti alışverişi sağlayıcısı geçiş kipiyle ilgili daha fazla bilgi için JMS **PROVIDERVERSION** özelliğini yapılandırma başlıklı konuya bakın.

Buna ek olarak, IBM MQ classes for JMS bu kipte bir kuyruk yöneticisine bağlanıyorsa, iletileri yayınlayan herhangi bir uygulamanın TopicConnectionüreticisi ya da konu nesnesi tarafından belirtilen akış kuyruğuna erişmesi gerekir. Varsayılan olarak, bu kuyruk SYSTEM.BROKER.DEFAULT.STREAM.

ConnectionConsumer, IBM MQ Resource Adapter ya da WebSphere Application Server IBM MQ ileti sistemi sağlayıcısını kullanıyorsanız, ek yetki gerekebilir.

ConnectionConsumer tarafından okunacak kuyrukların `get`, `inq`ve `browse` yetkileri olmalıdır. Sistem gitmeyen ileti kuyruğu ve ConnectionConsumer tarafından kullanılan herhangi bir arka plan kuyruğu ya da rapor kuyruğu `koyma` ve `geçiş` yetkililerine sahip olmalıdır.

Bir uygulama yayınlama/abone olma ileti alışverişi gerçekleştirmek için IBM MQ ileti alışverişi sağlayıcısı normal kipini kullandığında, uygulama kuyruk yöneticisi tarafından sağlanan tümleşik yayınlama/abone olma işlevini kullanır. Kullanılan konuların ve kuyrukların güvenliğinin sağlanmasına ilişkin bilgi için Güvenliğin yayınlanması/abone olunması başlıklı konuya bakın.

IBM MQ classes for JMS için bağlantı kipleri
Bir IBM MQ classes for JMS uygulaması, istemci ya da bağ tanımlama kipinde bir kuyruk yöneticisine bağlanabilir. İstemci kipinde IBM MQ classes for JMS , TCP/IP üzerinden kuyruk yöneticisine bağlanır. Bağ tanımlama kipinde IBM MQ classes for JMS , Java Native Interface (JNI) olanağını kullanarak kuyruk yöneticisine doğrudan bağlanır.

z/OS üzerinde WebSphere Application Server işletim sistemlerinde çalışan bir uygulama, bağ tanımlarında ya da istemci kipinde bir kuyruk yöneticisine bağlanabilir, ancak z/OS üzerinde başka bir ortamda çalışan bir uygulama yalnızca bağ tanımlama kipinde bir kuyruk yöneticisine bağlanabilir. Başka bir altyapıda çalışan bir uygulama, bağ tanımlarında ya da istemci kipinde bir kuyruk yöneticisine bağlanabilir.

Yürürlükteki ya da daha önceki desteklenen IBM MQ classes for JMS sürümünü yürürlükteki bir kuyruk yöneticisiyle kullanabilir ve yürürlükteki IBM MQ classes for JMSsürümüyle birlikte kuyruk yöneticisinin yürürlükteki ya da daha önceki bir sürümünü kullanabilirsiniz. Farklı sürümleri karıştırırsanız, işlev önceki sürümün düzeyiyle sınırlıdır.

Aşağıdaki bölümlerde, bağlantı kiplerinin her biri daha ayrıntılı olarak açıklanmaktadır.

İstemci kipi

İstemci kipinde bir kuyruk yöneticisine bağlanmak için IBM MQ classes for JMS uygulaması, kuyruk yöneticisinin çalıştığı sistemde ya da farklı bir sistemde çalışabilir. Her bir durumda IBM MQ classes for JMS , TCP/IP üzerinden kuyruk yöneticisine bağlanır.

Bağ tanımları kipi

Bağ tanımlama kipinde bir kuyruk yöneticisine bağlanmak için, IBM MQ classes for JMS uygulamasının kuyruk yöneticisinin çalıştığı sistemde çalışması gerekir.

IBM MQ classes for JMS , Java Native Interface (JNI) olanağını kullanarak kuyruk yöneticisine doğrudan bağlanır. Bağ tanımları iletimini kullanmak için IBM MQ classes for JMS , IBM MQ Java Yerel Arabirim kitaplıklarına erişimi olan bir ortamda çalıştırılmalıdır; daha fazla bilgi için bkz. “Java Native Interface (JNI) kitaplıklarının yapılandırılması” sayfa 92 .

IBM MQ classes for JMS , *ConnectOption*için aşağıdaki değerleri destekler:

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARD_BINDING
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING
- MQCNO_RESTRICT_CONN_TAG_QSG
- MQCNO_RESTRICT_CONN_TAG_Q_MGR

IBM MQ classes for JMStarafından kullanılan bağlantı seçeneklerini değiştirmek için CONNOPTConnection Factory özelliğini değiştirin.

Bağlantı seçenekleriyle ilgili daha fazla bilgi için bkz. “MQCONNX çağrısı kullanılarak bir kuyruk yöneticisiyle bağlantı kurulması” sayfa 708

Bağ tanımları iletimini kullanmak için, kullanılmakta olan Java Runtime Environment IBM MQ classes for JMS ' ın bağlandığı kuyruk yöneticisinin CCSID 'sini (Coded Character Set Identifier; Kodlanmış Karakter Takımı Tanıtıcısı) desteklemelidir.

Java Runtime Environment tarafından desteklenen CCSID ' lerin saptanmasına ilişkin ayrıntılar için IBM MQ Java için IBM MQ V7 sınıfları ya da JMS için IBM MQ V7 sınıfları kullanılırken Probe Tanıtıcısı 21 olan FDC üretilir.kısmına bakın.

IBM MQ classes for JMS uygulamalarının istemci kipinde bağlanabilmesi için kuyruk yöneticinizi yapılandırma
IBM MQ classes for JMS uygulamalarının istemci kipinde bağlanabilmesi için kuyruk yöneticinizi yapılandırmak üzere bir sunucu bağlantısı kanal tanımlaması yaratmanız ve bir dinleyici başlatmanız gerekir.

Sunucu bağlantısı kanal tanımı yaratılması

Tüm altyapılarda, bir sunucu bağlantısı kanal tanımlaması yaratmak için MQSC DEFINE CHANNEL komutunu kullanabilirsiniz. Aşağıdaki örneğe bakın:

```
DEFINE CHANNEL(JAVA.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
```

IBM i IBM iüzerinde, CRTMQMCHL Denetim dili (CL) komutunu aşağıdaki örnekte olduğu gibi kullanabilirsiniz:

```
CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN)
TRPTYPE(*TCP)
MQMNAME(QMGRNAME)
```

Bu komutta, *QMGRNAME* kuyruk yöneticinizin adıdır.

Linux Windows Linux ve Windowssistemlerinde, IBM MQ Explorerkomutunu kullanarak bir sunucu bağlantısı kanal tanımı da yaratabilirsiniz.

z/OS z/OS işletim ve denetim panolarını kullanarak bir sunucu bağlantısı kanal tanımı yaratabilirsiniz.

Kanalın adı (JAVA.CHANNEL (Kanal), uygulamanızın kuyruk yöneticisine bağlanmak için kullandığı bağlantı üreticisinin CHANNEL özelliği tarafından belirtilen kanal adıyla aynı olmalıdır. CHANNEL özelliğinin varsayılan değeri SYSTEM.DEF.SVRCONN.

Dinleyici başlatılması

Kuyruk yöneticiniz başlatılmadıysa, kuyruk yöneticiniz için bir dinleyici başlatmanız gerekir.

Multi Çoklu platformlarüzerinde, aşağıdaki örnekte gösterildiği gibi, MQSC komutunu DEFINE LISTENER kullanarak bir dinleyici yarattıktan sonra bir dinleyici başlatmak için MQSC START LISTENER komutunu kullanabilirsiniz:

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414)
START LISTENER(LISTENER.TCP)
```

z/OS z/OS' da, aşağıdaki örnekte olduğu gibi yalnızca START LISTENER komutunu kullanırsınız, ancak bir dinleyiciyi başlatmadan önce kanal başlatıcı adres alanının başlatılması gerektiğini unutmayın:

```
START LISTENER TRPTYPE(TCP) PORT(1414)
```

IBM i IBM iüzerinde, aşağıdaki örnekte olduğu gibi, bir dinleyiciyi başlatmak için STRMQMLSR CL komutunu da kullanabilirsiniz:

```
STRMQMLSR PORT(1414) MQMNAME(QMGRNAME)
```

Bu komutta, *QMGRNAME* kuyruk yöneticinizin adıdır.

ALW AIX, Linux, and Windowssistemlerinde, aşağıdaki örnekteki gibi bir dinleyici başlatmak için **runmqlsr** denetim komutunu da kullanabilirsiniz:

```
runmqlsr -t tcp -p 1414 -m QMgrName
```

Bu komutta, *QMgrName* kuyruk yöneticinizin adıdır.

Linux Windows Linux ve Windowssistemlerinde, IBM MQ Explorerkomutunu kullanarak bir dinleyici de başlatabilirsiniz.

z/OS z/OSüzerinde, bir dinleyici başlatmak için işlemleri ve denetim panolarını da kullanabilirsiniz.

Dinleyicinin dinlediği kapının numarası, uygulamanızın kuyruk yöneticisine bağlanmak için kullandığı bağlantı üreticisinin PORT özelliği tarafından belirtilen kapı numarasıyla aynı olmalıdır. PORT özelliğinin varsayılan değeri 1414 'tür.

IBM MQ classes for JMS için noktadan noktaya IVT

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingile bir noktadan noktaya kuruluş doğrulama sınaması (IVT) programı sağlanır. Program, bağ tanımlarında ya da istemci kipinde bir kuyruk yöneticisine bağlanır ve kuyruğa SYSTEM.DEFAULT.LOCAL.QUEUEve daha sonra, iletiyi kuyruktan alır. Program, yürütme sırasında dinamik olarak gerektirdiği tüm nesneleri yaratabilir ve yapılandırabilir ya da bir dizin hizmetinden yönetilen nesneleri almak için JNDI ' yı kullanabilir.

Sınama bağımsız olduğundan ve bir dizin hizmetinin kullanılmasını gerektirmediğinden, önce JNDI kullanmadan kuruluş doğrulama sınamasını çalıştırın. Denetlenen nesnelere ilişkin açıklamalar için Yönetim aracını kullanarak JMS nesnelerini yapılandırmabaşlıklı konuya bakın.

JNDI kullanmadan noktadan noktaya kuruluş doğrulama sınaması

Bu testte, IVT programı yürütme sırasında dinamik olarak gerektirdiği tüm nesneleri yaratır ve yapılandırır ve JNDI kullanmaz.

Multi Çoklu platformlarda, IVT programını çalıştırmak için bir komut dosyası sağlanır. Komut dosyası, AIX and Linux sistemlerinde **IVTRun** ve Windowssistemlerinde **IVTRun.bat** olarak adlandırılır. Komut dosyası, IBM MQ classes for JMS kuruluş dizininin `bin` altdizininde bulunur. Sınıf yolu (classpath) `com.ibm.mqjms.jar`içermelidir.

Sınamayı bağ tanımlama kipinde çalıştırmak için şu komutu girin:

```
IVTRun -nojndi [-m qmgr ] [-v providerVersion ] [-t]
```

Testi istemci kipinde çalıştırmak için önce kuyruk yöneticisini “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023içinde açıklandığı gibi ayarlayın. Kullanılacak kanalın varsayılan değeri **SYSTEM.DEF.SVRCONN** ve kullanılacak kuyruğun **SYSTEM.DEFAULT.LOCAL.QUEUE**olduğunu unutmayın ve aşağıdaki komutu girin:

```
IVTRun -nojndi -client -m qmgr -host hostname [-port port ] [-channel channel ]
[-v providerVersion ] [-ccsid ccsid ] [-t]
```

z/OS z/OS sistemlerinde eşdeğer bir komut dosyası sağlanmaz. Bunun yerine, Java sınıfını doğrudan çağırarak IVT ' yi bağ tanımlama kipinde çalıştırıyorsunuz. z/OS' da, IVT programının işlevsel olarak özdeş iki örneği arasından seçim yapabilirsiniz:

- `com.ibm.mq.jms.MQJMSIVT`, IBM MQ classes for JMS (JMS 2.0) ile kullanılabilir. Bu programı kullanmak için sınıf yolu `com.ibm.mqjms.jar` ya da `com.ibm.mq.allclient.jar`içermelidir.

- `com.ibm.mq.jakarta.jms.MQJMSIVT`, IBM MQ classes for Jakarta Messaging (Jakarta Messaging 3.0) ile kullanılabilir. Bu programı kullanmak için sınıf yolu `com.ibm.mq.jakarta.client.jar`içermelidir.

Sınamayı z/OSüzerinde bağ tanımlama kipinde çalıştırmak için şu komutu girin:

```
java com.ibm.mq.jms.MQJMSIVT -nojndi [-m qmgr ] [-v providerVersion ] [-t]
```

Komutlara ilişkin değiştirgeler aşağıdaki anlamlara sahiptir:

-m *qmgr*
IVT programının bağlandığı kuyruk yöneticisinin adı. Sınamayı bağ tanımlama kipinde çalıştırır ve bu değiştirgeyi atlarsanız, IVT programı varsayılan kuyruk yöneticisine bağlanır.

-host *anasistemadı*
Kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi.

-port *kapı*
Kuyruk yöneticisinin dinleyicisinin dinlediği kapının numarası. Varsayılan değer 1414değeridir.

-channel *kanal*
IVT programının kuyruk yöneticisine bağlanmak için kullandığı MQI kanalının adı. Varsayılan değer SYSTEM.DEF.SVRCONNdeğeridir.

-v *providerVersion*
IVT programının bağlanmayı beklediği kuyruk yöneticisinin yayın düzeyi.

Bu değiştirge, bir MQQueueConnectionFactory nesnesinin PROVIDERVERSION özelliğini ayarlamak için kullanılır ve geçerli değerler PROVIDERVERSION özelliğininkiyle aynıdır. Bu nedenle, geçerli değerleri de içinde olmak üzere, bu parametreyle ilgili daha fazla bilgi için bkz. JMS: changes to PROVIDERVERSION property ve Properties of IBM MQ classes for JMS objectsiçindeki PROVIDERVERSION özelliğinin açıklaması.

Varsayılan değer `unspecified`değeridir.

-ccsid *ccsid*
Bağlantı tarafından kullanılacak kodlanmış karakter takımının ya da kod sayfasının tanıtıcısı (CCSID). Varsayılan değer 819değeridir.

-t
İzleme etkinleştirildi. Varsayılan olarak izleme devre dışıdır.

Başarılı bir sınama, aşağıdaki örnek çıkışa benzer bir çıkış üretir:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2024. All
Rights Reserved.
WebSphere MQ classes for Java(tm) Message Service 7.0
Installation Verification Test

Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message
JMSMessage class: jms_text
JMSType:     null
JMSDeliveryMode: 2
JMSExpiration:  0
JMSPriority:   4
JMSMessageID:   ID:414d5120514d5f6d627720202020202001edb14620005e03
JMSTimestamp:   1187170264000
JMSCorrelationID: null
JMSDestination:  queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
```

```
JMSReplyTo:    null
JMSRedelivered:  false
JMSXUserID: mwhite
JMS_IBM_Encoding: 273
JMS_IBM_PutApplType: 28
JMSXAppID: IBM MQ Client for Java
JMSXDeliveryCount: 1
JMS_IBM_PutDate: 20070815
JMS_IBM_PutTime: 09310400
JMS_IBM_Format: MQSTR
JMS_IBM_MsgType: 8
A simple text message from the MQJMSIVT
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
```

JNDI kullanan noktadan noktaya kuruluş doğrulama sınaması

Bu testte, IVT programı bir dizin hizmetinden yönetilen nesneleri almak için JNDI ' yı kullanır.

Sınamayı çalıştırmadan önce, LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) sunucusuna ya da yerel dosya sistemine dayalı bir dizin hizmeti yapılandırmanız gerekir. IBM MQ JMS yönetim aracını, yönetilen nesneleri saklamak için dizin hizmetini kullanabilecek şekilde yapılandırmanız da gerekir. Bu önkoşullar hakkında daha fazla bilgi için bkz. “IBM MQ classes for JMS için önkoşullar” sayfa 84. IBM MQ JMS yönetim aracının nasıl yapılandırılacağı hakkında bilgi için JMS yönetim aracının yapılandırılmasıbaşlıklı konuya bakın.

IVT programı, dizin hizmetinden bir MQQueueConnectionFactory nesnesi ve bir MQQueue nesnesini almak için JNDI ' yı kullanabilmelidir. Bu yönetilen nesneleri sizin için yaratmak üzere bir komut dosyası sağlanır. Komut dosyası, AIX and Linux sistemlerinde IVTSetup ve Windowsüzerinde IVTSetup.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin `bin` altdizininde bulunur. Komut dosyasını çalıştırmak için şu komutu girin:

```
IVTSetup
```

Komut dosyası, yönetilen nesneleri yaratmak için IBM MQ JMS yönetim aracını çağırır.

MQQueueConnectionFactory nesnesi için ivtQCF adı verilir ve tüm özellikleri için varsayılan değerlerle yaratılır; bu, IVT programının bağ tanımlama kipinde çalıştığı ve varsayılan kuyruk yöneticisine bağlandığı anlamına gelir. IVT programının istemci kipinde çalışmasını ya da varsayılan kuyruk yöneticisinden başka bir kuyruk yöneticisine bağlanmasını istiyorsanız, MQQueueConnectionFactory nesnesinin uygun özelliklerini değiştirmek için IBM MQ JMS denetim aracını ya da IBM MQ Explorer kullanmanız gerekir. IBM MQ Explorer JMS yönetim aracının nasıl kullanılacağına ilişkin bilgi için Yönetim aracını kullanarak JMS nesnelerini yapılandırmabaşlıklı konuya bakın. IBM MQ Explorer'in nasıl kullanılacağına ilişkin bilgi için bkz. IBM MQ Explorer ' e Giriş ya da IBM MQ Explorerile birlikte sağlanan yardım.

MQQueue nesnesi için ivtQ adıyla bağ tanımlandı ve SYSTEM.DEFAULT.LOCAL.QUEUE.

Denetlenen nesneleri yarattığınızda, IVT programını çalıştırabilirsiniz. Sınamayı JNDI kullanarak çalıştırmak için şu komutu girin:

```
IVTRun -url "providerURL" [-icf initCtxFact ] [-t]
```

Komuttaki değiştirgeler aşağıdaki anlamlara sahiptir:

-url "*providerURL*"
Dizin hizmetinin birörnek kaynak konum belirleyicisi (URL). URL aşağıdaki biçimlerden birine sahip olabilir:

- `ldap://hostname/contextName` , LDAP sunucusuna dayalı bir dizin hizmeti için
- `file:/directoryPath` , yerel dosya sistemine dayalı bir dizin hizmeti için

URL ' yi tırnak işareti (") içine almanız gerektiğini unutmayın.

-icf *initCtxOlgu*
Aşağıdaki değerlerden biri olması gereken ilk bağlam üreticisinin sınıf adı:

- `com.sun.jndi.ldap.LdapCtxFactory`, LDAP sunucusuna dayalı bir dizin hizmeti için. Bu varsayılan değerdir.
- `com.sun.jndi.fscontext.RefFSContextFactory`, yerel dosya sistemine dayalı bir dizin hizmeti için.

-t
İzleme etkinleştirildi. Varsayılan olarak izleme devre dışıdır.

Başarılı bir test, JNDI kullanmadan başarılı bir test için buna benzer bir çıkış üretir. Ana fark, çıkışın bir MQQueueConnectionFactory nesnesini ve bir MQQueue nesnesini almak için JNDI kullandığını göstermesi.

Kesinlikle gerekli olmasa da, IVTSetup komut dosyası tarafından yaratılan yönetilen nesneleri silerek testten sonra toparlamak iyi bir uygulamadır. Bu amaçla bir komut dosyası sağlanır. Komut dosyası, Windowsüzerinde AIX and Linux sistemlerinde IVTTidy ve IVTTidy.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin `bin` altdizininde bulunur.

Noktadan noktaya kuruluş doğrulama sınaması için sorun belirleme

Multi

Kuruluş doğrulama sınaması aşağıdaki nedenlerden ötürü başarısız olabilir:

- IVT programı bir sınıf bulamadığını belirten bir ileti yazarsa, sınıf yolunuzun “IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama” sayfa 89içinde açıklandığı gibi doğru ayarlandığını doğrulayın.
- Sınama şu iletiyle başarısız olabilir:

```
Failed to connect to queue manager ' qmgr ' with connection mode ' connMode '
and host name ' hostname '
```

ve ilişkili neden kodu 2059. İletideki değişkenler aşağıdaki anlamlara sahiptir:

qmgr
IVT programının bağlanmaya çalıştığı kuyruk yöneticisinin adı. IVT programı bağ tanımlama kipinde varsayılan kuyruk yöneticisine bağlanmaya çalışıyorsa, araya bu ileti boş olur.

connMode
`Bindings` ya da `Client`olan bağlantı kipi.

hostname
Kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi.

Bu ileti, IVT programının bağlanmaya çalıştığı kuyruk yöneticisinin kullanılamadığı anlamına gelir. Kuyruk yöneticisinin çalışıp çalışmadığını denetleyin ve IVT programı varsayılan kuyruk yöneticisine bağlanmaya çalışıyorsa, kuyruk yöneticisinin sisteminiz için varsayılan kuyruk yöneticisi olarak tanımlandığını doğrulayın.

- Sınama şu iletiyle başarısız olabilir:

```
Failed to open MQ queue 'SYSTEM.DEFAULT.LOCAL.QUEUE'
```

Bu ileti, kuyruğun SYSTEM.DEFAULT.LOCAL.QUEUE , IVT programının bağlı olduğu kuyruk yöneticisinde yok. Diğer bir seçenek olarak, kuyruk varsa, IVT programı ileti koymak ve almak için etkinleştirilmediği için kuyruğu açamaz. Kuyruğun var olup olmadığını ve ileti yerleştirmek ve almak için etkinleştirilip etkinleştirilmediğini denetleyin.

- Sınama şu iletiyle başarısız olabilir:

```
Unable to bind to object
```

  Bu ileti, LDAP sunucusuna yönelik bir bağlantı olduğu, ancak LDAP sunucusunun doğru yapılandırılmadığı anlamına gelir. LDAP sunucusu Java nesnelerini saklamak için yapılandırılmamış ya da nesnelere ya da soneke ilişkin izinler doğru değil. Bu durumda daha fazla yardım için LDAP sunucunuza ilişkin belgelere bakın.

- Sınama şu iletiyle başarısız olabilir:

```
The security authentication was not valid that was supplied for
QueueManager ' qmgr ' with connection mode 'Client' and host name ' hostname '
```

  Bu ileti, kuyruk yöneticisinin sisteminizden gelen bir istemci bağlantısını kabul edecek şekilde doğru ayarlanmadığı anlamına gelir. Ayrıntılar için bkz. "Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması" sayfa 1023.

IBM MQ classes for JMS için IVT yayınlama/abone olma

IBM MQ classes for JMSile birlikte bir yayınlama/abone olma kuruluş doğrulama testi (IVT) programı sağlanır. Program, bağ tanımları ya da istemci kipinde bir kuyruk yöneticisine bağlanır, bir konuya abone olur, konuyla ilgili bir ileti yayınlar ve daha sonra, yeni yayınladığı iletiyi alır. Program, yürütme sırasında dinamik olarak gerektirdiği tüm nesneleri yaratabilir ve yapılandırabilir ya da bir dizin hizmetinden yönetilen nesneleri almak için JNDI ' yı kullanabilir.

Sınama bağımsız olduğundan ve bir dizin hizmetinin kullanılmasını gerektirmediğinden, önce JNDI kullanmadan kuruluş doğrulama sınamasını çalıştırın. Denetlenen nesnelere ilişkin açıklamalar için Yönetim aracını kullanarak JMS nesnelerini yapılandırmabaşlıklı konuya bakın.

JNDI kullanmadan yayınlama/abone olma kuruluş doğrulama sınaması

Bu testte, IVT programı yürütme sırasında dinamik olarak gerektirdiği tüm nesneleri yaratır ve yapılandırır ve JNDI kullanmaz.

IVT programını çalıştırmak için bir komut dosyası sağlanır. Komut dosyası, AIX and Linux sistemlerinde PSIVTRun ve Windowssisteminde PSIVTRun.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin `bin` altdizininde bulunur.

Sınamayı bağ tanımlama kipinde çalıştırmak için şu komutu girin:

```
PSIVTRun -nojndi [-m qmgr ] [-bqm brokerQmgr ] [-v providerVersion ] [-t]
```

Sınamayı istemci kipinde çalıştırmak için, önce kuyruk yöneticisini "Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması" sayfa 1023 içinde açıklandığı gibi ayarlayın ve kullanılacak kanalın varsayılan olarak SYSTEM.DEF.SVRCONN, ardından şu komutu girin:

```
PSIVTRun -nojndi -client -m qmgr -host hostname [-port port ] [-channel channel ]
[-bqm brokerQmgr ] [-v providerVersion ] [-ccsid ccsid ] [-t]
```

Komutlara ilişkin değiştirgeler aşağıdaki anlamlara sahiptir:

-m *qmgr*
IVT programının bağlandığı kuyruk yöneticisinin adı. Sınamayı bağ tanımlama kipinde çalıştırır ve bu değiştirgeyi atlarsanız, IVT programı varsayılan kuyruk yöneticisine bağlanır.

-host *anasistemadı*
Kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi.

-port *kapı*
Kuyruk yöneticisinin dinleyicisinin dinlediği kapının numarası. Varsayılan değer 1414değeridir.

-channel ***kanal***
IVT programının kuyruk yöneticisine bağlanmak için kullandığı MQI kanalının adı. Varsayılan değer SYSTEM.DEF.SVRCONNdeğeridir.

-bqm ***brokerQmgr***
Aracının çalıştığı kuyruk yöneticisinin adı. Varsayılan değer, IVT programının bağlandığı kuyruk yöneticisinin adıdır.

Bu değiştirge, kuyruk yöneticisi sürüm numarası v /7 ya da üstü ile ilgili değil.

-v ***providerVersion***
IVT programının bağlanmayı beklediği kuyruk yöneticisinin yayın düzeyi.

Bu değiştirge, bir MQTopicConnectionFactory nesnesinin PROVIDERVERSION özelliğini ayarlamak için kullanılır ve geçerli değerler PROVIDERVERSION özelliğininkiyle aynıdır. Bu nedenle, geçerli değerleri de içinde olmak üzere, bu parametreyle ilgili daha fazla bilgi için, IBM MQ classes for JMS nesnelerinin özellikleriiçindeki PROVIDERVERSION özelliğinin tanımına bakın.

Varsayılan değer unspecifieddeğeridir.

-ccsid ***ccsid***
Bağlantı tarafından kullanılacak kodlanmış karakter takımının ya da kod sayfasının tanıtıcısı (CCSID). Varsayılan değer 819değeridir.

-t
İzleme etkinleştirildi. Varsayılan olarak izleme devre dışıdır.

Başarılı bir sınama, aşağıdaki örnek çıkışa benzer bir çıkış üretir:

```
5724-H72, 5655-R36, 5724-L26, 5655-L82 (c) Copyright IBM Corp. 2008, 2024. All
Rights Reserved.
IBM MQ classes for Java Message Service 7.0
Publish/Subscribe Installation Verification Test

Creating a TopicConnectionFactory
Creating a Connection
Creating a Session
Creating a Topic
Creating a TopicPublisher
Creating a TopicSubscriber
Creating a TextMessage
Adding text
Publishing the message to topic://MQJMS/PSIVT/Information
Waiting for a message to arrive [5 secs max]...

Got message:
JMSMessage class: jms_text
JMSType:     null
JMSDeliveryMode: 2
JMSExpiration:  0
JMSPriority:   4
JMSMessageID:   ID: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
```

JNDI kullanarak yayınlama/abone olma kuruluş doğrulama sınaması

Bu testte, IVT programı bir dizin hizmetinden yönetilen nesneleri almak için JNDI ' yı kullanır.

Sınamayı çalıştırmadan önce, LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) sunucusuna ya da yerel dosya sistemine dayalı bir dizin hizmeti yapılandırmanız gerekir. IBM MQ JMS yönetim aracını, yönetilen nesneleri saklamak için dizin hizmetini kullanabilecek şekilde yapılandırmanız da gerekir. Bu önkoşullar hakkında daha fazla bilgi için bkz. “IBM MQ classes for JMS için önkoşullar” sayfa 84. IBM MQ JMS yönetim aracının nasıl yapılandırılacağı hakkında bilgi için JMS yönetim aracının yapılandırılmasıbaşlıklı konuya bakın.

IVT programının, dizin hizmetinden bir MQTopicConnectionFactory nesnesi ve bir MQTopic nesnesini almak için JNDI ' yı kullanabilmesi gerekir. Bu yönetilen nesneleri sizin için yaratmak üzere bir komut dosyası sağlanır. Komut dosyası, AIX and Linux sistemlerinde IVTSetup ve Windowsüzerinde IVTSetup.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin `bin` altdizininde bulunur. Komut dosyasını çalıştırmak için şu komutu girin:

```
IVTSetup
```

Komut dosyası, yönetilen nesneleri yaratmak için IBM MQ JMS yönetim aracını çağırır.

MQTopicConnectionFactory nesnesi için ivtTCF adı verilir ve tüm özellikleri için varsayılan değerlerle yaratılır; bu, IVT programının bağ tanımlama kipinde çalıştığı, varsayılan kuyruk yöneticisine bağlandığı ve yerleşik yayınlama/abone olma işlevini kullandığı anlamına gelir. IVT programının istemci kipinde çalışmasını istiyorsanız, varsayılan kuyruk yöneticisinden başka bir kuyruk yöneticisine bağlanın ya da yerleşik yayınlama/abone olma işlevi yerine IBM Integration Bus komutunu kullanın; MQTopicConnectionFactory nesnesinin uygun özelliklerini değiştirmek için IBM MQ JMS denetim aracını ya da IBM MQ Gezgini 'ni kullanmanız gerekir. IBM MQ JMS yönetim aracının nasıl kullanılacağına ilişkin bilgi için Yönetim aracını kullanarak JMS nesnelerini yapılandırmabaşlıklı konuya bakın. IBM MQ Explorer 'ın nasıl kullanılacağına ilişkin bilgi için IBM MQ Explorer ile sağlanan yardıma bakın.

MQTopic nesnesi için ivtT adı bağlanır ve MQJMS/PSIVT/Information değerine sahip TOPIC özelliği dışında, tüm özellikleri için varsayılan değerlerle yaratılır.

Denetlenen nesneleri yarattığınızda, IVT programını çalıştırabilirsiniz. Sınamayı JNDI kullanarak çalıştırmak için şu komutu girin:

```
PSIVTRun -url "providerURL" [-icf initCtxFact ] [-t]
```

Komuttaki değiştirgeler aşağıdaki anlamlara sahiptir:

-url "*providerURL*"
Dizin hizmetinin birörnek kaynak konum belirleyicisi (URL). URL aşağıdaki biçimlerden birine sahip olabilir:

- `ldap://hostname/contextName` , LDAP sunucusuna dayalı bir dizin hizmeti için
- `file:/directoryPath` , yerel dosya sistemine dayalı bir dizin hizmeti için

URL ' yi tırnak işareti (") içine almanız gerektiğini unutmayın.

-icf *initCtxOlgu*
Aşağıdaki değerlerden biri olması gereken ilk bağlam üreticisinin sınıf adı:

- `com.sun.jndi.ldap.LdapCtxFactory`, LDAP sunucusuna dayalı bir dizin hizmeti için. Bu varsayılan değerdir.
- `com.sun.jndi.fscontext.RefFSContextFactory`, yerel dosya sistemine dayalı bir dizin hizmeti için.

-t
İzleme etkinleştirildi. Varsayılan olarak izleme devre dışıdır.

Başarılı bir test, JNDI kullanmadan başarılı bir test için buna benzer bir çıkış üretir. Ana fark, çıkışın bir MQTopicConnectionFactory nesnesini ve bir MQTopic nesnesini almak için JNDI kullandığını göstermesi.

Kesinlikle gerekli olmasa da, IVTSetup komut dosyası tarafından yaratılan yönetilen nesneleri silerek testten sonra toparlamak iyi bir uygulamadır. Bu amaçla bir komut dosyası sağlanır. Komut dosyası, Windowsüzerinde AIX and Linux sistemlerinde IVTTidy ve IVTTidy.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin `bin` altdizininde bulunur.

Yayınlama/abone olma kuruluş doğrulama testi için sorun belirleme

Kuruluş doğrulama sınaması aşağıdaki nedenlerden ötürü başarısız olabilir:

- IVT programı bir sınıf bulamadığını belirten bir ileti yazarsa, sınıf yolunuzun "IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama" sayfa 89içinde açıklandığı gibi doğru ayarlandığını doğrulayın.
- Sınama şu iletiyle başarısız olabilir:

```
Failed to connect to queue manager ' qmgr ' with
connection mode ' connMode ' and host name ' hostname '
```

  ve ilişkili neden kodu 2059. İletideki değişkenler aşağıdaki anlamlara sahiptir:

  qmgr
  IVT programının bağlanmaya çalıştığı kuyruk yöneticisinin adı. IVT programı bağ tanımlama kipinde varsayılan kuyruk yöneticisine bağlanmaya çalışıyorsa, araya bu ileti boş olur.

  connMode
  `Bindings` ya da `Client`olan bağlantı kipi.

  hostname
  Kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi.

  Bu ileti, IVT programının bağlanmaya çalıştığı kuyruk yöneticisinin kullanılamadığı anlamına gelir. Kuyruk yöneticisinin çalışıp çalışmadığını denetleyin ve IVT programı varsayılan kuyruk yöneticisine bağlanmaya çalışıyorsa, kuyruk yöneticisinin sisteminiz için varsayılan kuyruk yöneticisi olarak tanımlandığını doğrulayın.
- Sınama şu iletiyle başarısız olabilir:

```
Unable to bind to object
```

  Bu ileti, LDAP sunucusuna yönelik bir bağlantı olduğu, ancak LDAP sunucusunun doğru yapılandırılmadığı anlamına gelir. LDAP sunucusu Java nesnelerini saklamak için yapılandırılmamış ya da nesnelere ya da soneke ilişkin izinler doğru değil. Bu durumda daha fazla yardım için LDAP sunucunuza ilişkin belgelere bakın.
- Sınama şu iletiyle başarısız olabilir:

```
The security authentication was not valid that was supplied for
QueueManager 'qmgr' with connection mode 'Client' and host name 'hostname'
```

  Bu ileti, kuyruk yöneticisinin sisteminizden gelen bir istemci bağlantısını kabul edecek şekilde doğru ayarlanmadığı anlamına gelir. Daha fazla bilgi için bkz "Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması" sayfa 1023.

JMS 2.0 *IBM MQ classes for JMS örnek uygulamalarının kullanılması*

IBM MQ classes for JMS örnek uygulamaları, JMS API ' nin ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluş ve ileti sistemi sunucunuzun kurulumunu doğrulamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

Bu görev hakkında

Kendi uygulamalarınızı yaratmak için yardıma gereksinim duyarsanız, örnek uygulamaları başlangıç noktası olarak kullanabilirsiniz. Her uygulama için hem kaynak hem de derlenmiş bir sürüm sağlanır. Örnek kaynak kodunu gözden geçirin ve uygulamanız için gerekli her bir nesneyi (ConnectionFactory, Connection, Session, Destination, and a Producer, and a Consumer ya da her ikisi) yaratmak ve uygulamanızın nasıl çalışmasını istediğinizi belirtmek için gereken belirli özellikleri ayarlamak için gereken anahtar adımlarını tanımlayın. Daha fazla bilgi için bkz "IBM MQ classes for JMS/Jakarta Messaging uygulamaları yazılıyor" sayfa 134. Örnekler, IBM MQ' in sonraki yayınlarında değiştirilebilir.

JMS 2.0için Çizelge 11 sayfa 115 , IBM MQ classes for JMS örnek uygulamalarının her bir platformda nereye kurulduğunu gösterir.

Not:

JM 3.0 IBM MQ classes for Jakarta Messagingiçin yeni örnekler hazırlanıyor.

Çizelge 11. IBM MQ classes for JMS örnek uygulamaları için kuruluş dizinleri

Hizmet olarak sunulan	**Dizin**
AIX AIX Linux Linux	*MQ_INSTALLATION_PATH*/samp/jms/samples
Windows Windows	*MQ_INSTALLATION_PATH*\tools\jms\samples
IBM i IBM i	/qibm/proddata/mqm/java/samples/jms/samples
z/OS z/OS	*MQ_INSTALLATION_PATH*/java/samples/jms

Bu dizin içinde, Çizelge 12 sayfa 115içinde gösterildiği gibi bir ya da daha fazla örnek uygulama içeren alt dizinler vardır.

Çizelge 12. IBM MQ classes for JMS örnek uygulamalar

Örneğin adı	**Açıklama**
JmsBrowser.java	Bir tüketici uygulaması tarafından alınma sırasına göre, adı belirtilen kuyruktaki tüm iletileri kaldırmadan görüntüleyen bir JMS kuyruk tarayıcısı uygulaması.
JmsConsumer.java	Bir tüketici uygulaması tarafından alınma sırasına göre, bağlantı üreticisi eşgörünümünü ve hedef eşgörünümü ilk bağlamda arayarak, adı belirtilen kuyruktaki tüm iletileri kaldırmadan inceleyen bir JMS kuyruk tarayıcısı uygulaması (Bu örnek yalnızca dosya sistemi bağlamını destekler).
JmsJndiConsumer.java	Adı belirtilen hedeften (kuyruk ya da konu) bağlantı üreticisi eşgörünümünü ve hedef eşgörünümü ilk bağlamda arayarak ileti alan bir JMS tüketici (alıcı ya da abone) uygulaması (Bu örnek yalnızca dosya sistemi bağlamını destekler).
JmsJndiProducer.java	Bağlantı üreticisi eşgörünümünü ve hedef eşgörünümü ilk bağlamda arayarak adı belirtilen hedefe (kuyruk ya da konu) yalın bir ileti gönderen bir JMS üreticisi (gönderen ya da yayınlayıcı) uygulaması (Bu örnek, yalnızca dosya sistemi bağlamını destekler).
JmsProducer.java	Adı belirtilen hedefe (kuyruk ya da konu) basit bir ileti gönderen bir JMS üreticisi (gönderen ya da yayınlayıcı) uygulaması.
/interactive/	
SampleConsumerJava.java	Bir konu/kuyruktan ileti alın.

Çizelge 12. IBM MQ classes for JMS örnek uygulamalar (devamı var)	
Örneğin adı	**Açıklama**
SampleProducerJava.java	İletileri bir konu/kuyruğa gönderin.
/interactive/helper/	
BaseOptions.java (Temel Seçenekler)	Kullanıcı seçenek (ler) i işlevselliğini sağlamak üzere genişletilebilen bir soyut sınıf.
IsValidType.java	Geçerlilik denetleyici sınıfları için soyut sınıf.
JmsApp.java	Tüketici/üretici işlevselliğini sağlamak için genişletilebilen bir soyut sınıf.
Keys.java	Örnek uygulamalara ilişkin seçenekleri tanımlayan bir anahtar kümesi.
Literals.java	Sabit hazır bilgiler kümesi.
MyContext.java	Seçeneklerin sunulduğu bağlam.
Options.java	Kullanıcı seçenek (ler) i için işlevsellik sağlar.
OptionsPresenter.java	Geçerli seçeneklerin sunulduğu bağlam.
/basit/	
SimpleAsyncPutPTP.java	Noktadan noktaya ileti sistemi için basit bir uygulama; ileti zamanuyumsuz olarak gönderilir (*anında ileti sistemi ve unut* olarak da bilinir). Hiçbir ileti alınmadı.
SimpleDurableSub.java	Sürekli abonelik olanağını gösteren basit bir uygulama.
SimpleJNDILookup.java	Başlangıç bağlamı kullanılarak JMS nesnelerinin aranmasını gösteren basit ve minimal bir uygulama. Kuyruk yöneticisiyle bağlantı kurulmaz ve ileti gönderilmez ya da alınmaz.
SimpleMQMDRead.java	Bir JMS uygulamasının MQ Message Descriptor (MQMD) alanlarını JMS ileti özellikleri olarak nasıl sağlayabileceğini gösteren basit bir uygulama. Hiçbir ileti gönderilmez; kullanılmakta olan kuyruğa bazı iletiler yerleştirildiği varsayılır.
SimpleMQMDWrite.java	Bir JMS uygulamasının MQ Message Descriptor (MQMD) alanlarını nasıl yazabileceğini gösteren basit bir uygulama. Hiçbir ileti alınmadı.
SimplePTP.java	Noktadan noktaya ileti sistemi için minimum ve basit bir uygulama.
SimplePubSub.java	Yayınlama-abone olma ileti sistemi için minimal ve basit bir uygulama.
SimpleReadAheadPTP.java	Noktadan noktaya ileti sistemi için basit bir uygulama; iletiler kuyruk yöneticisinden (önden okuma olanağı olarak da bilinir) akıtılır. Hiçbir ileti gönderilmez; kullanılmakta olan kuyruğa bazı iletiler yerleştirildiği varsayılır.
SimpleRequestor.java	İstek iletisi göndermek için istekte bulunan ve ardından yanıtı bekleyen ve alan basit bir uygulama. Not: Başka bir uygulamanın istek iletisini işleyeceği ve yanıt iletisini göndereceği varsayılır.

Çizelge 12. IBM MQ classes for JMS örnek uygulamalar (devamı var)	
Örneğin adı	**Açıklama**
SimpleResponder.java	Bir iletinin hedefini dinleyen ve iletinin replyTo hedefine yanıt gönderen basit bir uygulamadır. Uygulama, SimpleRequestor örneğiyle birlikte çalışmak üzere yazılır.
SimpleRetainedPub.java	Alıkonan bir yayını gösteren basit bir uygulama. Hiçbir ileti alınmadı.
SimpleWMQJMSPTP.java	Noktadan noktaya ileti sistemi için minimum ve basit bir uygulama.
SimpleWMQJMSPubSub.java	Yayınlama/abone olma ileti sistemi için minimum ve basit bir uygulama.

IBM MQ classes for JMS , örnek uygulamaları çalıştırmak için kullanılabilecek `runjms` adlı bir komut dosyası sağlar. Bu komut dosyası, IBM MQ classes for JMS örnek uygulamalarını çalıştırmanıza izin vermek için IBM MQ ortamını ayarlar.

Çizelge 13 sayfa 117 içinde her altyapıda komut dosyasının yeri gösterilir:

Çizelge 13. `runjms` komut dosyasının konumu	
Hizmet olarak sunulan	**Dizin**
AIX AIX Linux Linux	*MQ_INSTALLATION_PATH*`/java/bin/runjms`
Windows Windows	*MQ_INSTALLATION_PATH*`\java\bin\runjms.bat`

IBM i IBM i	`/qibm/proddata/mqm/java/bin/runjms`
	veya
	`/qibm/proddata/mqm/java/bin/runjms64`

z/OS z/OS	*MQ_INSTALLATION_PATH*`/java/bin/runjms`

Örnek bir uygulamayı çağırmak üzere `runjms` komut dosyasını kullanmak için aşağıdaki adımları izleyin:

Yordam

1. Bir komut istemi açın ve çalıştırmak istediğiniz örnek uygulamayı içeren dizine gidin.
2. Aşağıdaki komutu girin:

   ```
   Path to the runjms script/runjms sample_application_name
   ```

   Örnek uygulama, gereksinim duyduğu parametrelerin bir listesini görüntüler.
3. Örneği bu parametrelerle çalıştırmak için aşağıdaki komutu girin:

   ```
   Path to the runjms script/runjms sample_application_name parameters
   ```

Örnek

Linux Örneğin, Linuxüzerinde `JmsBrowser` örneğini çalıştırmak için aşağıdaki komutları girin:

```
cd /opt/mqm/samp/jms/samples
/opt/mqm/java/bin/runjms JmsBrowser -m QM1 -d LQ1
```

İlgili kavramlar

“IBM MQ classes for JMS için kurulu olan” sayfa 84
IBM MQ classes for JMSkurulurken bir dizi dosya ve dizin oluşturulur. Windows' ta, ortam değişkenleri otomatik olarak ayarlanarak kuruluş sırasında bazı yapılandırma gerçekleştirilir. Diğer platformlarda ve belirli Windows ortamlarında IBM MQ classes for JMS uygulamalarını çalıştırmadan önce ortam değişkenlerini ayarlamanız gerekir.

IBM MQ classes for JMS/Jakarta Messaging ile sağlanan komut dosyaları

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingkullanılırken gerçekleştirilmesi gereken ortak görevlere yardımcı olmak için bir dizi komut dosyası sağlanır.

Çizelge 14 sayfa 118 içinde tüm komut dosyaları ve kullanımları listelenir. Komut dosyaları, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging kuruluş dizininin `bin` altdizininde bulunur.

Çizelge 14. IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging ile birlikte sağlanan komut dosyaları

Yardımcı Program	**Kullanım**
Temizleme [1]	Bu komut dosyası önceki yayınlarla uyumluluk için korunur, ancak işlev gerçekleştirmez. Abonelik bilgilerinin el ile temizlenmesi artık gerekli değildir.
DefaultConfiguration	Windowsdışındaki platformlarda varsayılan yapılandırma uygulamasını çalıştırır.
formatLog [1]	Bu komut dosyası önceki yayınlarla uyumluluk için korunur, ancak işlev gerçekleştirmez. Günlük çıkışı okunabilir metinde üretilir.
IVTRUN [1] IVTSetup [1] IVTTidy [1]	“IBM MQ classes for JMS için noktadan noktaya IVT” sayfa 107içinde açıklandığı gibi noktadan noktaya kuruluş doğrulama sınamasında kullanılır.
JM 3.0 JMS30Admin [1]	IBM MQ Jakarta Messaging yönetim aracını Yönetim aracının başlatılmasıkonusunda açıklandığı gibi çalıştırır.
JM 3.0 JMS30Admin.config	IBM MQ Jakarta Messaging yönetim aracına ilişkin yapılandırma dosyası, JMS yönetim aracının yapılandırılmasıkonusunda açıklandığı gibi.
JMS 2.0 JMSAdmin [1]	IBM MQ JMS yönetim aracını Yönetim aracının başlatılmasıkonusunda açıklandığı gibi çalıştırır.
JMS 2.0 JMSAdmin.config	IBM MQ JMS yönetim aracına ilişkin yapılandırma dosyası, JMS yönetim aracının yapılandırılmasıkonusunda açıklandığı gibi.
PSIVTRUN [1]	“IBM MQ classes for JMS için IVT yayınlama/abone olma” sayfa 111içinde açıklandığı gibi, yayınlama/abone olma kuruluş doğrulama sınaması programını çalıştırır.
PSReportDump.class	Bu sınıf, önceki yayınlarla uyumluluk için korunur, ancak işlev gerçekleştirmez.

Çizelge 14. IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging ile birlikte sağlanan komut dosyaları (devamı var)

Yardımcı Program	Kullanım
V 9.3.0 V 9.3.0 JM 3.0 setjms30env "2" sayfa 119	Linux AIX Jakarta Messaging 3.0için, bir IBM MQ classes for JMS uygulamasını AIX and Linux sistemlerinde 32 bit Java sanal makinesinde (JVM) çalıştırmaya ilişkin ortam değişkenlerini "IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama" sayfa 89içinde açıklandığı gibi ayarlar.
JMS 2.0 setjmsenv "2" sayfa 119	Linux AIX JMS 2.0için, bir IBM MQ classes for JMS uygulamasını AIX and Linux sistemlerinde 32 bit Java sanal makinesinde (JVM) çalıştırmaya ilişkin ortam değişkenlerini "IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama" sayfa 89içinde açıklandığı gibi ayarlar.
V 9.3.0 V 9.3.0 JM 3.0 setjms30env64 "2" sayfa 119	Linux AIX Jakarta Messaging 3.0için, "IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama" sayfa 89içinde açıklandığı gibi, AIX and Linux sistemlerinde 64 bit JVM ' de bir IBM MQ classes for JMS uygulamasını çalıştırmak için ortam değişkenlerini ayarlar.
JMS 2.0 setjmsenv64 "2" sayfa 119	Linux AIX JMS 2.0için, "IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama" sayfa 89içinde açıklandığı gibi, AIX and Linux sistemlerinde 64 bit JVM ' de bir IBM MQ classes for JMS uygulamasını çalıştırmak için ortam değişkenlerini ayarlar.

Not:

1. Windows' da dosya adı `.bat`uzantısına sahiptir.
2. Bu komut dosyaları yalnızca AIX and Linux üzerinde kullanılabilir. Windowsişletim sistemlerinde, IBM MQkurulduktan sonra **setmqenv**komutunu çalıştırın. Daha fazla bilgi için bkz "IBM MQ classes for JMS/Jakarta Messaging için ortam değişkenlerini ayarlama" sayfa 89.

IBM MQ classes for JMS ile OSGi desteği

OSGi, uygulamaların kod paketleri olarak konuşlandırılmasını destekleyen bir çerçeve sağlar. OSGi paketleri, IBM MQ classes for JMS' nin bir parçası olarak sağlanır.

IBM MQ classes for JMS aşağıdaki OSGi paketlerini içerir.

com.ibm.msg.client.osgi.jms*version_number*.jar
IBM MQ classes for JMSiçindeki ortak kod katmanı. JMSiçin IBM MQ sınıflarının katmanlı mimarisi hakkında bilgi için bkz. IBM MQ JMS mimarisi sınıfları.

com.ibm.msg.client.osgi.jms.prereq_*version_number*.jar
Ortak katman için önkoşul olan Java arşiv (JAR) dosyaları.

com.ibm.msg.client.osgi.commonservices.j2se_*version_number*.jar
Java Platform, Standard Edition (Java SE) uygulamaları için ortak hizmetler.

com.ibm.msg.client.osgi.nls_*version_number*.jar
Ortak katman için iletiler.

com.ibm.msg.client.osgi.wmq_*version_number*.jar
IBM MQ classes for JMSiçindeki IBM MQ ileti alışverişi sağlayıcısı. IBM MQ classes for JMSkatmanlı mimarisi hakkında bilgi için bkz. IBM MQ JMS mimarisi sınıfları.

com.ibm.msg.client.osgi.wmq.prereq_*version_number*.jar
IBM MQ ileti alışverişi sağlayıcısı için önkoşul JAR dosyaları.

com.ibm.msg.client.osgi.wmq.nls_*version_number*.jar
IBM MQ ileti alışverişi sağlayıcısına ilişkin iletiler.

com.ibm.mq.jakarta.osgi.allclient_*version_number*.jar
V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging 3.0için bu JAR dosyası, uygulamaların IBM MQ classes for JMS ve IBM MQ classes for Javadosyalarını kullanmasını sağlar ve PCF iletilerini işlemek için kodu da içerir.

com.ibm.mq.jakarta.osgi.allclientprereqs_*version_number*.jar
V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging 3.0için, bu JAR dosyası `com.ibm.mq.jakarta.osgi.allclient_version_number.jar`ile ilgili önkoşulları sağlar.

com.ibm.mq.osgi.allclient_*version_number*.jar
JMS 2.0 JMS 2.0için bu JAR dosyası, uygulamaların hem IBM MQ classes for JMS hem de IBM MQ classes for Javakullanmasına olanak sağlar ve PCF iletilerini işlemek için kodu da içerir.

com.ibm.mq.osgi.allclientprereqs_*version_number*.jar
JMS 2.0 JMS 2.0için, bu JAR dosyası `com.ibm.mq.osgi.allclient_version_number.jar`ile ilgili önkoşulları sağlar.

Burada *version_number* , kurulu IBM MQ sürüm numarasıdır.

Paketler, IBM MQ kurulumunuzun `java/lib/OSGi` alt dizinine ya da Windowsüzerindeki `java\lib\OSGi` klasörüne kurulur.

IBM MQ 8.0olanağından, yeni uygulamalar için `com.ibm.mq.osgi.allclient_8.0.0.0.jar`ve `com.ibm.mq.osgi.allclientprereqs_8.0.0.0.jar` paketlerini kullanın. Bu kod paketlerinin kullanılması, aynı OSGi çerçevesi içinde hem IBM MQ classes for JMS , hem de IBM MQ classes for Java çalıştırılamamasının kısıtlamasını kaldırır; ancak, diğer tüm kısıtlamalar geçerli olmaya devam eder.

IBM MQ kurulumunuzun `java/lib/OSGi` alt dizinine ya da Windowsüzerindeki `java\lib\OSGi` klasörüne kurulan `com.ibm.mq.osgi.java`*version_number*`.jar`paketi, IBM MQ classes for Java' nin bir parçasıdır. Bu kod paketi, IBM MQ classes for JMS yüklü olan bir OSGi yürütme ortamına yüklenmemelidir.

IBM MQ classes for JMS için OSGi paketleri, OSGi Yayın Düzeyi 4 belirtimine yazıldı. Bunlar OSGi Yayın Düzeyi 3 ortamında çalışmaz.

OSGi yürütme ortamının gereken DLL dosyalarını ya da paylaşılan kitaplıkları bulabilmesi için sistem yolunuzu ya da kitaplık yolunu doğru olarak ayarlamanız gerekir.

IBM MQ classes for JMSiçin OSGi paketlerini kullanırsanız, geçici konular çalışmaz. Buna ek olarak, OSGi gibi birden çok sınıflı bir yükleyici ortamındaki sınıfların yüklenmesinde oluşan bir sorun nedeniyle Java içine yazılan kanal çıkış sınıfları desteklenmez. Bir kullanıcı paketi IBM MQ classes for JMS kod paketlerini bilebilir, ancak IBM MQ classes for JMS kod paketleri herhangi bir kullanıcı kod paketini bilemez. Sonuç olarak, IBM MQ classes for JMS kod paketinde kullanılan sınıf yükleyici, kullanıcı kod paketinde bulunan bir kanal çıkış sınıfını yükleyemez.

OSGi hakkında daha fazla bilgi için OSGi Alliance web sitesine bakın.

z/OS MQ Adv. VUE JMS/Jakarta Messaging üzerinde çalışan toplu iş uygulamalarına istemci bağlanırlığı z/OS

Belirli koşullar altında, z/OS üzerindeki bir IBM MQ classes for JMS/Jakarta Messaging uygulaması, z/OS üzerindeki bir kuyruk yöneticisine istemci bağlantısı kullanarak bağlanabilir. İstemci bağlantısının kullanılması IBM MQ topolojilerini basitleştirebilir.

Bir istemci bağlantısı kullanarak, bir IBM MQ classes for JMS/Jakarta Messaging uygulaması toplu iş ortamında çalışıyorsa ve aşağıdaki koşullardan biri geçerliyse, uygulama uzak bir z/OS kuyruk yöneticisine bağlanabilir:

- LTS V 9.3.4 IBM MQ classes for JMS/Jakarta Messaging kodu IBM MQ 9.3.4 ya da sonraki bir sürümde ya da Long Term Support APAR PH56722 uygulanmış olarak bulunur. Kuyruk yöneticisi desteklenen herhangi bir sürümde olabilir.
- Bağlı olduğu kuyruk yöneticisi IBM MQ Advanced for z/OS Value Unit Edition yetkisiyle çalışıyor ve bu nedenle **ADVCAP** parametresi ENABLED olarak ayarlanmış. Kuyruk yöneticisi desteklenen herhangi bir sürümde olabilir.

IBM MQ Advanced for z/OS Value Unit Edition hakkında daha fazla bilgi için bkz. IBM MQ ürün tanıtıcıları ve dışa aktarma bilgileri.

QMGRPRODile ilgili daha fazla bilgi için bkz. DISPLAY QMGR **ADVCAP** ve START QMGR .

Toplu işin desteklenen tek ortam olduğunu unutmayın; CICS için JMS/Jakarta Messaging ya da IMSiçin JMS/Jakarta Messaging desteği yoktur.

z/OS üzerindeki bir IBM MQ classes for JMS/Jakarta Messaging uygulaması, z/OSayarlanmamış bir kuyruk yöneticisine bağlanmak için istemci kipi bağlantısını kullanamaz.

z/OS üzerinde bir IBM MQ classes for JMS/Jakarta Messaging uygulaması istemci kipini kullanarak bağlanmayı denerse ve buna izin verilmezse, JMSFMQ0005 kural dışı durum iletisi yayınlanır.

Advanced Message Security (AMS) desteği

IBM MQ classes for JMS/Jakarta Messaging istemci uygulamaları, bu konuda daha önce açıklanan koşullara bağlı olarak, uzak z/OS kuyruk yöneticilerine bağlanırken AMS kullanabilir.

AMS ' u bu şekilde kullanmak için, istemci uygulamalarının `keystore.conf`içinde `jceracfks` anahtar deposu tipini kullanması gerekir; burada:

- Özellik adı öneki `jceracfks` ve bu ad öneki büyük ve küçük harfe duyarlı değildir.
- Anahtar deposu bir RACF anahtarlığı.
- Parolalar gerekli değildir ve yoksayılır. Bunun nedeni, RACF anahtarlarının parola kullanmaması olabilir.
- Sağlayıcıyı belirtirseniz, sağlayıcı `IBMJCE`olmalıdır.

`jceracfks` komutunu AMSile kullanırken anahtar deposu şu biçimde olmalıdır: `safkeyring://`*`user`*`/`*`keyring`*; burada:

- `safkeyring` bir hazır bilgi ve bu ad büyük ve küçük harfe duyarlı değildir,,,
- *`user`* , anahtarlık sahibinin RACF kullanıcı kimliğidir.
- *`keyring`* , RACF anahtarının adıdır ve anahtarlık adı büyük ve küçük harfe duyarlıdır

Aşağıdaki örnekte, kullanıcı için standart AMS anahtarlık `JOHNDOE`kullanılmıştır:

```
jceracfks.keystore=safkeyring://JOHNDOE/drq.ams.keyring
```

İlgili kavramlar

“Java üzerinde çalışan toplu iş uygulamalarına istemci bağlanırlığı z/OS” sayfa 357
Belirli koşullar altında, z/OS üzerindeki bir IBM MQ classes for Java uygulaması, z/OS üzerindeki bir kuyruk yöneticisine istemci bağlantısı kullanarak bağlanabilir. İstemci bağlantısının kullanılması IBM MQ topolojilerini basitleştirebilir.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging ürünlerini ayrı olarak edinme

IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging kullanan uygulamaları çalıştırmak için gereken kitaplıklar ve yardımcı programlar, Fix Central' den yüklediğiniz kendi kendini açan JAR dosyasında bulunur. Yalnızca bu dosyaları (örneğin, bir yazılım yönetimi aracında devreye alma ya da bağımsız istemci uygulamalarıyla kullanım için) almak istiyorsanız bunu yaparsınız.

Başlamadan önce

Bu görevi başlatmadan önce, makinenizde bir Java runtime environment (JRE) kurulu olduğundan ve JRE ' nin sistem yoluna eklendiğinden emin olun.

Bu kuruluş işleminde kullanılan Java kuruluş programı, kök kullanıcı ya da belirli bir kullanıcı olarak çalıştırılmasını gerektirmez. Tek gereksinim, çalıştırıldığı kullanıcının, dosyaların yerleştirilmesini istediğiniz dizine yazma erişimine sahip olması.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Aynı uygulamada hem Jakarta Messaging 3.0 API hem de JMS 2.0 API ' nin kullanılması desteklenmez. Daha fazla bilgi için, bkz. JMS/Jakarta İleti Sistemi için IBM MQ sınıflarının kullanılması.

Bu görev hakkında

Kendi kendini ayıklayan JAR dosyası, karşıdan yüklemenin boyutunu ve çıkarma işlemini gerçekleştirmek için gereken süreyi en aza indirmek için kullanılır. Bu JAR dosyasının tam içeriği ve dosyaları ayıkladığı alt dizinler, IBM MQsürümüne bağlıdır.

Kendi kendini açan JAR dosyasını çalıştırdığınızda, kabul edilmesi gereken IBM MQ lisans sözleşmesi görüntülenir. Ayrıca, alma işlemi için üst dizini değiştirmenizi de sağlar.

IBM MQ 9.3için, kendi kendini açan JAR dosyası dosyaları aşağıdaki dizin yapısına çıkarır:

wmq/JavaEE
IBM MQ kaynak bağdaştırıcısı EAR ve RAR dosyaları.

JMS 2.0 Aşağıdaki dosyalar JMS 2.0 ve JMS 1.1 nesneleriyle birlikte kullanılır:

- `wmq.jmsra.ivt.ear`
- `wmq.jmsra.rar`

JM 3.0 Jakarta Messaging 3.0 nesneleriyle kullanılmak üzere eşdeğer bir küme de vardır:

- `wmq.jakarta.jmsra.ivt.ear`. Kuruluş Doğrulama Sınaması dosyalarını içerir.
- `wmq.jakarta.jmsra.rar`. Kaynak bağdaştırıcısı dosyalarını içerir.

wmq/JavaSE

wmq/JavaSE/bin
JMSAdmin ve **JMS30Admin** araçları. JMS ya da Jakarta Messaging nesnelerini gösteren JNDI varlıklarını tanımlamak için kullanılır.

JMS 2.0 Aşağıdaki dosyalar JMS 2.0 ve JMS 1.1 nesneleriyle birlikte kullanılır:

- `JMSAdmin.bat`
- `JMSAdmin`
- `JMSAdmin.config`

JM 3.0 Jakarta Messaging 3.0 nesneleriyle kullanılmak üzere eşdeğer bir küme de vardır:

- `JMS30Admin.bat`. Aracı Windowsüzerinde başlatmak için kullanılan bir dosya.
- `JMS30Admin`. Aracı Linux ve UNIX platformlarında başlatmak için kullanılan bir komut dosyası.
- `JMS30Admin.config`. Araç için örnek bir yapılandırma dosyası.

Not:

- **JMSAdmin** aracı kendi kendini açan JAR dosyasına eklenmeden önce, bu dizindeki dosyalar `wmq/JavaSE`üst dizinindeydi.

- Kendi kendini açan JAR dosyası kullanılarak kurulan bir istemci, bir dosya sistemi bağlamı (`.bindings` dosyası) içinde Java ileti alışverişi denetimli nesneleri yaratmak için **JMSAdmin** ya da **JMS30Admin** aracını kullanabilir. İstemci de bu yönetilen nesneleri arayabilir ve kullanabilir.
- JMS 2.0 JMS 2.0 ve JMS 1.1 nesneleriyle kullanılacak **JMSAdmin** aracı, IBM MQ 9.2.0 Fix Pack 2 ve IBM MQ 9.2.2adresindeki kendi kendini açan JAR dosyasına eklendi.
- JM 3.0 Jakarta Messaging 3.0 nesneleriyle kullanılacak **JMS30Admin** aracı, IBM MQ 9.3.0adresindeki kendi kendini açan JAR dosyasına eklendi.

wmq/JavaSE/lib

Advanced Message Security, Şifreleme İletisi Sözdizimini (CMS) desteklemek için aşağıdaki açık kaynak Bouncy Castle paketlerini kullanır. Bkz. Support for non-IBM JRE with AMS.

V 9.3.5 IBM MQ 9.3.5' dan Continuous Delivery için:

- `bcpkix-jdk18on.jar`
- `bcprov-jdk18on.jar`
- `bcutil-jdk18on.jar`

LTS IBM MQ 9.3.5öncesinde Long Term Support ve Continuous Delivery için:

- `bcpkix-jdk15on.jar`
- `bcprov-jdk15on.jar`
- `bcutil-jdk15on.jar`

Aşağıdaki dosyaların her biri, belirli JMS ya da Jakarta Messaging düzeylerine ilişkin sınıfları içerir:

- JMS 2.0 `com.ibm.mq.allclient.jar` (JMS 2.0 ve JMS 1.1)
- JM 3.0 `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0)

Diğer önkoşul JAR dosyaları:

- Removed V 9.3.3 `com.ibm.mq.traceControl.jar`. IBM MQ classes for JMS uygulamalarına ilişkin izlemeyi dinamik olarak denetlemek için kullanılır.
- `fscontext.jar`. Uygulamanız bir dosya sistemi bağlamı kullanarak JNDI aramaları gerçekleştiriyorsa gereklidir.
- V 9.3.3 `jackson-annotations.jar`, `jackson-core.jar`, `jackson-databind.jar`: Bir kuyruk yöneticisine güvenli TLS bağlantıları yaratılırken CipherSuite ve CipherSpec eşlemelerini gerçekleştirmek için kullanılan sınıfları içerir.
- JM 3.0 `jakarta.jms-api.jar`. Jakarta Messaging 3.0 arabirimini ve Kural Dışı Durum tanımlarını içerir.
- JMS 2.0 `jms.jar`. JMS 2.0 arabirimini ve Kural Dışı Durum tanımlarını içerir.
- `org.json.jar`. IBM MQ classes for JMS ' in JSON biçimli CCDT dosyalarını yorumlamasına izin veren sınıfları içerir.
- `providerutil.jar`. Uygulamanız bir dosya sistemi bağlamı kullanarak JNDI aramaları gerçekleştiriyorsa gereklidir.

Not: Stabilized `com.ibm.mq.allclient.jar` ve `com.ibm.mq.jakarta.client.jar` her ikisi de IBM MQ classes for Java' nin bir kopyasını içerir. Ancak IBM MQ 9.0içinde bu sınıflar, IBM MQ 8.0içinde gönderilen düzeyde işlevsel olarak stabilize edilmiş olarak bildirilir. Bkz. IBM MQ 9.0' da kullanımdan kaldırma, dengeleme ve dengeleme.

wmq/OSGi

IBM MQ OSGi istemcisi kod paketleri:

- JM 3.0 `com.ibm.mq.jakarta.osgi.allclient_V.R.M.F.jar`
- JM 3.0 `com.ibm.mq.jakarta.osgi.allclientprereqs_V.R.M.F.jar`
- JMS 2.0 `com.ibm.mq.osgi.allclient_V.R.M.F.jar`
- JMS 2.0 `com.ibm.mq.osgi.allclientprereqs_V.R.M.F.jar`

Burada *V.R.M.F* , Sürüm, Yayın, Değişiklik ve Düzeltme Paketi numarasıdır.

Yordam

1. IBM MQ Java / JMS istemcisi JAR dosyasını Fix Centraladresinden yükleyin.
   a) Bu bağlantıyı tıklatın: IBM MQ Java / JMS client.
   b) Görüntülenen kullanılabilir düzeltmeler listesinde IBM MQ sürümünüze ilişkin istemciyi bulun.
   Örneğin:

   ```
   release level: 9.3.0.0-IBM-MQ-Install-Java-All
   Long Term Support: 9.3.0.0 IBM MQ JMS and Java 'All Client'
   ```

   Daha sonra istemci dosya adını tıklatın ve karşıdan yükleme işlemini izleyin.
2. Dosyayı karşıdan yüklediğiniz dizinden alma işlemini başlatın.
   Alma işlemini başlatmak için aşağıdaki biçimde bir komut girin:

   ```
   java -jar V.R.M.F-IBM-MQ-Install-Java-All.jar
   ```

   Burada *V.R.M.F* , ürün sürümü numarasıdır; örneğin, `9.3.0.0`ve *V.R.M.F*`-IBM-MQ-Install-Java-All.jar` , Fix Centraladresinden yüklenen dosyanın adıdır.
   Örneğin, IBM MQ 9.3.0 yayınına ilişkin JMS istemcisini çıkarmak için aşağıdaki komutu kullanabilirsiniz:

   ```
   java -jar 9.3.0.0-IBM-MQ-Install-Java-All.jar
   ```

   Not: Bu kuruluşu gerçekleştirmek için, makinenizde kurulu bir JRE olması ve sistem yoluna eklenmiş olması gerekir.

   Komutu girdiğinizde aşağıdaki bilgiler görüntülenir:

   ```
   IBM MQ V9.3ürününü kullanmadan, ayıklamadan ya da kurmadan önce kabul etmeniz gerekir
   1 'in koşulları. IBM Uluslararası Lisans Sözleşmesi-Değerlendirme
   Programlar 2. IBM Uluslararası Program Lisans Sözleşmesi ve ek
   lisans bilgileri. Lütfen aşağıdaki lisans sözleşmelerini dikkatle okuyun.

   Lisans sözleşmesi,
   --viewLicenseSözleşmesi seçeneği.

   Lisans koşullarını şimdi görüntülemek için Enter, atlamak için 'x' tuşuna basın.
   ```
3. Lisans koşullarını inceleyin ve kabul edin:
   a) Lisansı görüntülemek için Enter tuşuna basın.
   Diğer bir seçenek olarak, lisansın görüntülenmesini atlamak için x tuşuna basın.
   Lisans görüntülendikten sonra ya da x tuşuna basarsanız hemen aşağıdaki ileti görüntülenir:

   ```
   Ek lisans bilgileri,
   --viewLicenseBilgi seçeneği.

   Ek lisans bilgilerini şimdi görüntülemek için Enter, atlamak için 'x' tuşuna basın.
   ```

   b) Ek lisans koşullarını görüntülemek için Enter tuşuna basın.
   Diğer bir seçenek olarak, ek lisans koşullarının görüntülenmesini atlamak için x tuşuna basın.
   Ek lisans koşulları görüntülendikten sonra ya da hemen x tuşuna basarsanız aşağıdaki ileti görüntülenir:

```
Aşağıdaki "I Agree" (Kabul ediyorum) seçeneğini belirleyerek,
lisans sözleşmesi ve geçerliyse,IBM dışı koşullar. Yapmazsanız
kabul ediyorum, "Kabul etmiyorum" seçeneğini belirleyin.

[ 1] I Agree (Kabul Ediyorum) ya da [ 2] I do not Agree (Kabul Ediyorum) seçeneğini
belirleyin:
```

c) Lisans sözleşmesini kabul etmek ve kuruluş dizinini seçmeye devam etmek için 1 'i seçin.

Diğer bir seçenek olarak, kuruluşu hemen sona erdirmek için 2 'yi seçin.

1 'i seçerseniz, aşağıdaki iletiye benzer bir ileti görüntülenir:

```
Ürün dosyaları için dizin girin ya da varsayılan değeri kabul etmek için boş bırakın.
Varsayılan hedef dizin H: \downloads değeridir.

Ürün dosyaları için hedef dizin?
```

4. Alma işlemi için üst dizini belirtin.

Varsayılan konum yürürlükteki dizindir.

- Ürün dosyalarını varsayılan konuma çıkarmak istiyorsanız, değer belirtmeden Enter tuşuna basın.
- Ürün dosyalarını farklı bir konuma çıkarmak istiyorsanız, dosyaları çıkarmak istediğiniz dizinin adını belirtin ve daha sonra, alma işlemini başlatmak için Enter tuşuna basın.

Belirttiğiniz dizin adı önceden var olmamalıdır; tersi durumda, alma işlemini başlattığınızda bir hata bildirilir ve hiçbir dosya kurulmaz.

Önceden varolmaması koşuluyla, belirlenen dizin yaratılır ve program dosyaları bu dizine çıkarılır. Kuruluş sırasında, belirttiğiniz üst dizin içinde wmq adlı yeni bir dizin yaratılır.

wmq dizininde aşağıdaki içerikle üç alt dizin (JavaEE, JavaSEve OSGi) oluşturulur:

JavaEE

JM 3.0 wmq.jakarta.jmsra.ivt.ear

JM 3.0 wmq.jakarta.jmsra.rar

JMS 2.0 wmq.jmsra.ivt.ear

JMS 2.0 wmq.jmsra.rar

JavaSE

Bu dizin aşağıdaki alt dizinleri ve dosyaları içerir:

JavaSE/lib

V 9.3.5 bcpkix-jdk18on.jar

LTS bcpkix-jdk15on.jar

V 9.3.5 bcprov-jdk18on.jar

LTS bcprov-jdk15on.jar

V 9.3.5 bcutil-jdk18on.jar

LTS bcutil-jdk15on.jar

JMS 2.0 com.ibm.mq.allclient.jar

JM 3.0 com.ibm.mq.jakarta.client.jar

Removed V 9.3.3 com.ibm.mq.traceControl.jar

fscontext.jar

V 9.3.3 jackson-annotations.jar

V 9.3.3 jackson-core.jar

V 9.3.3 jackson-databind.jar

jms.jar

org.json.jar

providerutil.jar

JavaSE/bin

JMSAdmin.bat

JMSAdmin

JMSAdmin.config

OSGi.

JM 3.0 com.ibm.mq.jakarta.osgi.allclient_*V.R.M.F*.jar

JM 3.0 com.ibm.mq.jakarta.osgi.allclientprereqs_*V.R.M.F*.jar

JMS 2.0 com.ibm.mq.osgi.allclient_*V.R.M.F*.jar

JMS 2.0 com.ibm.mq.osgi.allclientprereqs_*V.R.M.F*.jar

Çıkarma işlemi tamamlandığında, aşağıdaki örnekte gösterildiği gibi bir onay iletisi görüntülenir:

```
Dosyalar H: \downloads\wmq dizinine açılıyor
Tüm ürün dosyaları başarıyla çıkarıldı.
```

IBM MQ classes for JMS/Jakarta Messaging içinde izin verme

Java nesne diziselleştirme ve dizisel biçimden geri çevirme mekanizması, olası bir güvenlik riski olarak tanımlandı. IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging içinde izin verilen bazı diziselleştirme risklerine karşı koruma sağlar.

Bu görev hakkında

Dizisel biçimden geri çevirme, rasgele Java nesnelerini somutlaştırdığı için, Java nesne diziselleştirme ve dizisel biçimden geri çevirme mekanizması olası bir güvenlik riski olarak tanıtıldı; burada, çeşitli sorunlara neden olmak için kötü amaçlı olarak gönderilen veriler olabilir. Diziselleştirmenin dikkate değer bir uygulaması, rasgele nesneleri kapsüllemek ve aktarmak için diziselleştirmeyi kullanan Jakarta Messaging 3.0 ve Java Message Service 2.0 ObjectMessages içinde bulunur.

Diziselleştirme izin verme listesi, diziselleştirmenin ortaya çıkardığı bazı risklere karşı olası bir azaltma durumudur. Hangi sınıfların kapsüllenip ObjectMessagesöğesinden çıkarılabileceğini belirtik olarak belirterek, izin verme bazı diziselleştirme risklerine karşı koruma sağlar.

İlgili kavramlar

“IBM MQ classes for JMS uygulamalarının Java security manager altında çalıştırılması” sayfa 102
IBM MQ classes for JMS , Java security manager etkinken çalışabilir. Java security manager etkin durumdayken uygulamaları başarıyla çalıştırmak için Java Virtual Machine (JVM) ürününüzü uygun bir ilke yapılandırma dosyasıyla yapılandırmanız gerekir.

Kavramlara izin verme

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingürününde, JMS ObjectMessage arabiriminin somutlamasında sınıfların listelenmesi için destek vardır. Bu, Java nesne diziselleştirme ve dizisel biçimden geri çevirme mekanizmasıyla ilgili olabilecek bazı güvenlik risklerine karşı olası bir azaltma sağlar.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging içinde izin verme

Önemli:

Mümkün olduğunda, *izin listesi* terimi *beyaz liste*teriminin yerini almıştır. IBM MQ 9.0 ve sonraki yayınlarda, bu konuda sözü edilen bazı Java sistem özelliği adlarını içerir. Var olan bir yapılandırmayı değiştirmeniz gerekmez. Önceki sistem özellik adları da çalışmaya devam eder.

IBM MQ classes for JMS (JMS 2.0) V 9.3.0 V 9.3.0 ve IBM MQ classes for Jakarta Messaging (Jakarta Messaging 3.0) , JMS ObjectMessage arabiriminin somutlamasında sınıfların listelenmesini destekler.

- JMS 2.0 IBM MQ classes for JMSiçin ilgili özellik adları **`com.ibm.mq.jms.allowlist.*`**' dir.
- JM 3.0 IBM MQ classes for Jakarta Messagingiçin ilgili özellik adları şunlardır: **`com.ibm.mq.jakarta.jms.allowlist.*`**

İzin verilen liste, hangi Java sınıflarının ObjectMessage.setObject() ile diziselleştirilebileceğini ve ObjectMessage.getObject() ile dizisel biçimden geri çevrilebileceğini tanımlar.

- JMS 2.0 ObjectMessage ile izin verilen listede bulunmayan bir sınıfın eşgörünümünü diziselleştirme ya da dizisel biçimden geri çevirme girişimleri, nedeni java.io.InvalidClassException olan bir javax.jms.MessageFormatException yayınlanmasına neden olur.
- JM 3.0 ObjectMessage ile izin verilen listede bulunmayan bir sınıfın eşgörünümünü diziselleştirme ya da dizisel biçimden geri çevirme girişimleri, nedeni java.io.InvalidClassException olan bir jakarta.jms.MessageFormatException yayınlanmasına neden olur.

İzin listesi üretiliyor

Önemli: IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging bir izin listesiyle dağıtılamaz. ObjectMessages kullanılarak aktarılacak sınıfların seçimi bir uygulama tasarımı seçeneğidir ve IBM MQ bunu önleyemez.

Bu nedenle, izin verme mekanizması iki işletim kipine izin verir:

Keşif
Bu kipte düzenek, ObjectMessagesiçinde diziselleştirildiği ya da dizisel biçimden geri çevrildiği gözlemlenen tüm sınıfları raporlayan tam olarak nitelenmiş sınıf adlarının bir listesini üretir.

Zorlama
Bu kipte, düzenek izin verilmeyen sınıfları diziselleştirme ya da dizisel biçimden geri çevirme girişimlerini reddederek izin verilmesini zorlar.

Bu düzeneği kullanmak istiyorsanız, diziselleştirilmiş ve dizisel biçimden geri çevrilmiş sınıfların listesini toplamak için başlangıçta DISCOVERY kipinde çalışmanız, listeyi gözden geçirmeniz ve izin verilen listenizin temeli olarak kullanmanız gerekir. Listeyi değiştirmeden kullanmak da uygun olabilir, ancak bunu yapmaya karar vermeden önce listenin gözden geçirilmesi gerekir.

İzin verme mekanizmasının denetlenmesi

İzin verme mekanizmasını denetlemek için üç sistem özelliği vardır:

com.ibm.mq.jms.allowlist (JMS 2.0) ve com.ibm.mq.jakarta.jms.allowlist (Jakarta Messaging 3.0)
Bu özellik aşağıdaki yollardan biriyle belirtilebilir:

- İzin verilen listeyi içeren dosyanın dosya URI biçiminde (`file:`ile başlayarak) yol adı. DISCOVERY kipinde, bu dosyaya izin verme mekanizması tarafından yazılır. Dosya var olmamalıdır. Dosya varsa, düzenek üzerine yazmak yerine kural dışı durum verir. YAPTIRIM kipinde, bu dosya izin verme mekanizması tarafından okunur.
- İzin listesini oluşturan tam olarak nitelenmiş sınıf adlarından virgülle ayrılmış olarak.

Bu özellik ayarlanmazsa, izin verme listesi mekanizması etkin değildir.

Java security managerkullanıyorsanız, IBM MQ classes for JMS JAR dosyalarının bu dosyaya okuma ve yazma erişimine sahip olduğundan emin olmanız gerekir.

com.ibm.mq.jms.allowlist.discover (JMS 2.0) ve com.ibm.mq.jakarta.jms.allowlist.discover (Jakarta Messaging 3.0)

- Bu özellik ayarlanmazsa ya da false olarak ayarlanırsa, izin verme listesi mekanizması UYGULAMA kipinde çalışır.
- Bu özellik true değerine ayarlanırsa ve izin verilen liste bir dosya URI 'si olarak belirtildiyse, izin verilen liste mekanizması DISCOVERY kipinde çalışır.
- Bu özellik true olarak ayarlanırsa ve izin verilen liste sınıf adları listesi olarak belirtildiyse, izin verme listesi mekanizması uygun bir kural dışı durum verir.
- Bu özellik true değerine ayarlıysa ve izin listesi com.ibm.mq.jms.allowlist ya da com.ibm.mq.jakarta.jms.allowlist özelliğikullanılarak belirtilmediyse, izin verme listesi düzeneği etkin değildir.
- Bu özellik true olarak ayarlanırsa ve izin verilen liste dosyası zaten varsa, izin listesi mekanizması bir java.io.InvalidClassException verir ve girişler dosyaya eklenmez.

com.ibm.mq.jms.allowlist.mode (JMS 2.0) ve com.ibm.mq.jakarta.jms.allowlist.mode (Jakarta Messaging 3.0)

Bu dizgi özelliği şu üç yoldan biriyle belirtilebilir:

- Bu özellik SERIALIZE olarak ayarlanırsa, UYGULAMA kipi izin listesi geçerlilik denetimini yalnızca ObjectMessage.setObject() yönteminde gerçekleştirir.
- Bu özellik DESERIALIZE olarak ayarlanırsa, UYGULAMA kipi izin listesi geçerlilik denetimini yalnızca ObjectMessage.getObject() yönteminde gerçekleştirir.
- Bu özellik ayarlanmazsa ya da başka bir değere ayarlanırsa, UYGULAMA kipi ObjectMessage.getObject() ve ObjectMessage.setObject() yöntemlerinde izin listesi geçerlilik denetimi gerçekleştirir.

İzin listesi dosyasının biçimi

İzin verilen liste dosyası biçiminin ana özellikleri şunlardır:

- İzin verilen liste dosyası, altyapıya uygun satır sonlarıyla varsayılan altyapı dosyası kodlamasında.

  Not: Bir izin listesi dosyası kullanılıyorsa, bu dosya her zaman JVM için varsayılan dosya kodlaması kullanılarak yazılır ve okunur.

  İzin verilen liste dosyası aşağıdaki yöntemlerden biriyle oluşturulursa bu sorun olmaz:

  – z/OS z/OS üzerinde çalışan ve z/OSüzerinde de çalışan diğer bağımsız uygulamalar tarafından kullanılan bağımsız bir uygulama tarafından oluşturulur.
  – Herhangi bir altyapıda WebSphere Application Server içinde çalışan ve başka bir WebSphere Application Serveryönetim ortamı tarafından kullanılan bir uygulama tarafından oluşturulur.
  – Multi IBM MQ for Multiplatformsüzerinde çalışan ve IBM MQ for Multiplatformsüzerinde çalışan diğer bağımsız uygulamalar tarafından ya da herhangi bir altyapıda WebSphere Application Server içinde çalışan uygulamalar tarafından kullanılan bağımsız bir uygulama tarafından oluşturulur.

  Ancak WebSphere Application Server ASCII kullandığından ve bağımsız bir JVM EBCDIC kullandığından, izin verilen liste dosyası aşağıdaki yollardan biriyle oluşturulursa dosya kodlama sorunları ortaya çıkacak:

  – z/OSüzerinde üretilir, daha sonra z/OS ya da WebSphere Application Serverdışında bir platformda çalışan bağımsız uygulamalar tarafından kullanılır.
  – WebSphere Application Server ya da z/OSdışında bir platformda çalışan bağımsız bir uygulama tarafından oluşturulur ve z/OSüzerinde bağımsız bir uygulama tarafından kullanılır.
- Boş olmayan her satır, tam olarak nitelenmiş bir sınıf adı içerir. Boş satırlar yoksayılır.
- Açıklamalar eklenebilir-satırın sonuna kadar '#' karakterini izleyen her şey yoksayılır.
- Çok temel bir joker karakter mekanizması vardır:

  - '*' bir sınıf adının **son** öğesi olabilir.
  - '*', bir sınıf adının **tek** öğesiyle (yani, sınıfla, ancak paketin bir parçasıyla değil) eşleşir.

  Bu nedenle `com.ibm.mq.*`, `com.ibm.mq.MQMessage` ile eşleşecek, ancak `com.ibm.mq.jmqi.remote.api.RemoteFA`ile eşleşmeyecek.

  Joker karakter, belirtik paket adı olmayan sınıflar için varsayılan paketteki sınıflar için çalışmadığı için "*" sınıf adı reddedildi.
- Hatalı biçimlendirilmiş izin verilen liste dosyaları; örneğin, joker karakterin son öğe olmadığı `com.ibm.mq.*.Message`gibi bir girdi içeren dosyalar, java.lang.IllegalArgumentException ile sonuçlanır.
- Boş bir izin verilen liste dosyası, ObjectMessagekullanımını tamamen devre dışı bırakma etkisine sahiptir.

İzin verilen listenin virgülle ayrılmış bir liste olarak biçimi

İzin listesi için virgülle ayrılmış bir liste olarak aynı genel arama karakteri mekanizması kullanılabilir.

- '*', bir komut satırında ya da bir kabuk komut dosyasında ya da toplu iş dosyasında belirtildiyse, işletim sistemi tarafından genişletilebilir; bu nedenle özel işlem gerekebilir.
- '#' açıklama karakteri yalnızca bir dosya belirtildiğinde geçerlidir. İzin verilen liste, sınıf adlarının virgülle ayrılmış bir listesi olarak belirtilirse, işletim sisteminin ya da kabuğun işlemediği varsayılarak, birçok AIX and Linux kabuğundaki varsayılan açıklama karakteri olduğu için, bu olağan bir karakter olarak işlenir.

İzin listesi ne zaman oluyor?

İzin verme, uygulama ilk olarak bir ObjectMessage setMessage() ya da getMessage() yöntemini çalıştırdığında başlatılır.

Sistem özellikleri değerlendirilir, izin verilen liste dosyası açılır ve UYGULAMA kipinde, düzenek kullanıma hazırlandığında izin verilen sınıfların listesi yüklenir. Bu noktada, uygulamaya ilişkin IBM MQ JMS günlük dosyasına bir giriş yazılır.

Mekanizma kullanıma hazırlandığında, parametreleri değişmeyebilir. Başlatma zamanı, uygulama davranışına bağlı olduğu için kolayca tahmin edilemediğinden. Bu nedenle, sistem özelliği ayarları ve izin verilen liste dosyası içeriği, uygulamanın başlatıldığı andan itibaren düzeltilmiş olarak kabul edilmelidir. Sonuçlar garanti edilmediği için, uygulama çalışırken izin verilen dosyanın özelliklerini ya da içeriğini değiştirmeyin.

Dikkate alınacak noktalar

Java diziselleştirme mekanizmasının içerdiği riskleri azaltmaya yönelik en iyi yaklaşım, ObjectMessageyerine JSON kullanımı gibi veri aktarımına yönelik alternatif yaklaşımları araştırmaktır. Advanced Message Security (AMS) mekanizmalarının kullanılması, iletilerin güvenilir kaynaklardan geldiğinden emin olarak daha fazla güvenlik ekleyebilir.

Uygulamanızla Java security manager mekanizmasını kullanıyorsanız, aşağıdaki izinleri vermeniz gerekir:

- FilePermission (Dosya İzni), UYGULAMA kipi için okuma iznine sahip olarak, DISCOVER kipi için yazma iznine sahip olarak kullandığınız herhangi bir izin listesi dosyasında yer alır.
- JMS 2.0 **`com.ibm.mq.jms.allowlist`**, **`com.ibm.mq.jms.allowlist.discover`**ve **`com.ibm.mq.jms.allowlist.mode`** özelliklerinde PropertyPermission (okuma).
- JM 3.0 **`com.ibm.mq.jakarta.jms.allowlist`**, **`com.ibm.mq.jakarta.jms.allowlist.discover`**ve **`com.ibm.mq.jakarta.jms.allowlist.mode`** özelliklerinde PropertyPermission (okuma).

Daha fazla bilgi

İzin verilen listelerle ilgili daha fazla bilgi için bkz. “JMS ya da Jakarta Messaging izin listesini ayarlama ve kullanma” sayfa 130 ve “WebSphere Application Server içinde izin verme listesi” sayfa 132 .

İlgili kavramlar

“IBM MQ classes for JMS uygulamalarının Java security manager altında çalıştırılması” sayfa 102
IBM MQ classes for JMS , Java security manager etkinken çalışabilir. Java security manager etkin durumdayken uygulamaları başarıyla çalıştırmak için Java Virtual Machine (JVM) ürününüzü uygun bir ilke yapılandırma dosyasıyla yapılandırmanız gerekir.

JMS ya da Jakarta Messaging izin listesini ayarlama ve kullanma

Bu bilgiler, bir izin listesinin nasıl çalıştığını ve bir uygulamanın işleyebileceği ObjectMessages tiplerini içeren bir izin listesi dosyası oluşturmak için IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging içindeki işlevselliği kullanarak bir izin listesi dosyasının nasıl ayarlandığını açıklar.

Başlamadan önce

Önemli:

Mümkün olduğunda, *izin listesi* terimi *beyaz liste*teriminin yerini almıştır. IBM MQ 9.0 ve sonraki yayınlarda, bu konuda sözü edilen bazı Java sistem özelliği adlarını içerir. Var olan bir yapılandırmayı değiştirmeniz gerekmez. Önceki sistem özellik adları da çalışmaya devam eder.

Bu görevi başlatmadan önce, “Kavramlara izin verme” sayfa 126 öğesini okuduğunuzdan ve anladığınızdan emin olun.

Bu görev hakkında

JMS ve Jakarta Messaging ortak çok şey paylaştığı için, bu konuda JMS ile ilgili daha fazla başvuru, her ikisine de atıfta bulunma olarak alınabilir. Tüm farklar gerektiği şekilde vurgulanır.

İzin verme işlevini etkinleştirdiğinizde, IBM MQ classes for JMS bu işlevi aşağıdaki şekillerde kullanır:

- Bir uygulama ObjectMessagegöndermek istediğinde, aşağıdaki çağrıyı yaparak bu iletiyi iki yoldan biriyle yaratabilir:
  - Session.createObjectMessage(Diziselleştirilebilir) yöntemi, ileti içinde bulunacak nesneyi geçirir.
  - Session.createObjectMessage() yöntemi, boş bir ObjectMessageyaratmak ve daha sonra, ObjectMessageiçinde gönderilecek nesneyi saklamak için ObjectMessage.setObject(Diziselleştirilebilir) yöntemini çağırır.

  Session.createObjectMessage(Serializable) ya da ObjectMessage.setObject(Serializable) yöntemleri çağrıldığında, JMS sınıfları geçirilen nesnenin izin listesinde adı geçen bir tipte olup olmadığını denetler.

  Söz edilen bir tipse, nesne diziselleştirilip ObjectMessageiçinde saklanır. Ancak, nesne izin verilen listede olmayan bir tipse, IBM MQ classes for JMS iletiyi içeren bir JMSException yayınladı:

  ```
  JMSCC0052: Nesne diziselleştirilirken kural dışı durum oluştu:
  'java.io.InvalidClassException: < object class>; Sınıf diziselleştirilemeyebilir
  ya da '< allowlist>' izin verilen listesine dahil edilmediği için dizisel biçimden geri
  çevrildi.
  ```

  Uygulamaya geri dönün.

  Önemli: Kural dışı durum Session.createObjectMessage(Serializable) yönteminden yayınlandıysa, ObjectMessage yaratılmaz. Benzer şekilde, ObjectMessage.setObject(Serializable) yönteminden JMSException yayınlarsa, nesne ObjectMessagenesnesine eklenmez.
- Bir uygulama bir ObjectMessagealırsa, içerdiği nesneyi almak için ObjectMessage.getObject() yöntemini çağırır. Bu yöntem çağrıldığında, IBM MQ classes for JMS izin verilen listede belirtilen tipte bir nesne olup olmadığını görmek için ObjectMessageiçinde bulunan nesne tipini denetleyin.

  Bu durumda, nesne dizisel biçimden geri çevrilerek uygulamaya döndürülür. Ancak, nesne izin verilen listede olmayan bir tipse, IBM MQ classes for JMS iletiyi içeren bir JMSException yayınladı:

```
JMSCC0053: Bir ileti dizisel biçimden geri çevrilirken kural dışı durum oluştu:
'java.io.InvalidClassException: < object class>; Sınıf
diziselleştirilmiş ya da dizisel biçimden geri çevrilmiştir; diziselleştirilmiş ya da dizisel
biçimden geri çevrilmiştir.
allowlist '< allowlist>'. '.
```

Uygulamaya geri dönün.

Örneğin, uygulamanızın java.net.URI: tipinde bir nesne içeren bir ObjectMessage göndermek için aşağıdaki kodu içerdiğini varsayın:

```
java.net.URL testURL = new java.net.URL("https://www.ibm.com/");
ObjectMessage msg = session.createObjectMessage(testURL);
sender.send(msg);
```

İzin verme etkin olmadığı için, uygulama iletiyi istenen hedefe başarıyla yerleştirebiliyor.

Tek bir girdi içeren `C:\allowlist.txt` adlı bir dosya oluşturup `java.net.`URLuygulamayı Java sistem özelliği kümesiyle yeniden başlatacaksanız:

```
-Dcom.ibm.mq.jms.allowlist=file:/C:/allowlist.txt
```

izin listesi işlevi etkinleştirildi. Uygulama, izin verilen listede belirtildiği gibi, java.net.URI tipinde bir nesne içeren ObjectMessage yaratabilir ve gönderebilir.

Ancak, `allowlist.txt` dosyasını, izin listesi işlevselliği etkinleştirilmeye devam ettiği için `java.util.Calendar`tek bir girdiyi içerecek şekilde değiştirirseniz, uygulama çağrıldığında:

```
ObjectMessage msg = session.createObjectMessage(testURL);
```

IBM MQ classes for JMS izin verilen listeyi denetleyin ve java.net.URIiçin bir giriş içermediğini bulun.

Sonuç olarak, JMSCC0052 iletisini içeren bir JMSException yayınlanır.

Benzer şekilde, bu kodu kullanarak ObjectMessages alan başka bir uygulamanız olduğunu varsayalım:

```
ObjectMessage message = (ObjectMessage)receiver.receive(30000);
if (message != null) {
    Object messageBody = objectMessage.getObject();
    if (messageBody instanceof java.net.URI) {
      :    :    :    :    :    :    :
```

İzin verilmezse, uygulama herhangi bir nesne içeren ObjectMessages alabilir. Uygulama daha sonra, uygun işlemeyi gerçekleştirmeden önce nesnenin java.net.URL tipinde olup olmadığını denetler.

Uygulamayı şimdi Java sistem özelliğiyle başlatacaksanız:

```
-Dcom.ibm.mq.jms.allowlist=java.net.URL
```

set, izin verilen listeleme işlevi açık. Uygulama aradığında:

```
Object messageBody = objectMessage.getObject();
```

ObjectMessage.getObject() yöntemi yalnızca java.net.URLtipindeki nesneleri döndürür.

ObjectMessage içinde bulunan nesne bu tipte değilse, ObjectMessage.getObject() yöntemi JMSCC0053 iletisini içeren bir JMSException verir. Daha sonra uygulamanın iletiyle ne yapacağına karar vermesi gerekir; örneğin, ileti o kuyruk yöneticisi için teslim edilmeyen ileti kuyruğuna taşınabilir.

Uygulama yalnızca ObjectMessage içindeki nesne java.net.URLtipindeyse olağan bir şekilde döner.

Yordam

1. Aşağıdaki Java sistem özellikleri belirtilmiş olarak, ObjectMessagesişlemini işleyen uygulamayı çalıştırın:

```
-Dcom.ibm.mq.jms.allowlist.discover=true
-Dcom.ibm.mq.jms.allowlist=file:/<path to your allowlist file>
```

Uygulama çalıştığında IBM MQ classes for JMS , uygulamanın işlediği nesne tiplerini içeren bir dosya oluşturur.

2. Uygulama belirli bir süre boyunca ObjectMessages temsili örneğini işledikten sonra durdurun.

   İzin verilen liste dosyası, uygulamanın çalışırken işlediği ObjectMessages içinde bulunan tüm nesne tiplerinin bir listesini içerir.

   Uygulamayı yeterli bir süre çalıştırdıysanız, bu liste büyük olasılıkla uygulamanın işleyebileceğini ObjectMessages içinde bulunan tüm olası nesne tiplerini içerir.
3. Aşağıdaki sistem özelliği ayarlanmış olarak uygulamayı yeniden başlatın:

```
-Dcom.ibm.mq.jms.allowlist=file:/<path to your allowlist file>
```

   Bu, izin verilmesini etkinleştirir ve IBM MQ classes for JMS izin verilenler listesinde olmayan bir tipte bir ObjectMessage algılarsa, JMSCC0052 ya da JMSCC0053 iletisini içeren bir JMSException yayınlanır.

WebSphere Application Server içinde izin verme listesi

WebSphere Application Serveriçinde IBM MQ classes for JMS izin listesini kullanma.

Önemli:

Mümkün olduğunda, *izin listesi* terimi *beyaz liste*teriminin yerini almıştır. IBM MQ 9.0 ve sonraki yayınlarda, bu konuda sözü edilen bazı Java sistem özelliği adlarını içerir. Var olan bir yapılandırmayı değiştirmeniz gerekmez. Önceki sistem özellik adları da çalışmaya devam eder.

WebSphere Application Server kuruluşunuzda, izin verilen listelemeyi destekleyen bir IBM MQ kaynak bağdaştırıcısı sürümü olduğundan emin olmanız gerekir.

İki ürünün kullanılmasıyla ilgili ek bilgi için bkz. “IBM MQ ve WebSphere Application Server ' yi birlikte kullanma” sayfa 479 .

IBM MQ 9.0.0 Fix Pack 1 , uygun işlevselliği içerir.

Uygulama sunucusu güncellendikten sonra Java sistem özelliklerini kullanabilirsiniz:

- `-Dcom.ibm.mq.jms.allowlist`
- `-Dcom.ibm.mq.jms.allowlist.discover`

“JMS ya da Jakarta Messaging izin listesini ayarlama ve kullanma” sayfa 130içinde açıklanmıştır.

Not: Java sistem özelliklerini, uygulama sunucusunu çalıştırmak için kullanılan Java Virtual Machine üzerinde soysal JVM bağımsız değişkenleri olarak ayarlamanız ve değişikliklerin yürürlüğe girmesi için uygulama sunucusu yeniden başlatılmalıdır.

Daha fazla bilgi için Java sanal makine ayarlarında *Soysal JVM bağımsız değişkenleri* bölümüne bakın.

Özellikleri ayarlamak için, *Süreç tanımlamaları* içindeki Java Virtual Machine penceresine gidin ve uygun bağımsız değişkeni girin.

Aşağıdaki ayar:

```
-Dcom.ibm.mq.jms.allowlist=<youruserId>_MyObject
```

Uygulama sunucusunun *kullanıcıKimliği_MyObject*izin listesini kullanmasına neden olur. Yalnızca bu tipteki nesneler uygulama sunucusu tarafından işlenir.

Aşağıdaki ayarlar:

```
-Dcom.ibm.mq.jms.allowlist.discover=true
-Dcom.ibm.mq.jms.allowlist=file:C/:allowlist.txt
```

Uygulama sunucusunu *Discover* kipini kullanacak şekilde yapılandırın ve uygulama sunucusunun işlediği JMS ObjectMessagesayrıntılarını `C:\allowlist.txt` dosyasına kaydedin.

Aşağıdaki ayar:

```
-Dcom.ibm.mq.jms.allowlist=file:C/:allowlist.txt
```

Uygulama sunucusunun `C:/allowlist.txt`dosyasını yüklemesine ve izin verilen listeyi saptamak için bu dosyadaki bilgileri kullanmasına neden olur.

İlgili kavramlar

“IBM MQ classes for JMS uygulamalarının Java security manager altında çalıştırılması” sayfa 102
IBM MQ classes for JMS , Java security manager etkinken çalışabilir. Java security manager etkin durumdayken uygulamaları başarıyla çalıştırmak için Java Virtual Machine (JVM) ürününüzü uygun bir ilke yapılandırma dosyasıyla yapılandırmanız gerekir.

IBM MQ classes for JMS içinde karakter dizgisi dönüştürmeleri

IBM MQ classes for JMS , karakter dizgisi dönüştürmesi için doğrudan CharsetEncoders ve CharsetDecoders karakterlerini kullanır. Karakter dizilimi dönüştürmesi için varsayılan davranış, iki sistem özellğiyle yapılandırılabilir. Eşlenebilir karakterler içeren iletilerin işlenmesi, UnmappableCharacterişlemini ve değiştirme baytlarını ayarlamak için ileti özellikleriyle yapılandırılabilir.

IBM MQ 8.0öncesinde, IBM MQ classes for JMS içindeki dizgi dönüştürmeleri `java.nio.charset.Charset.decode(ByteBuffer)` ve `Charset.encode(CharBuffer)` yöntemleri çağrılarak gerçekleştiriliyordu.

Bu yöntemlerden birinin kullanılması, bozuk biçimli ya da çevrilemeyen verilerin varsayılan olarak değiştirilmesine (REPLACE) neden olur. Bu davranış, uygulamalardaki hataları gizleyebilir ve çevrilmiş verilerde beklenmeyen karakterlere (örneğin, ?) yol açabilir.

IBM MQ 8.0' den bu tür sorunları daha erken ve daha etkili bir şekilde saptamak için IBM MQ classes for JMS , CharsetEncoders ve CharsetDecoders öğelerini doğrudan kullanır ve bozuk biçimli ve çevrilemeyen verilerin işlenmesini belirtik olarak yapılandırır. Varsayılan davranış, uygun bir `MQException`yayınlayarak REPORT bu tür sorunlardır.

Yapılandırılıyor

UTF-16 ' dan (Javaiçinde kullanılan karakter gösterimi) UTF-8gibi bir yerel karakter kümesine çevrilmesi *kodlama*olarak adlandırılırken, tersi yönde çevirme *kod çözme*olarak adlandırılır.

Kod çözme işlemi, `CharsetDecoder`sıçin varsayılan davranışı alır ve kural dışı durum yayınlanarak hataları raporlar.

Bir ayar, hem kodlama hem de kod çözme sırasında hata işlemeyi denetlemek üzere bir `java.nio.charset.CodingErrorAction` belirtmek için kullanılır. Kodlama sırasında, yerine koyma baytını ya da baytlarını denetlemek için başka bir ayar kullanılır. Kod çözme işlemlerinde varsayılan Java yerine koyma dizgisi kullanılır.

UnmappableCharacter IBM MQ classes for JMS içindeki işlem ve değiştirme byte ayarları

IBM MQ 8.0' den UnmappableCharacterişlemini ve değiştirme baytlarını ayarlamak için aşağıdaki iki özellik kullanılabilir. Uygun değişmez tanımlamaları `com.ibm.msg.client.wmq.WMQConstants`' da yer almalıdır.

JMS_IBM_UNMAPPABLE_ACTION

Bir karakter bir kodlama ya da kod çözme işleminde eşlenemediğinde uygulanacak `CodingErrorAction` ' u ayarlar ya da alır.

Bunu aşağıdaki gibi `CodingErrorAction.{REPLACE|REPORT|IGNORE}.toString()` olarak ayarlamalısınız:

```
public static final String JMS_IBM_UNMAPPABLE_ACTION = "JMS_IBM_Unmappable_Action";
```

JMS_IBM_UNMAPPABLE_REPLASMAN
Bir karakter bir kodlama işleminde eşlenemediğinde uygulanacak yerine koyma baytlarını belirler ya da alır.

Kod çözme işlemlerinde varsayılan Java yerine koyma dizgisi kullanılır.

```
public static final String JMS_IBM_UNMAPPABLE_REPLACEMENT = "JMS_IBM_Unmappable_Replacement";
```

`JMS_IBM_UNMAPPABLE_ACTION` ve `JMS_IBM_UNMAPPABLE_REPLACEMENT` özellikleri, hedeflerde ya da iletilerde ayarlanabilir. Bir iletide ayarlanan bir değer, iletinin gönderildiği hedefte ayarlanan değeri geçersiz kılar.

`JMS_IBM_UNMAPPABLE_REPLACEMENT` ' in tek bir bayt olarak ayarlanması gerektiğini unutmayın.

Sistem varsayılanlarını ayarlamak için sistem özellikleri

IBM MQ 8.0' den, karakter dizgisi dönüşümüne ilişkin varsayılan davranışı yapılandırmak için aşağıdaki iki Java sistem özelliği kullanılabilir.

com.ibm.mq.cfg.jmqi.UnmappableCharacterAction
Kodlama ve kod çözme sırasında çevrilemeyen veriler için yapılacak işlemi belirtir. Değer `REPORT`, `REPLACE`ya da `IGNORE`olabilir.

com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement
Kodlama işleminde bir karakter eşlenemediğinde uygulanacak yerine koyma baytlarını belirler ya da alır. Kod çözme işlemlerinde varsayılan Java yerine koyma dizgisi kullanılır.

Java karakteri ile yerli byte gösterimleri arasındaki karışıklığı önlemek için,
yerel karakter takımındaki yerine koyma byte 'ını gösteren ondalık sayı olarak
`com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement` belirtmeniz gerekir.

Örneğin, yerel karakter takımı ASCII tabanlıysa (örneğin, ISO-8859-1), yerel karakter takımı EBCDIC ise 111 ise ?ondalık değeri 63 'tür.

Not: Bir MQMD ya da MQMessage nesnesinde **`unmappableAction`** ya da **`unMappableReplacement`** alanları ayarlandıysa, bu alanların değerlerinin Java sistem özelliklerinden öncelikli olduğunu unutmayın. Bu, Java sistem özellikleri tarafından belirtilen değerlerin, gerekirse her bir ileti için geçersiz kılınmasına olanak sağlar.

İlgili kavramlar
“IBM MQ classes for Java içinde karakter dizgisi dönüştürmeleri” sayfa 338
IBM MQ classes for Java , karakter dizgisi dönüştürmesi için doğrudan CharsetEncoders ve CharsetDecoders karakterlerini kullanır. Karakter dizilimi dönüştürmesi için varsayılan davranış, iki sistem özellğiyle yapılandırılabilir. Eşlenebilir karakterler içeren iletilerin işlenmesi `com.ibm.mq.MQMD`aracılığıyla yapılandırılabilir.

IBM MQ classes for JMS/Jakarta Messaging uygulamaları yazılıyor

JMS modeline kısa bir giriş yaptıktan sonra, bu bölümde IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging uygulamalarının nasıl yazılacağına ilişkin ayrıntılı yönergeler yer alır.

Bu görev hakkında

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Aynı uygulamada hem Jakarta Messaging 3.0 API hem de JMS 2.0 API ' nin kullanılması desteklenmez. Daha fazla bilgi için, bkz. JMS/Jakarta İleti Sistemi için IBM MQ sınıflarının kullanılması.

İlgili kavramlar
IBM MQ classes for Jakarta Messaging: genel bakış

JMS ve Jakarta Messaging modeli

JMS ve Jakarta Messaging modeli, Java uygulamalarının ileti alışverişi işlemlerini gerçekleştirmek için kullanabileceği bir arabirim kümesini tanımlar. IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , ileti sistemi sağlayıcılarıdır. JMS ve Jakarta Messaging nesnelerinin IBM MQ kavramlarıyla nasıl ilişkili olduğunu tanımlar. JMS ve Jakarta Messaging belirtimleri, bazı JMS ve Jakarta Messaging nesnelerinin denetlenmesini bekler.

JMS 2.0 IBM MQ 8.0 , JMS 1.1'den klasik API' yi korurken basitleştirilmiş bir API tanıtan JMS standardının JMS 2.0 sürümü için destek ekledi.

V 9.3.0 JM 3.0 V 9.3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Aynı uygulamada hem JMS 2.0 API hem de Jakarta Messaging 3.0 API ' nin kullanılması desteklenmez.

Not: Jakarta Messaging 3.0için JMS belirtimi denetimi, Oracle öğesinden Java Community Processöğesine taşınır. Ancak Oracle , Java Topluluk Sürecine taşınmayan diğer Java teknolojilerinde kullanılan "javax" adının denetimini elinde tutar. Bu nedenle Jakarta Messaging 3.0 , JMS 2.0 ile işlevsel olarak eşdeğerdir, ancak adlandırma konusunda bazı farklılıklar vardır:

- 3.0 sürümünün resmi adı Java Message Serviceyerine Jakarta Messaging .
- Paket ve sabit adlara `javax`yerine `jakarta` öneki eklenir. Örneğin, JMS 2.0 içinde bir ileti alışverişi sağlayıcısına ilk bağlantı bir `javax.jms.Connection` nesnesidir ve Jakarta Messaging 3.0 içinde bir `jakarta.jms.Connection` nesnesidir.

JMS 2.0 javax.jms paketleri JMS arabirimlerini tanımlar ve bir JMS sağlayıcısı bu arabirimleri belirli bir ileti sistemi ürünü için uygular. IBM MQ classes for JMS , IBM MQiçin JMS arabirimlerini uygulayan bir JMS sağlayıcısıdır.

JM 3.0 jakarta.jms paketleri Jakarta Messaging arabirimlerini tanımlar ve bir Jakarta Messaging sağlayıcısı bu arabirimleri belirli bir ileti sistemi ürünü için gerçekleştirir. IBM MQ classes for Jakarta Messaging , IBM MQiçin Jakarta Messaging arabirimlerini uygulayan bir Jakarta Messaging sağlayıcısıdır.

JMS ve Jakarta Messaging ortak çok şey paylaştığı için, bu konuda JMS ile ilgili daha fazla başvuru, her ikisine de atıfta bulunma olarak alınabilir. Tüm farklar gerektiği şekilde vurgulanır.

Basitleştirilmiş API

JMS 2.0 , etki alanına özgü ve etki alanından bağımsız arabirimleri JMS 1.1'den korurken basitleştirilmiş API' yi tanıttı. Basitleştirilmiş API, ileti göndermek ve almak için gereken nesne sayısını azaltır ve aşağıdaki arabirimlerden oluşur:

ConnectionFactory
ConnectionFactory , bir JMS istemcisi tarafından bağlantı yaratmak için kullanılan denetlenen bir nesnedir. Bu arabirim, klasik API 'de de kullanılır.

JMSBağlam
Bu nesne, klasik API ' nin Bağlantı ve Oturum nesnelerini birleştirir. JMSBağlam nesneleri, temeldeki bağlantı yinelenirken diğer JMSBağlam nesnelerinden yaratılabilir.

JMSÜretici
JMSüreticisi bir JMSbağlamı tarafından yaratılır ve bir kuyruğa ya da konuya ileti göndermek için kullanılır. JMSÜretici nesnesi, iletiyi göndermek için gereken nesnelerin yaratılmasına neden olur.

JMSTüketim
JMSConsumer bir JMSContext tarafından oluşturulur ve bir konudan ya da kuyruktan ileti almak için kullanılır.

Basitleştirilmiş API ' nin çeşitli etkileri vardır:

- JMSBağlam nesnesi her zaman temel bağlantıyı otomatik olarak başlatır.

- JMSÜreticiler ve JMSTüketiciler artık iletinin getBody yöntemini kullanarak tüm ileti nesnesini almak zorunda kalmadan doğrudan ileti gövdeleriyle çalışabilirler.
- JMSProducer nesnesinde, bir 'body' göndermeden önce yöntem zincirleme kullanılarak ileti özellikleri ayarlanabilir. JMSÜreticisi, iletiyi göndermek için gereken tüm nesnelerin yaratılmasını işleyecek. JMS 2.0kullanılarak özellikler ayarlanabilir ve aşağıdaki gibi bir ileti gönderilebilir:

```
context.createProducer().
setProperty("foo", "bar").
setTimeToLive(10000).
setDeliveryMode(NON_PERSISTENT).
setDisableMessageTimestamp(true).
send(dataQueue, body);
```

JMS 2.0 , iletilerin birden çok tüketici arasında paylaşılabileceği paylaşılan abonelikleri de tanıttı. Tüm JMS 1.1 abonelikleri paylaşılmayan abonelikler olarak kabul edilir.

Klasik API

Aşağıdaki liste, klasik API ' nin ana JMS arabirimlerini özetler:

Hedef
Hedef, bir uygulamanın iletileri gönderdiği ya da bir uygulamanın iletileri aldığı ya da her ikisini birden aldığı bir kaynaktır.

ConnectionFactory
ConnectionFactory nesnesi, bir bağlantıya ilişkin yapılandırma özellikleri kümesini içerir. Uygulama, bağlantı yaratmak için bir bağlantı üreticisi kullanır.

Bağlantı
Bağlantı nesnesi, bir uygulamanın bir ileti sistemi sunucusuyla olan etkin bağlantısını kapsüller. Bir uygulama, oturum yaratmak için bir bağlantı kullanır.

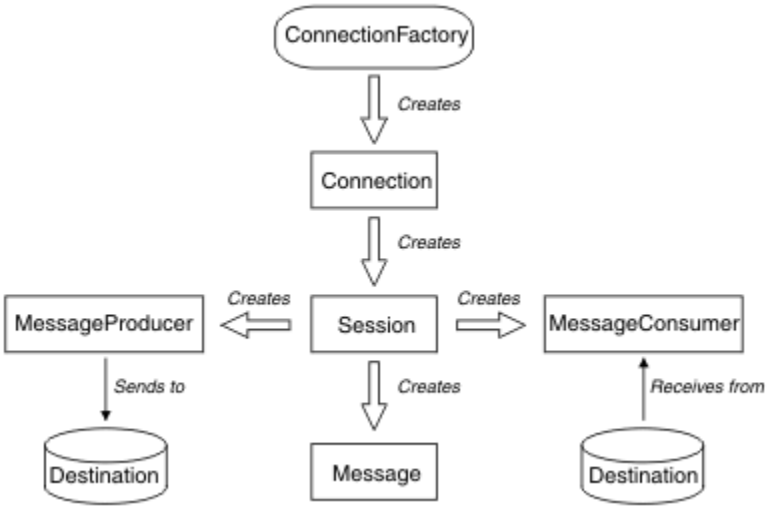
Oturum
Oturum, ileti göndermek ve almak için tek iş parçacıklı bir bağlamdır. Bir uygulama, ileti, ileti üreticileri ve ileti tüketicileri yaratmak için oturum kullanır. Bir oturum işlemden geçildi ya da işlem yapılmadı.

İleti
İleti nesnesi, bir uygulamanın gönderdiği ya da aldığı bir iletiyi içerir.

MessageProducer
Bir uygulama, bir hedefe ileti göndermek için ileti üreticiyi kullanır.

MessageConsumer
Bir uygulama, hedefe gönderilen iletileri almak için bir ileti tüketicisi kullanır.

Şekil 9 sayfa 136 içinde bu nesneler ve ilişkileri gösterilmektedir.



Şekil 9. JMS nesneleri ve ilişkileri

Bir Hedef, ConnectionFactoryya da Bağlantı nesnesi, çok iş parçacıklı bir uygulamanın farklı iş parçacıkları tarafından eşzamanlı olarak kullanılabilir, ancak bir Oturum, MessageProducerya da MessageConsumer nesnesi farklı iş parçacıkları tarafından eşzamanlı olarak kullanılamaz. Bir Oturum, MessageProducerya da MessageConsumer nesnesinin eşzamanlı olarak kullanılmamasını sağlamanın en basit yolu, her iş parçacığı için ayrı bir Oturum nesnesi yaratmaktır.

İleti sistemi etki alanları

JMS iki ileti alışverişi biçemini destekler:

- Noktadan noktaya ileti sistemi
- İleti alışverişi yayınla/abone ol

Bu ileti alışverişi stillerine *ileti alışverişi etki alanları*da denir ve bir uygulamada her iki ileti alışverişi biçemini de birleştirebilirsiniz. Noktadan noktaya iletişim etki alanında hedef bir kuyruktur ve yayınlama/abone olma etki alanında hedef bir konudur.

JMS before JMS 1.1sürümleriyle, noktadan noktaya iletişim etki alanı için programlama bir arabirim ve yöntem kümesi kullanır ve yayınlama/abone olma etki alanı için programlama başka bir küme kullanır. İki grup birbirine benzer ama ayrı. JMS 1.1' den her iki ileti sistemi etki alanını destekleyen ortak bir arabirim ve yöntem kümesini kullanabilirsiniz. Ortak arabirimler, her ileti sistemi etki alanının etki alanından bağımsız bir görünümünü sağlar. Çizelge 15 sayfa 137 içinde JMS etki alanından bağımsız arabirimler ve bunlara karşılık gelen etki alanına özgü arabirimler listelenir.

Çizelge 15. JMS etki alanından bağımsız ve etki alanına özgü arabirimler

Etki alanından bağımsız arabirimler	**Noktadan noktaya etki alanı için etki alanına özgü arabirimler**	**Yayınlama/abone olma etki alanı için etki alanına özgü arabirimler**
ConnectionFactory	QueueConnectionÜreticisi	TopicConnectionÜreticisi
Bağlantı	QueueConnection	TopicConnection
Hedef	Kuyruk	Konu
Oturum	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

JMS 2.0 IBM MQ classes for JMS 2.0 , hem önceki JMS 1.1 etki alanına özgü arabirimleri hem de JMS 2.0basitleştirilmiş API 'sini destekler. Bu nedenle IBM MQ classes for JMS 2.0 , var olan uygulamalarda yeni işlev geliştirilmesi de dahil olmak üzere var olan uygulamaların bakımı için kullanılabilir.

JM 3.0 IBM MQ classes for Jakarta Messaging 3.0 , aynı arabirimlerin Jakarta Messaging sürümlerini destekler ve yeni uygulama geliştirme için önerilir.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingsistemlerinde JMS nesneleri IBM MQ kavramlarıyla aşağıdaki şekillerde ilişkilidir:

- Bağlantı nesnesi, bağlantıyı yaratmak için kullanılan bağlantı üreticisinin özelliklerinden türetilen özelliklere sahiptir. Bu özellikler, bir uygulamanın kuyruk yöneticisine nasıl bağlanacağını denetler. Bu özelliklere örnek olarak, kuyruk yöneticisinin adı ve istemci kipinde kuyruk yöneticisine bağlanan bir uygulama için, kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi verilebilir.
- Oturum nesnesi, oturumun hareket kapsamını tanımlayan bir IBM MQ bağlantı tanıtıcısını içerir.
- MessageProducer nesnesi ve MessageConsumer nesnesi, her biri bir IBM MQ nesne tanıtıcısını kapsıyor.

IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingkullanırken, IBM MQ ' in tüm normal kuralları geçerlidir. Özellikle, bir uygulamanın uzak kuyruğa ileti gönderebildiğini, ancak yalnızca uygulamanın bağlı olduğu kuyruk yöneticisinin iyeliğindeki bir kuyruktan ileti alabildiğini unutmayın.

JMS belirtimi, ConnectionFactory ve Destination nesnelerinin denetlenmesini bekler. Bir denetimci merkezi bir havuzda yönetilen nesneler yaratır ve bunların bakımını yapar; JMS uygulaması bu nesneleri Java Naming and Directory Interface (JNDI) olanağını kullanarak alır.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingiçinde, Hedef arabiriminin somutlaması Kuyruk ve Konunun soyut bir üst sınıfıdır; dolayısıyla, Hedef somut örneği bir Kuyruk nesnesi ya da Konu nesnesidir. Etki alanından bağımsız arabirimler, bir kuyruğu ya da konuyu hedef olarak görür. Bir MessageProducer ya da MessageConsumer nesnesine ilişkin ileti alışverişi etki alanı, hedefin bir kuyruk mu, yoksa bir konu mu olduğuna göre belirlenir.

Bu nedenle, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging içinde, aşağıdaki tipteki nesneler denetlenebilirler:

- ConnectionFactory
- QueueConnectionÜreticisi
- TopicConnectionÜreticisi
- Kuyruk
- Konu
- XAConnectionFactory
- XAQueueConnectionÜreticisi
- XATopicConnectionÜreticisi

İlgili kavramlar

IBM MQ Java dil arabirimleri

“Bağlantı üreticileri ve hedefleri oluşturma ve yapılandırma” sayfa 197

Bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging uygulaması, bunları bir Java Naming and Directory Interface (JNDI) ad alanından yönetilen nesneler olarak alarak, IBM JMS uzantılarını kullanarak ya da IBM MQ JMS uzantılarını kullanarak bağlantı üreticileri ve hedefleri yaratabilir. Bir uygulama, bağlantı üreticileri ve hedeflerinin özelliklerini ayarlamak için IBM JMS uzantılarını ya da IBM MQ JMS uzantılarını da kullanabilir.

JMS ileti

JMS iletileri, bir üstbilgiden, özelliklerden ve bir gövdeden oluşur. JMS , beş tip ileti gövdesini tanımlar.

JMS iletileri aşağıdaki bölümlerden oluşur:

Üstbilgi
Tüm iletiler aynı üstbilgi alanları kümesini destekler. Üstbilgi alanları, hem istemciler hem de sağlayıcılar tarafından iletileri tanımlamak ve yönlendirmek için kullanılan değerleri içerir.

Özellikler
Her ileti, uygulama tanımlı özellik değerlerini desteklemek için yerleşik bir olanak içerir. Özellikler, uygulama tanımlı iletileri süzmek için verimli bir mekanizma sağlar.

Gövde
JMS , kullanılmakta olan ileti sistemi stillerinin çoğunu kapsayan beş tip ileti gövdesini tanımlar:

Akış
Java temel değerlerinden oluşan bir akış. Sıralı olarak doldurulur ve okunur.

Harita
Adların dizgi ve değerlerin Java ilkel tipler olduğu bir ad-değer çiftleri kümesi. Girdilere sırayla ya da adla rasgele olarak erişilebilir. Girdilerin sırası tanımlı değil.

Metin
java.lang.Stringiçeren bir ileti.

Nesne
Diziselleştirilebilir Java nesnesi içeren bir ileti

Bayt
Yorumlanmayan bayt akışı. Bu ileti tipi, bir gövdenin var olan bir ileti biçimiyle eşleşmesi için kelimenin tam anlamıyla kodlanması içindir.

JMSCorrelationID üstbilgi alanı, bir iletiyi başka bir iletiye bağlamak için kullanılır. Genellikle bir yanıt iletisini istekte bulunan iletisiyle bağlar. JMSCorrelationID , sağlayıcıya özgü bir ileti tanıtıcısını, uygulamaya özgü bir dizgiyi ya da sağlayıcıya özgü bir byte [] değerini içerebilir.

JMS içindeki ileti seçiciler
İletiler, uygulama tanımlı özellik değerlerini içerebilir. Bir uygulama, JMS sağlayıcısı süzgeç iletileri için ileti seçicilerini kullanabilir.

Bir ileti, uygulama tanımlı özellik değerlerini desteklemek için yerleşik bir olanak içerir. Bu, bir iletiye uygulamaya özgü üstbilgi alanları eklemek için bir mekanizma sağlar. Özellikler, ileti seçicileri kullanarak bir uygulamanın JMS sağlayıcısının uygulamaya özgü ölçütleri kullanarak iletileri kendi adına seçmesini ya da süzmesini sağlar. Uygulama tanımlı özellikler aşağıdaki kurallara uymalıdır:

- Özellik adları, ileti seçici tanıtıcısına ilişkin kurallara uymalıdır.
- Özellik değerleri Boole, bayt, kısa, int, long, float, double ve String olabilir.
- JMSX ve JMS_ adı önekleri ayrılmıştır.

Özellik değerleri, bir ileti gönderilmeden önce ayarlanır. Bir istemci bir ileti aldığında, ileti özellikleri salt okunur olur. Bir istemci bu noktada özellikleri ayarlamaya çalışırsa, bir MessageNotWriteableException yayınlanır. clearProperties çağrılırsa, özellikler artık hem okunabilir hem de yazılabilir.

Bir özellik değeri, ileti gövdesindeki bir değeri yineleyebilir. JMS , bir özelliğe yapılabilecek bir ilke tanımlamaz. Ancak uygulama geliştiricileri, JMS sağlayıcılarının bir ileti gövdesindeki verileri, ileti özelliklerindeki verilerden daha verimli şekilde işlediğini bilmelidirler. En iyi performans için, uygulamalar ileti özelliklerini yalnızca bir ileti üstbilgisini özelleştirmeleri gerektiğinde kullanmalıdır. Bunu yapmanın birincil nedeni, özelleştirilmiş ileti seçimini desteklemektir.

JMS ileti seçici, bir istemcinin ileti üstbilgisini kullanarak ilgilendiği iletileri belirtmesini sağlar. Yalnızca seçiciyle eşleşen üstbilgilere sahip iletiler teslim edilir.

İleti seçiciler ileti gövdesi değerlerine başvuruda bulunamaz.

Seçici, ileti üstbilgisi alanı ve özellik değerleri seçicide karşılık gelen tanıtıcılarının yerine geçtiğinde true olarak değerlendirildiğinde ileti seçici bir iletiyle eşleşir.

İleti seçici, sözdizimi SQL92 koşullu ifade sözdiziminin bir alt kümesine dayalı olan bir Dizgi 'dir. Bir ileti seçicinin değerlendirildiği sıra, öncelik düzeyinde soldan sağa doğrudur. Bu sırayı değiştirmek için parantezleri kullanabilirsiniz. Önceden tanımlanmış seçici hazır bilgileri ve işleç adları burada büyük harfle yazılır; ancak, bunlar büyük/küçük harfe duyarlı değildir.

İleti seçicinin içeriği

Bir ileti seçici şunları içerebilir:

- Hazır Bilgiler
  - Dizgi hazır bilgisi tırnak içine alınır. Çift tırnak işareti bir tırnak işaretini temsil eder. Örnekler: 'literal' ve 'literal'. Java dizgi hazır bilgileri gibi, bunlar da Unicode karakter kodlamasını kullanır.
  - Tam sayısal hazır bilgi, ondalık noktası olmayan bir sayısal değerdir; örneğin, 57, -957 ve +62. Java uzun aralığındaki sayılar desteklenir.
  - Yaklaşık sayısal hazır bilgi, bilimsel gösterimde yer alan 7E3 ya da -57.9E2gibi sayısal bir değerdir ya da 7., -95.7ya da +6.2gibi bir onlu sayısal değerdir. Java çift duyarlıklı sayı aralığındaki sayılar desteklenir.
  - Boole hazır bilgileri TRUE ve FALSE.
- Tanıtıcılar:

  - Tanıtıcı, Java harf ve Java basamaktan oluşan sınırsız uzunluklu bir dizidir; bunlardan ilki Java harfi olmalıdır. Harf, Character.isJavayönteminin true değerini döndürdüğü herhangi bir karakterdir. Buna _ ve $dahildir. Harf ya da sayı, Character.isJavaLetterOrDigit yönteminin true değerini döndürdüğü herhangi bir karakterdir.
  - Tanıtıcılar NULL, TRUE ya da FALSE adları olamaz.
  - Tanıtıcılar NOT, AND, OR, BETWEEN, LIKE, IN ya da IS olamaz.
  - Tanıtıcılar, üstbilgi alanı başvuruları ya da özellik başvurulardır.
  - Tanıtıcılar büyük ve küçük harfe duyarlıdır.
  - İleti üstbilgisi alanı başvuruları şunlarla sınırlıdır:
    - JMSDeliveryMode
    - JMS Önceliği
    - JMSMessageID
    - JMSTimestamp
    - JMSCorrelationID
    - JMSType

    JMSMessageID, JMSTimestamp, JMSCorrelationIDve JMSType değerleri boş değerli olabilir ve varsa, boş değer olarak kabul edilir.
  - JMSX ile başlayan her ad, JMStanımlı bir özellik adıdır.
  - JMS_ ile başlayan herhangi bir ad, sağlayıcıya özgü bir özellik adıdır.
  - JMS ile başlamayan her ad, uygulamaya özgü bir özellik adıdır. Bir iletide var olmayan bir özelliğe başvuru varsa, değeri NULL olur. Varsa, değeri ilgili özellik değeridir.
- Beyaz alan, Javaiçin tanımlananla aynıdır: boşluk, yatay sekme, form besleme ve çizgi sonlandırıcı.
- İfadeler:
  - Seçici, koşullu bir ifadedir. Doğru eşleşmeleri değerlendiren bir seçici; false ya da unknown olarak değerlendirilen bir seçici eşleşmiyor.
  - Aritmetik ifadeler kendilerinden, aritmetik işlemlerden, tanıtıcılardan (sayısal hazır bilgi olarak kabul edilen bir değerle) ve sayısal hazır bilgilerden oluşur.
  - Koşullu ifadeler kendilerinden, karşılaştırma işlemlerinden ve mantıksal işlemlerden oluşur.
- İfadelerin değerlendirilme sırasını ayarlamak için standart bracketing () desteklenir.
- Öncelik sırasına göre mantıksal işleçler: NOT, AND, OR.
- Karşılaştırma işleçleri: =, >, > =, <, < =, < > (eşit değil).
  - Yalnızca aynı tipte değerler karşılaştırılabilir. Bir özel durum, tam sayısal değerleri karşılaştırmak ve yaklaşık sayısal değerleri karşılaştırmak için geçerli olması. (Gereken tip dönüştürme, Java sayısal promosyon kurallarına göre tanımlanır.) Farklı tipleri karşılaştırma girişimi varsa, seçici her zaman false olur.
  - Dizgi ve Boole karşılaştırması = ve < > ile sınırlıdır. İki dizgi yalnızca aynı karakter sırasını içeriyorsa eşittir.
- Öncelik sırasına göre aritmetik işleçler:
  - +,-birli.
  - *,/, çarpma ve bölme.
  - +,-, ekleme ve çıkarma.
  - NULL değerinde aritmetik işlemler desteklenmez. Bunlar denenirse, tüm seçici her zaman false olur.
  - Aritmetik işlemler Java sayısal yükseltmesi kullanmalıdır.
- arithmetic-expr1 [NOT] BETWEEN arithmetic-expr2 ve arithmetic-expr3 karşılaştırma işleci:
  - Yaş BETWEEN 15 and 19 yaş > = 15 AND yaş < = 19 ile eşdeğerdir.

  - Yaş 15-19 ARASINDA DEğIL, < 15 OR yaş > 19 yaşına eşdeğerdir.
  - BETWEEN işleminin ifadelerinden herhangi biri NULL ise, işlemin değeri false olur. NOT BETWEEN işleminin ifadelerinden herhangi biri NULL ise, işlemin değeri true olur.
- tanıtıcı [NOT] IN (string-literal1, string-literal2, ...) karşılaştırma işleci; burada tanıtıcı bir Dizgi ya da NULL değerine sahiptir.
  - Ülke IN ('İngiltere ',' ABD ',' Fransa ')' İngiltere 'için doğru ve' Peru ' için yanlış. Bu ifade (Ülke = 'UK ') OR (Ülke = 'US') OR (Ülke = 'Fransa ') ifadesiyle eşdeğerdir.
  - Ülke NOT IN ('UK ',' US ',' France '),' UK 'için yanlış ve' Peru ' için doğru. NOT ((Ülke = 'UK ') OR (Ülke = 'US') OR (Ülke = 'Fransa ')) ifadesine eşdeğerdir.
  - IN ya da NOT IN işleminin tanıtıcısı NULL ise, işlemin değeri bilinmiyor.
- tanıtıcı [NOT] LIKE kalıp-değeri [ESCAPE kaçış-karakteri] karşılaştırma işleci; burada tanıtıcı bir dizgi değerine sahiptir. pattern-value bir dizgi hazır bilgisidir; _ herhangi bir tek karakteri,% ise herhangi bir karakter dizisini (boş sıra da içinde olmak üzere) gösterir. Diğer tüm karakterler kendilerini savunurlar. İsteğe bağlı kaçış karakteri, örüntü değerindeki _ ve% özel anlamlarına kaçış karakteri olarak kullanılan tek karakterli bir dizgi hazır bilgisidir.
  - phone LIKE '12%3' 123 ve 12993 için doğru, 1234 için yanlış.
  - LIKE 'l_se' sözcüğü "kaybet" için doğru, "gevşek" için yanlış.
  - LIKE '_ %' ESCAPE' \ ', "_foo" için doğru, "bar" için yanlış.
  - telefon NOT LIKE '12%3', 123 ve 12993 için false ve 1234 için true değeridir.
  - LIKE ya da NOT LIKE işleminin tanıtıcısı NULL ise, işlemin değeri bilinmiyor.
- tanıtıcı IS NULL karşılaştırma işleci, boş bir üstbilgi alanı değeri ya da eksik bir özellik değeri için sınamalar.
  - prop_name IS NULL (boş değerli).
- tanıtıcı IS NOT NULL karşılaştırma işleci, boş olmayan bir üstbilgi alanı değerinin ya da bir özellik değerinin varlığına ilişkin sınamalar.
  - prop_name IS NOT NULL (Boş değerli değil).

İleti seçici örneği

Aşağıdaki ileti seçici, ileti tipi otomobili, rengi mavi ve ağırlığı 2500 lbs 'den fazla olan iletileri seçer:

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

NULL özellik değerleri

Önceki listede belirtildiği gibi, özellik değerleri NULL olabilir. NULL değer içeren seçici ifadelerinin değerlendirilmesi, SQL 92 NULL anlambilim ile tanımlanır. Aşağıdaki listede bu anlambiliklere ilişkin kısa bir açıklama verilmiştir:

- SQL, NULL değerini bilinmeyen olarak kabul eder.
- Bilinmeyen bir değerle karşılaştırma ya da aritmetik her zaman bilinmeyen bir değer verir.
- IS NULL işleci, bilinmeyen bir değeri TRUE değerine dönüştürür.
- IS NOT NULL işleci, bilinmeyen bir değeri FALSE değerine dönüştürür.

JMSMessageID ve JMSCorrelationID özel davranışı

JMS için IBM MQ sınıfları, JMSMessageID ya da JMSCorrelationIDtemelinde bir kuyruktan ileti seçilirken eniyilemeler içerir.

Bir uygulama formun seçicisini belirtiyorsa:

JMSMessageID= 'ID:*message_id*'

Burada *message_id* , standart IBM MQ ileti tanıtıcısını içeren bir Dizgidir; JMS için IBM MQ sınıfları, belirtilen ileti tanıtıcısına sahip iletiyi almak için **MatchOption** MQMO_MATCH_MSG_ID ' yi kullanır.

Örneğin, 414D51207061756C745639314C545320C57C1A5F25ECE602 ileti tanıtıcısını bir kuyruktan almak için bir uygulamanın aşağıdaki ileti seçiciyi kullanması gerekir:

```
JMSMessageID='ID:414D51207061756C745639314C545320C57C1A5F25ECE602'
```

Benzer şekilde, uygulama şu biçime sahip bir seçici belirtiyorsa:

JMSCorrelationID = 'ID:*correlation_id*'

Burada *correlation_id* , standart bir IBM MQ ilinti tanıtıcısı içeren bir dizgidir; JMS 'ye ilişkin IBM MQ sınıfları, kuyruktan belirtilen ilinti tanıtıcısına sahip iletiyi almak için **MatchOption** MQMO_MATCH_CORREL_ID' yi kullanır.

Aşağıdaki örnekte, 414D51207061756C745639314C545320846E5B5F25B1CC02ilinti tanıtıcısına sahip bir iletiyi almak için bir ileti seçici kullanılır:

```
JMSCorrelationID='ID:414D51207061756C745639314C545320846E5B5F25B1CC02'
```

Bir ileti seçici *message_id* ya da *correlation_id*için tüm sıfırların değerini içeriyorsa, kuyruktaki herhangi bir iletiyle eşleşir. Örneğin, bir uygulama seçiciyi kullanıyorsa:

```
JMSMessageID='ID:000000000000000000000000000000000000000000000000'
```

kuyruktaki herhangi bir ileti eşleşme olarak kabul edilir ve uygulamaya döndürülür.

MQMO_MATCH_MSG_ID ve MQMO_MATCH_CORREL_ID **MatchOptions**ile ilgili daha fazla bilgi için bkz. MatchOptions (MQLONG).

Sınırlamalar

SQL sabit ondalık karşılaştırmayı ve aritmetiği desteklese de, JMS ileti seçicileri desteklemez. Bu nedenle, tam sayısal hazır bilgiler ondalık olmayan değerlerle sınırlıdır. Bu nedenle, yaklaşık bir sayısal değer için alternatif bir gösterim olarak ondalık içeren sayısal değerler de vardır.

SQL açıklamaları desteklenmiyor.

JMS iletilerinin IBM MQ iletileriyle eşlenmesi
IBM MQ iletileri, bir İleti Tanımlayıcı, isteğe bağlı bir MQRFH2 üstbilgisinden ve bir gövdeden oluşur. JMS iletisinin içeriği kısmen eşlenir ve kısmen bir IBM MQ iletisine kopyalanır.

Bu konuda, bu bölümün ilk kısmında açıklanan JMS ileti yapısının bir IBM MQ iletisiyle nasıl eşlendiği açıklanmaktadır. JMS ve geleneksel IBM MQ uygulamaları arasında ileti iletmek isteyen programcılar ilgilenir. Ayrıca, iki JMS uygulaması arasında iletilen iletileri (örneğin, bir IBM Integration Bus uygulamasında) işlemek isteyen kişileri de ilgilendiriyor.

Bir uygulama bir aracıya gerçek zamanlı bağlantı kullanıyorsa bu kısım geçerli değildir. Bir uygulama gerçek zamanlı bağlantı kullandığında, tüm iletişim doğrudan TCP/IP üzerinden gerçekleştirilir; IBM MQ kuyrukları ya da iletileri dahil değildir.

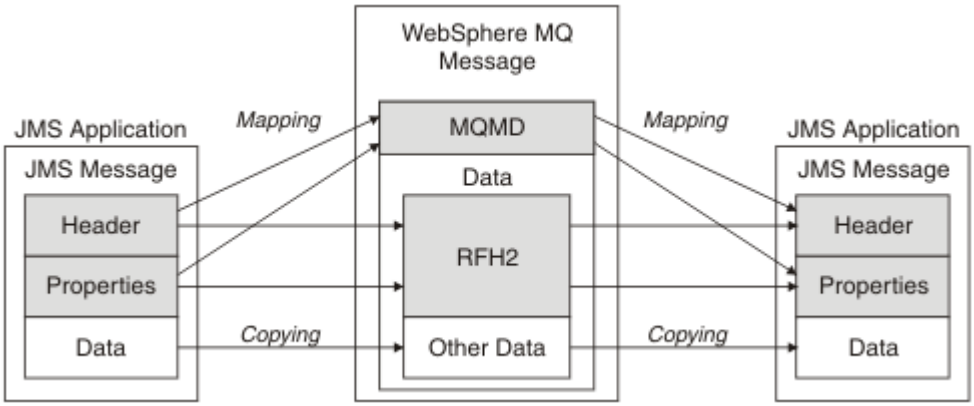
IBM MQ iletileri üç bileşenden oluşur:

- IBM MQ Message Descriptor (MQMD)
- IBM MQ MQRFH2 üstbilgisi
- İleti gövdesi.

MQRFH2 isteğe bağlıdır ve giden iletiye eklenmesi JMS Hedef sınıfındaki TARGCLIENT işareti tarafından yönetilir. Bu işareti IBM MQ JMS yönetim aracını kullanarak ayarlayabilirsiniz. MQRFH2 , JMS' ye özgü bilgileri taşıdığından, gönderen, alıcı hedefin bir JMS uygulaması olduğunu bildiğinde bunu her zaman iletiye ekleyin. Olağan durumda,JMS dışı bir uygulamaya doğrudan ileti gönderirken MQRFH2 öğesini atlayın. Bunun nedeni, böyle bir uygulamanın IBM MQ iletisinde MQRFH2 beklememesi olabilir.

Gelen bir iletinin MQRFH2 üstbilgisi yoksa, varsayılan olarak iletinin JMSReplyTo üstbilgi alanından türetilen Kuyruk ya da Konu nesnesi, kuyruğa ya da konuya gönderilen bir yanıt iletisinin de MQRFH2 üstbilgisine sahip olmaması için bu işareti ayarlamıştır. Bağlantı üreticisinin TARGCLIENTMATCHING özelliğini NO olarak ayarlayarak, özgün iletinin MQRFH2 üstbilgisi varsa, yanıt iletisine MQRFH2 üstbilgisi ekleme davranışını kapatabilirsiniz.

Şekil 10 sayfa 143 , bir JMS iletisinin yapısının IBM MQ iletisine nasıl dönüştürüleceğini ve yeniden nasıl geri döneceğini gösterir:



Şekil 10. MQRFH2 üstbilgisi kullanılarak iletilerin JMS ve IBM MQ arasında nasıl dönüştürüldüğü

Yapılar iki şekilde dönüştürülür:

Eşleme
MQMD, JMS alanıyla eşdeğer bir alan içeriyorsa, JMS alanı MQMD alanıyla eşlenir. Bir JMS uygulamasınınJMS dışı bir uygulamayla iletişim kurarken bu alanları alması ya da ayarlanması gerekebileceğinden, ek MQMD alanları JMS özellikleri olarak gösterilir.

Kopyalama
MQMD eşdeğeri yoksa, bir JMS üstbilgi alanı ya da özelliği MQRFH2içindeki bir alan olarak dönüştürülebilir.

MQRFH2 üstbilgisi ve JMS
Bu konu derlemi, ileti içeriğiyle ilişkilendirilmiş JMS' ye özgü verileri taşıyan MQRFH Sürüm 2 üstbilgisini açıklar. MQRFH Sürüm 2 üstbilgisi genişletilebilir ve JMSile doğrudan ilişkili olmayan ek bilgileri de taşıyabilir. Ancak bu bölüm yalnızca JMStarafından kullanımını kapsar.Tam açıklama için bkz. MQRFH2 -Kurallar ve biçimlendirme üstbilgisi 2.

Üstbilginin iki bölümü vardır: Sabit bir bölüm ve değişken bir bölüm.

Sabit bölüm
Sabit bölüm, *standart* IBM MQ üstbilgi kalıbında modellenir ve aşağıdaki alanlardan oluşur:

StrucId (MQCHAR4)
Yapı tanıtıcısı.

MQRFH_STRUC_ID (değer: "RFH ") olmalıdır (ilk değer).

MQRFH_STRUC_ID_ARRAY (değer: "R", "F", "H", " ") tanımlı.

Sürüm (MQLONG)
Yapı sürümü numarası.

MQRFH_VERSION_2 (değer: 2) (başlangıç değeri) olmalıdır.

StrucLength (MQLONG)
NameValueVeri alanları da içinde olmak üzere toplam MQRFH2uzunluğu.

StrucLength değerine ayarlanan değer 4 'ün katı olmalıdır (bunu başarmak için NameValueVeri alanlarındaki veriler boşluk karakterleriyle doldurulabilir).

Kodlama (MQLONG)
Veri kodlaması.

MQRFH2 ' yi izleyen iletinin bölümündeki herhangi bir sayısal verinin kodlanması (sonraki üstbilgi ya da bu üstbilgiyi izleyen ileti verileri).

CodedCharSetId (MQLONG)
Kodlanmış karakter takımı tanıtıcısı.

MQRFH2 ' yi (sonraki üstbilgi ya da bu üstbilgiyi izleyen ileti verileri) izleyen iletinin bölümündeki herhangi bir karakter verisinin gösterimi.

Biçim (MQCHAR8)
Biçim adı.

MQRFH2' yi izleyen ileti bölümünün biçim adı.

İşaretler (MQLONG)
Bayraklar.

MQRFH_NO_FLAGS = 0. İşaretler ayarlanmadı.

NameValueCCSID (MQLONG)
Bu üstbilgide bulunan NameValueveri karakter dizilimleri için kodlanmış karakter takımı tanıtıcısı (CCSID). NameValueVerileri, üstbilgide (StrucID ve Format) bulunan diğer karakter dizilimlerinden farklı olan bir karakter takımındaki kodlanabilir.

NameValueCCSID 'si 2 baytlık Unicode CCSID 'si (1200, 13488 ya da 17584) ise, Unicode 'un bayt sırası, MQRFH2' deki sayısal alanların bayt sıralaması ile aynıdır. (Örneğin, Sürüm, StrucLengthve NameValueCCSID değerinin kendisi.)

Çizelge 16. NameValueCCSID alanı için olası değerler

CCSID	**Anlamı**
1200	UTF-16, desteklenen en son Unicode sürümü
13488	UTF-16, Unicode sürümü 2.0 altkümesi
17584	UTF-16, Unicode sürümü 3.0 altkümesi (Euro simgesini içerir)
1208	UTF-8, desteklenen en son Unicode sürümü

Değişken bölümü
Değişken kısım sabit kısmı izler. Değişken kısmı, MQRFH2 klasörlerinin değişken sayısını içerir. Her klasör, değişken sayıda öğe ya da özellik içerir. Klasörler grubuyla ilgili özellikler. JMS tarafından oluşturulan MQRFH2 üstbilgileri aşağıdaki klasörlerden herhangi birini içerebilir:

`mcd` klasörü

`mcd` , iletinin biçimini açıklayan özellikleri içerir. Örneğin, ileti hizmeti etki alanı `Msd` özelliği bir JMS iletisini `JMSTextMessage`, `JMSBytesMessage`, `JMSStreamMessage`, `JMSMapMessage`, `JMSObjectMessage`eya da boş değer olarak tanımlar.

`mcd` klasörü, `MQRFH2`içeren bir JMS iletisinde her zaman bulunur.

Bu ileti, her bana yılında IBM Integration Bus' dan gönderilen bir `MQRFH2` iletisini içeren iletide bulunur. Bir iletinin etki alanını, biçimini, tipini ve ileti kümesini açıklar.

Çizelge 17. `mcd` özellik adı, eşanlamlı, veri tipi ve klasör

Özellik eşanlamlısı	**Özellik adı**	**Veri tipi**	**Klasör**
	`mcd.Msd`	`string`	`<mcd><Msd>`*`messageDomain`*`</Msd></mcd>`

Çizelge 17. mcd özellik adı, eşanlamlı, veri tipi ve klasör (devamı var)			
Özellik eşanlamlısı	**Özellik adı**	**Veri tipi**	**Klasör**
	mcd.Set	string	<mcd><Set>*messageDomain*</Set></mcd>
	mcd.Type	string	<mcd><Type>*messageDomain*</Type></mcd>
	mcd.Fmt	string	<mcd><Fmt>*messageDomain*</Fmt></mcd>

mcd klasörüne kendi özelliklerinizi eklemeyin.

jms klasörü

jms , MQMDiçinde tam olarak ifade edilemeyen JMS üstbilgi alanlarını ve JMSX özelliklerini içerir. jms klasörü her zaman bir JMS MQRFH2içinde bulunur.

usr klasörü

usr , iletiyle ilişkili uygulama tanımlı JMS özelliklerini içerir. usr klasörü, yalnızca bir uygulama uygulama tanımlı bir özellik ayarladıysa bulunur.

mqext klasörü

mqext aşağıdaki özellik tiplerini içerir:

- Yalnızca WebSphere Application Servertarafından kullanılan özellikler.
- İletilerin gecikmeli teslimiyle ilgili özellikler.

Uygulama, IBM tarafından tanımlanan özelliklerden en az birini ya da kullanılan teslim gecikmesini ayarladıysa klasör vardır.

Çizelge 18. mqext özellik adı, eşanlamlı, veri tipi ve klasör			
Özellik eşanlamlısı	**Özellik adı**	**Veri tipi**	**Klasör**
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>*wrmCorrelator*</Wrm></mqext>
JMSDeliveryTime	mqext.Dlt	i8	<mqext><Dlt>*DeliveryTime*</Dlt></mqext>
JMSDeliveryDelay	mqext.Dly	i8	<mqext><Dly>*DeliveryTime*</Dly></mqext>

mqext klasörüne kendi özelliklerinizi eklemeyin.

mqps klasörü

mqps , yalnızca IBM MQ yayınlama/abone olma tarafından kullanılan özellikleri içerir. Klasör, yalnızca uygulama, tümleşik yayınlama/abone olma özelliklerinden en az birini ayarladıysa bulunur.

Çizelge 19. mqps özellik adı, eşanlamlı, veri tipi ve klasör			
Özellik eşanlamlısı	**Özellik adı**	**Veri tipi**	**Klasör**
MQTopicString	mqps.Top	string	<mqps><Top>*topicString*</Top></mqps>

Çizelge 19. mqps özellik adı, eşanlamlı, veri tipi ve klasör (devamı var)			
Özellik eşanlamlısı	**Özellik adı**	**Veri tipi**	**Klasör**
MQSubUserVerileri	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>

mqps klasörüne kendi özelliklerinizi eklemeyin.

Çizelge 20 sayfa 146 , özellik adlarının tam listesini gösterir.

Çizelge 20. MQRFH2 tarafından kullanılan klasörler ve özellikler JMS				
JMS alan adı	**Java tip**	**MQRFH2 klasör adı**	**Özellik adı**	**Tip/Değerler**
JMSDestination (JMS Hedefi)	Hedef	JMS	Dst	dizgi
JMSExpiration	uzun	JMS	ÜS	i8
JMS Önceliği	int	JMS	Pri.	i4
JMSDeliveryMode	int	JMS	Dlv.	i4
JMSCorrelationID	Dizgi	JMS	CID	dizgi
JMSReplyTo	Hedef	JMS	Rto.	dizgi
JMSTimestamp	uzun	JMS	Tms	i8
JMSType	Dizgi	MCD	Tip, Küme, Fmt	dizgi
JMSXGroupID	Dizgi	JMS	GID	dizgi
JMSXGroupSeq	int	JMS	Sıra	i4
xxx (kullanıcı tanımlı)	Herhangi Biri	USR	xxx	Herhangi Biri
		MCD	MSD	jms_none jms_text jms_bytes jms_map jms_stream jms_nesnesi

NameValueUzunluğu (MQLONG)
Bu uzunluk alanını hemen izleyen NameValueveri dizgisinin bayt cinsinden uzunluğu (kendi uzunluğunu içermez).

NameValueVerileri (MQCHARn)
Uzunluğu bayt olarak önceki NameValueUzunluk alanı tarafından verilen tek bir karakter dizgisi. Bir özellikler sırasını tutan bir klasör içerir. Her özellik, adı klasör adı olan bir XML öğesi içinde bulunan bir ad/tip/değer üçlüsüdür:

```
<foldername>
triplet1 triplet2 .....  tripletn </foldername>
```

Kapanış `</foldername>` etiketini doldurma karakterleri olarak boşluklar izleyebilir. Her üçlü, XML benzeri bir sözdizimi kullanılarak kodlanır:

```
<name dt='datatype'>value</name>
```

Veri tipi önceden tanımlı olduğundan, `dt='datatype'` öğesi isteğe bağlıdır ve birçok özellik için atlanır. İçerildiyse, `dt=` etiketinden önce bir ya da daha çok boşluk karakteri eklenmelidir.

name
Özelliğin adıdır; bkz. Çizelge 20 sayfa 146.

datatype
Katlama işleminden sonra, Çizelge 21 sayfa 147içinde listelenen veri tiplerinden biriyle eşleşmelidir.

value
Çizelge 21 sayfa 147içindeki tanımlar kullanılarak, iletilecek değerin dizgi gösterimidir.

Boş değer, aşağıdaki sözdizimi kullanılarak kodlanır:

```
<name dt='datatype' xsi:nil='true'></name>
```

`xsi:nil='false'`kullanmayın.

Çizelge 21. Özellik veri tipleri

Veri tipi	Tanım
dizgi	< ve & dışında herhangi bir karakter sırası
boole	`0` ya da `1` karakteri (`0` = false, `1` = true)
bin.hex	Sekizliyi gösteren onaltılı basamaklar
i1	İsteğe bağlı işaretli (kesir ya da üs olmadan) `0..9`rakamları kullanılarak ifade edilen bir sayı. -128-127 aralığında olmalıdır (bu değerler de içinde olmak üzere)
i2	İsteğe bağlı işaretli (kesir ya da üs olmadan) `0..9`rakamları kullanılarak ifade edilen bir sayı. -32768-32767 (bu değerler de içinde olmak üzere) aralığında olmalıdır
i4	İsteğe bağlı işaretli (kesir ya da üs olmadan) `0..9`rakamları kullanılarak ifade edilen bir sayı. -2147483648-2147483647 (bu değerler de içinde olmak üzere) aralığında olmalıdır
i8	İsteğe bağlı işaretli (kesir ya da üs olmadan) `0..9`rakamları kullanılarak ifade edilen bir sayı. -9223372036854775808-9223372036854775807 aralığında olmalıdır (bu değerler de içinde olmak üzere)
int	İsteğe bağlı işaretli (kesir ya da üs olmadan) `0..9`rakamları kullanılarak ifade edilen bir sayı. i8ile aynı aralıkta olmalıdır. Gönderen belirli bir duyarlığı özellikle ilişkilendirmek istemiyorsa, bu, i * tiplerinden birinin yerine kullanılabilir.

Çizelge 21. Özellik veri tipleri (devamı var)	
Veri tipi	**Tanım**
r4	Kayar noktalı sayı, kayar noktalı sayı < = 3.40282347E+38, > = 1.175E-37 ifade edilen rakamlar `0..9`, isteğe bağlı işaret, isteğe bağlı kesirli rakamlar, isteğe bağlı üstel
r8	Kayar noktalı sayı, kayar noktalı sayı < = 1.7976931348623E+308, > = 2.225E-307 rakamlar kullanılarak ifade edilir `0..9`, isteğe bağlı işaret, isteğe bağlı kesirli basamaklar, isteğe bağlı kesirli basamaklar

Bir dizgi değeri boşluk içerebilir. Bir dizgi değerinde aşağıdaki çıkış sıralarını kullanmalısınız:

- & karakteri için `&`
- < karakteri için `<`

Aşağıdaki çıkış sıralarını kullanabilirsiniz, ancak bunlar gerekli değildir:

- > karakteri için `>`
- ' karakteri için `&apos;`
- " karakteri için `&quot;`

İlgili MQMD alanlarını içeren JMS alanları ve özellikleri
Bu çizelgeler, JMS üstbilgi alanları, JMS özellikleri ve JMS sağlayıcısına özgü özelliklere eşdeğer MQMD alanlarını gösterir.

Çizelge 22 sayfa 148 içinde JMS üstbilgi alanları ve Çizelge 23 sayfa 148 içinde doğrudan MQMD alanlarıyla eşlenen JMS özellikleri listelenir. Çizelge 24 sayfa 149 içinde sağlayıcıya özgü özellikler ve bunların eşlendiği MQMD alanları listelenir.

Çizelge 22. MQMD alanlarıyla eşleyen JMS üstbilgi alanları			
JMS üstbilgi alanı	**Java tip**	**MQMD alanı**	**C tipi**
JMSDeliveryMode	int	Kalıcılık	MQLONG
JMSExpiration	uzun	Son kullanma tarihi	MQLONG
JMS Önceliği	int	Öncelik	MQLONG
JMSMessageID	Dizgi	MsgID	MQBYTE24
JMSTimestamp	uzun	PutDate PutTime	MQCHAR8 MQCHAR8
JMSCorrelationID	Dizgi	CorrelId	MQBYTE24

Çizelge 23. JMS özellikleri MQMD alanlarıyla eşleme			
JMS özellik	**Java tip**	**MQMD alanı**	**C tipi**
JMSXUserID	Dizgi	UserIdentifier	MQCHAR12
JMSXAppID	Dizgi	PutApplAdı	MQCHAR28
JMSXDeliveryCount	int	BackoutCount	MQLONG
JMSXGroupID	Dizgi	GroupId	MQBYTE24
JMSXGroupSeq	int	MsgSeqNumarası	MQLONG

Çizelge 24. MQMD alanlarıyla JMS sağlayıcıya özgü özellikler eşlemesi			
JMS sağlayıcıya özgü özellik	**Java tip**	**MQMD alanı**	**C tipi**
JMS_IBM_Report_Exception	int	Rapor	MQLONG
JMS_IBM_Report_Expiration	int	Rapor	MQLONG
JMS_IBM_Report_COA	int	Rapor	MQLONG
JMS_IBM_Report_COD	int	Rapor	MQLONG
JMS_IBM_Report_PAN	int	Rapor	MQLONG
JMS_IBM_Report_NAN	int	Rapor	MQLONG
JMS_IBM_Report_Pass_Msg_ID	int	Rapor	MQLONG
JMS_IBM_Report_Pass_Correl_ID	int	Rapor	MQLONG
JMS_IBM_Report_Discard_Msg	int	Rapor	MQLONG
JMS_IBM_MsgType	int	MsgType	MQLONG
JMS_IBM_Feedback	int	Geri bildirim	MQLONG
JMS_IBM_Biçimi	Dizgi	Biçim"1" sayfa 149	MQCHAR8
JMS_IBM_PutApplTipi	int	PutApplTipi	MQLONG
JMS_IBM_Kodlaması	int	Kodlama	MQLONG
JMS_IBM_Character_Set	Dizgi	CodedCharacterSetId "2" sayfa 149	MQLONG
JMS_IBM_PutDate	Dizgi	PutDate	MQCHAR8
JMS_IBM_PutTime	Dizgi	PutTime	MQCHAR8
JMS_IBM_Last_Msg_In_Group	boole	MsgFlags	MQLONG

Not:

1. JMS_IBM_Format, ileti gövdesinin biçimini temsil eder. Bu, iletinin JMS_IBM_Format özelliğini ayarlayan uygulama tarafından tanımlanabilir (8 karakterlik bir sınır olduğunu unutmayın) ya da varsayılan olarak JMS ileti tipine uygun ileti gövdesinin IBM MQ biçimine ayarlanabilir. JMS_IBM_Format, yalnızca ileti RFH ya da RFH2 kısımları içermezse MQMD Format alanıyla eşlenir. Tipik bir iletide, ileti gövdesinden hemen önce gelen RFH2 ' nin Biçim alanıyla eşlenir.
2. JMS_IBM_Character_Set özellik değeri, sayısal CodedCharacterSetId değerinin Java karakter kümesi eşdeğerini içeren bir Dizgi değeridir. MQMD alanı CodedCharacterSetId , JMS_IBM_Character_Set özelliği tarafından belirtilen Java karakter kümesi dizgisinin eşdeğerini içeren sayısal bir değerdir.

JMS alanlarının IBM MQ alanlarıyla (giden iletiler) eşlenmesi
Bu çizelgeler, JMS üstbilgi ve özellik alanlarının send () ya da publish () sırasında MQMD ve MQRFH2 alanlarıyla nasıl eşlendiğini gösterir.

Çizelge 25 sayfa 150 , JMS üstbilgi alanlarının send () ya da publish () zamanındaki MQMD/RFH2 alanlarıyla nasıl eşlendiğini gösterir. Çizelge 26 sayfa 150 , JMS özelliklerinin send () ya da publish () zamanındaki MQMD/RFH2 alanlarıyla nasıl eşlendiğini gösterir. Çizelge 27 sayfa 151 , JMS sağlayıcıya özgü özelliklerin send () ya da publish () sırasında MQMD alanlarıyla nasıl eşlendiğini gösterir.

İleti Nesnesi Tarafından Ayarla işaretli alanlar için, iletilen değer, JMS iletisinde send () ya da publish () işleminden hemen önce tutulan değerdir. JMS iletisindeki değer, işlem tarafından değiştirilmeden bırakılır.

Gönderme yöntemiyle ayarlanan alanlar için, send () ya da publish () işlemi gerçekleştirildiğinde (JMS iletisinde tutulan herhangi bir değer yoksayılır) bir değer atanır. JMS iletisindeki değer, kullanılan değeri gösterecek şekilde güncellenir.

Yalnızca Al olarak işaretli alanlar iletilmez ve send () ya da publish () ile iletide değişmeden bırakılır.

Çizelge 25. Giden ileti alanı eşlemesi

JMS üstbilgi alanı adı	**İletim için kullanılan MQMD alanı**	**Üstbilgi**	**Ayarlayan**
JMSDestination (JMS Hedefi)		MQRFH2	Gönderme Yöntemi
JMSDeliveryMode	Kalıcılık	MQRFH2	Gönderme Yöntemi
JMSExpiration	Son kullanma tarihi	MQRFH2	Gönderme Yöntemi
JMS Önceliği	Öncelik	MQRFH2	Gönderme Yöntemi
JMSMessageID	MsgID		Gönderme Yöntemi
JMSTimestamp	PutDate/PutTime		Gönderme Yöntemi
JMSCorrelationID	CorrelId	MQRFH2	İleti Nesnesi
JMSReplyTo	ReplyToQ/ReplyToQMgr	MQRFH2	İleti Nesnesi
JMSType		MQRFH2	İleti Nesnesi
Yeniden Teslim edilen JMS			Yalnızca alma

Not:

1. MQMD alanı CodedCharacterSetId , JMS_IBM_Character_Set özelliği tarafından belirtilen Java karakter kümesi dizgisinin eşdeğerini içeren sayısal bir değerdir.

Çizelge 26. Giden ileti JMS özellik eşlemesi

JMS özellik adı	**İletim için kullanılan MQMD alanı**	**Üstbilgi**	**Ayarlayan**
JMSXUserID	UserIdentifier		Gönderme Yöntemi
JMSXAppID	PutApplAdı		Gönderme Yöntemi
JMSXDeliveryCount			Yalnızca alma
JMSXGroupID	GroupId	MQRFH2	İleti Nesnesi
JMSXGroupSeq	MsgSeqNumarası	MQRFH2	İleti Nesnesi

Not:

Bu özellikler JMS belirtimi tarafından salt okunur olarak tanımlanır ve (bazı durumlarda isteğe bağlı olarak) JMS sağlayıcısı tarafından ayarlanır.

IBM MQ classes for JMS içinde bu özelliklerden ikisi uygulama tarafından geçersiz kılınabilir. Bunu yapmak için, aşağıdaki özellikleri ayarlayarak hedefin uygun şekilde yapılandırıldığından emin olun:

1. `WMQConstants.WMQ_MQMD_MESSAGE_CONTEXT` özelliğini `WMQConstants.WMQ_MDCTX_SET_ALL_CONTEXT`olarak ayarlayın.
2. `WMQConstants.WMQ_MQMD_WRITE_ENABLED` özelliğini `true`olarak ayarlayın.

Uygulama aşağıdaki özellikleri geçersiz kılabilir:

JMSXAppID
Bu özellik, iletide `WMQConstants.JMS_IBM_MQMD_PUTAPPLNAME` özelliği ayarlanarak geçersiz kılınabilir-değer bir Java Dizesi olmalıdır.

JMSXGroupID
Bu özellik, iletide `WMQConstants.JMS_IBM_MQMD_GROUPID` özelliği ayarlanarak geçersiz kılınabilir-değer bir bayt dizisi olmalıdır.

Çizelge 27. Giden ileti JMS sağlayıcıya özgü özellik eşlemesi

JMS sağlayıcıya özgü özellik adı	İletim için kullanılan MQMD alanı	Üstbilgi	Ayarlayan
JMS_IBM_Report_Exception	Rapor		İleti Nesnesi
JMS_IBM_Report_Expiration	Rapor		İleti Nesnesi
JMS_IBM_Report_COA/COD	Rapor		İleti Nesnesi
JMS_IBM_Report_NAN/PAN	Rapor		İleti Nesnesi
JMS_IBM_Report_Pass_Msg_ID	Rapor		İleti Nesnesi
JMS_IBM_Report_Pass_Correl_ID	Rapor		İleti Nesnesi
JMS_IBM_Report_Discard_Msg	Rapor		İleti Nesnesi
JMS_IBM_MsgType	MsgType		İleti Nesnesi
JMS_IBM_Feedback	Geri bildirim		İleti Nesnesi
JMS_IBM_Biçimi	Biçim		İleti Nesnesi
JMS_IBM_PutApplTipi	PutApplTipi		Gönderme Yöntemi
JMS_IBM_Kodlaması	Kodlama		İleti Nesnesi
JMS_IBM_Character_Set	CodedCharacterSetId		İleti Nesnesi
JMS_IBM_PutDate	PutDate		Gönderme Yöntemi
JMS_IBM_PutTime	PutTime		Gönderme Yöntemi
JMS_IBM_Last_Msg_In_Group	MsgFlags		İleti Nesnesi

send () ya da publish () sırasında JMS üstbilgi alanlarının eşlenmesi
Bu notlar, send () ya da publish () yolundaki JMS alanlarının eşlemesiyle ilgilidir.

JMSDestination- MQRFH2
Bu, bir alıcı JMS ' in eşdeğer bir hedef nesneyi yeniden oluşturması için hedef nesnenin ayırt edici özelliklerini diziselleştiren bir dizgi olarak saklanır. MQRFH2 alanı URI olarak kodlanmıştır (URI gösteriminin ayrıntıları için bkz. “Birörnek kaynak tanıtıcıları (URI ' ler)” sayfa 214).

JMSReplyTo - MQMD.ReplyToQ, ReplyToQMgr, MQRFH2
Kuyruk adı MQMD.ReplyToQ alanı ve kuyruk yöneticisi adı ReplyToQMgr alanlarına kopyalanır. Hedef uzantı bilgileri (hedef nesnede tutulan diğer yararlı ayrıntılar) MQRFH2 alanına kopyalanır. MQRFH2 alanı bir URI olarak kodlanır (URI gösteriminin ayrıntıları için bkz. “Birörnek kaynak tanıtıcıları (URI ' ler)” sayfa 214).

JMSDeliveryMode - MQMD.Persistence

JMSDeliveryMode değeri, Hedef Nesne onu geçersiz kılmadıkça, send () ya da publish () Yöntemi ya da MessageProducertarafından ayarlanır. JMSDeliveryMode değeri MQMD.Persistence alanı:

- JMS değeri PERSISTENT, MQPER_PERSISTENT ile eşdeğerdir
- JMS değeri NON_PERSISTENT, MQPER_NOT_PERSISTENT ile eşdeğerdir

MQQueue persistence özelliği WMQConstants.WMQ_PER_QDEFolarak ayarlanmazsa, teslim kipi değeri MQRFH2içinde de kodlanır.

MQMD.Expiry, MQRFH2

JMSExpiration, süre bitimini (yürürlükteki saat ve yaşam süresi toplamı) saklar, MQMD ise yaşam süresini saklar. Ayrıca, JMSExpiration milisaniye olarak, ancak MQMD.Expiry saniyenin onda birindedir.

- send () yöntemi sınırsız bir yaşam süresi ayarlarsa, MQMD.Expiry MQEI_UNLIMITED olarak ayarlandı ve MQRFH2içinde hiçbir JMSExpiration kodlanmadı.
- send () yöntemi 214748364.7 saniyeden (yaklaşık 7 yıl) daha kısa bir süre için yaşam süresi ayarlarsa, yaşam süresi MQMD.Expiryve süre bitimi (milisaniye), MQRFH2içinde bir i8 değeri olarak kodlanır.
- send () yöntemi bir süreyi 214748364.7 saniyeden uzun bir süre için ayarlarsa, MQMD.Expiry MQEI_UNLIMITED olarak ayarlandı. Gerçek süre bitim süresi (milisaniye), MQRFH2içinde i8 değeri olarak kodlanır.

MQMD.Priority

JMSPriority değerini (0-9) MQMD öncelik değeriyle (0-9) doğrudan eşleyin. JMSPriority varsayılan olmayan bir değere ayarlanırsa, öncelik düzeyi MQRFH2içinde de kodlanır.

JMSMessageID kaynak MQMD.MessageID

JMS ' den gönderilen tüm iletilerin, IBM MQtarafından atanan benzersiz ileti tanıtıcıları var. Atanan değer MQMD.MessageId alanı ve JMSMessageID alanında uygulamaya geri aktarılır. IBM MQ messageId 24 baytlık bir ikili değerdir, JMSMessageID ise bir dizedir. JMSMessageID , öneki şu karakterler olan 48 onaltılı karakterden oluşan bir sıraya dönüştürülen ikili messageId değerinden oluşur: JMS , ileti tanıtıcılarının üretilmesini geçersiz kılmak için ayarlanabilecek bir ipucu sağlar. Bu ipucu yoksayılır ve her durumda benzersiz bir tanıtıcı atanır. Bir send () üzerine yazılmadan önce JMSMessageID alanında ayarlanan herhangi bir değer.

MQMD.MessageID, bunu “IBM MQ classes for JMS uygulamasından ileti tanımlayıcısının okunması ve yazılması” sayfa 235içinde açıklanan IBM MQ JMS uzantılarından biriyle yapabilirsiniz.

JMSTimestamp- MQRFH2

Gönderme sırasında, JMSTimestamp alanı JVM ' nin saatine göre ayarlanır. Bu değer MQRFH2olarak ayarlanır. Bir send () üzerine yazılmadan önce JMSTimestamp alanında ayarlanan herhangi bir değer. Ayrıca bkz. JMS_IBM_PutDate ve JMS_IBM_PutTime özellikleri.

JMSType- MQRFH2

Bu dizgi MQRFH2 mcd.Type alanında ayarlanır. URI biçimindeyse, mcd.Set ve mcd.Fmt alanlarını da etkileyebilir.

JMSCorrelationID - MQMD.CorrelId, MQRFH2

JMSCorrelationID aşağıdakilerden birini tutabilir:

Sağlayıcıya özgü bir ileti tanıtıcısı

Bu, daha önce gönderilen ya da alınan bir iletiden gelen bir ileti tanıtıcısıdır ve `ID:` öneki kaldırılmış, geri kalan karakterler ikili değere dönüştürülmüş ve daha sonra MQMD.CorrelId alanı.

Sağlayıcı-yerel byte [] değeri

Değer MQMD.CorrelId alanı boş değer ile dolduruldu ya da gerekirse 24 byte 'a kesildi. MQRFH2içinde kodlanmış CorrelId değeri yok.

Uygulamaya özgü bir dizgi

Değer MQRFH2' ye kopyalanır. UTF8 biçiminde dizginin ilk 24 baytı MQMD.CorrelID.

JMS özellik alanlarını eşleme

Bu notlar, IBM MQ iletilerinde JMS özellik alanlarının eşlenmesine atıfta bulunur.

MQMD UserIdentifier ' dan JMSXUserID
JMSXUserID , gönderme çağrısından dönüşte ayarlanır.

MQMD PutApplAdından JMSXAppID
JSMXAppID , gönderme çağrısından dönüşte ayarlanır.

JMSXGroupID - MQRFH2 (noktadan noktaya)
Noktadan noktaya iletilerde, JMSXGroupID MQMD GroupID alanına kopyalanır. JMSXGroupID önek tanıtıcısı ile başlıyorsa, ikili değere dönüştürülür. Tersi durumda, UTF8 dizgisi olarak kodlanır. Değer 24 bayt uzunluğuna kadar doldurulur ya da gerekirse kesilir. MQMF_MSG_IN_GROUP işareti ayarlandı.

JMSXGroupID - MQRFH2 (yayınlama/abone olma)
Yayınlama/abone olma iletileri için, JMSXGroupID dizgi olarak MQRFH2 ' ye kopyalanır.

JMSXGroupSeq MQMD MsgSeqNumarası (noktadan noktaya)
Noktadan noktaya iletilerde, JMSXGroupSeq MQMD MsgSeqSayı alanına kopyalanır. MQMF_MSG_IN_GROUP işareti ayarlandı.

JMSXGroupSeq MQMD MsgSeqNumarası (yayınlama/abone olma)
Yayınlama/abone olma iletileri için JMSXGroupSeq , MQRFH2 ' ye i4olarak kopyalanır.

Sağlayıcıya özgü alanların JMS eşlenmesi
Aşağıdaki notlar, JMS sağlayıcısına özgü alanların IBM MQ iletileriyle eşlenmesine atıfta bulunur.

JMS_IBM_Report_*XXX* -MQMD Raporu
JMS uygulaması, aşağıdaki JMS_IBM_Report_*XXX* özelliklerini kullanarak MQMD Report seçeneklerini ayarlayabilir. Tek MQMD, birkaç JMS_IBM_Report_*XXX* özelliğiyle eşlendi.

JMS_IBM_Report_*XXX* sabitleri `com.ibm.msg.client.jakarta.wmq.WMQConstants` ya da `com.ibm.msg.client.wmq.WMQConstants`içinde bulunur.

JMS_IBM_Report_Exception

MQRO_EXCEPTION ya da
MQRO_EXCEPTION_WITH_DATA ya da
MQRO_EXCEPTION_WITH_FULL_DATA

JMS_IBM_Report_Expiration

MQRO_SÜRE bitimi ya da
MQRO_EXPIRATION_WITH_DATA ya da
MQRO_EXPIRATION_WITH_FULL_DATA

JMS_IBM_Report_COA

MQRO_COA ya da
MQRO_COA_WITH_DATA ya da
MQRO_COA_WITH_FULL_DATA

JMS_IBM_Report_COD

MQRO_COD ya da
MQRO_COD_WITH_DATA ya da
MQRO_COD_WITH_FULL_DATA

JMS_IBM_Report_PAN
MQRO_PAN

JMS_IBM_Report_NAN
MQRO_NAN

JMS_IBM_Report_Pass_Msg_ID
MQRO_PASS_MSG_ID

JMS_IBM_Report_Pass_Correl_ID
MQRO_PASS_CORREL_ID

JMS_IBM_Report_Discard_Msg
MQRO_DISCARD_MSG

MQRO değerleri `com.ibm.mq.constants.CMQC`.

JMS_IBM_MsgType -MQMD MsgType
Değer doğrudan MQMD MsgTypeile eşlenir. Uygulama, JMS_IBM_MsgTypeiçin belirtik bir değer belirlemediyse, varsayılan bir değer kullanılır. Bu varsayılan değer aşağıdaki gibi belirlenir:

- JMSReplyTo bir IBM MQ kuyruk hedefine ayarlanırsa, MSGType MQMT_REQUEST değerine ayarlanır
- JMSReplyTo ayarlanmazsa ya da IBM MQ kuyruk hedefi dışında bir değere ayarlanırsa, MsgType , MQMT_DATAGRAM değerine ayarlanır

JMS_IBM_Feedback to MQMD Feedback
Değer, doğrudan MQMD Geribildirimi ile eşlenir.

JMS_IBM_Format-MQMD Biçimi
Değer, doğrudan MQMD Biçimine eşlenir.

JMS_IBM_Encoding to MQMD Encoding (MQMD Kodlaması)
Ayarlanırsa, bu özellik Hedef Kuyruk ya da Konunun sayısal kodlamasını geçersiz kılar.

JMS_IBM_Character_Set to MQMD CodedCharacterSetId
Ayarlanırsa, bu özellik Hedef Kuyruk ya da Konunun kodlanmış karakter kümesi özelliğini geçersiz kılar.

MQMD PutDate ' ten JMS_IBM_PutDate
Bu özelliğin değeri, gönderme sırasında doğrudan MQMD ' deki PutDate alanından ayarlanır. Bir gönderme üzerine yazılmadan önce JMS_IBM_PutDate özelliğinde ayarlanan herhangi bir değer. Bu alan, YYYYAAGG ' nin IBM MQ Tarih biçimindeki sekiz karakterden oluşan bir dizgidir. Bu özellik, iletinin kuyruk yöneticisine göre konma zamanını saptamak için JMS_IBM_PutTime özelliğiyle birlikte kullanılabilir.

MQMD PutTime ' dan JMS_IBM_PutTime
Bu özelliğin değeri, gönderme sırasında doğrudan MQMD ' deki PutTime alanından ayarlanır. Bir gönderme üzerine yazılmadan önce JMS_IBM_PutTime özelliğine ayarlanan herhangi bir değer. Bu alan, HHMMSSTH ' nin IBM MQ Saat biçiminde sekiz karakterlik bir Dizilim 'dir. Bu özellik, iletinin kuyruk yöneticisine göre yerleştirildiği saati belirlemek için JMS_IBM_PutDate özelliğiyle birlikte kullanılabilir.

JMS_IBM_Last_Msg_In_Group-MQMD MsgFlags
Noktadan noktaya ileti alışverişi için bu Boole değeri, MQMD MsgFlags alanındaki MQMF_LAST_MSG_IN_GROUP işaretiyle eşlenir. Bu ileti, genellikle JMSXGroupID ve JMSXGroupSeq özellikleriyle, bir kalıt IBM MQ uygulamasına bu iletinin bir gruptaki son ileti olduğunu göstermek için kullanılır. Bu özellik, yayınlama/abone olma ileti sistemi için yoksayılır.

IBM MQ alanlarının JMS alanlarıyla (gelen iletiler) eşlenmesi
Bu çizelgeler, get () ya da receive () zamanındaki JMS üstbilgi ve özellik alanlarının MQMD ve MQRFH2 alanlarıyla nasıl eşlendiğini gösterir.

Çizelge 28 sayfa 154 , get () ya da receive () zamanında JMS üstbilgi alanlarının MQMD/MQRFH2 alanlarıyla nasıl eşlendiğini gösterir. Çizelge 29 sayfa 155 , JMS özellik alanlarının get () ya da receive () zamanındaki MQMD/MQRFH2 alanlarıyla nasıl eşlendiğini gösterir. Çizelge 30 sayfa 155 , JMS sağlayıcısına özgü özelliklerin nasıl eşlendiğini gösterir.

Çizelge 28. Gelen ileti JMS üstbilgi alanı eşlemesi

JMS üstbilgi alanı adı	**MQMD alanı alındı**	**MQRFH2 alanı şu adresten alındı:**
JMSDestination (JMS Hedefi)		jms.Dst ya da mqps.Top “1” sayfa 155
JMSDeliveryMode	Kalıcılık“2” sayfa 155	jms.Dlv “2” sayfa 155

Çizelge 28. Gelen ileti JMS üstbilgi alanı eşlemesi (devamı var)		
JMS üstbilgi alanı adı	**MQMD alanı alındı**	**MQRFH2 alanı şu adresten alındı:**
JMSExpiration		jms.Exp
JMS Önceliği	Öncelik	
JMSMessageID	MsgID	
JMSTimestamp	PutDate “2” sayfa 155 PutTime “2” sayfa 155	jms.Tms “2” sayfa 155
JMSCorrelationID	CorrelId “2” sayfa 155	jms.Cid “2” sayfa 155
JMSReplyTo	ReplyToS “2” sayfa 155 ReplyToQMgr “2” sayfa 155	jms.Rto “2” sayfa 155
JMSType		mcd.Type, mcd.Set, mcd.Fmt
Yeniden Teslim edilen JMS	BackoutCount	

Not:

1. Hem jms.Dst hem de mqps.Top ayarlanırsa, jms.Dst içindeki değer kullanılır.
2. MQRFH2 ya da MQMD 'den değer alabilen özellikler için, her ikisi de kullanılabiliyorsa, MQRFH2 ' deki ayar kullanılır.
3. JMS_IBM_Character_Set özellik değeri, sayısal CodedCharacterSetId değerinin Java karakter kümesi eşdeğerini içeren bir Dizgi değeridir.

Çizelge 29. Gelen ileti özelliği eşlemesi		
JMS özellik adı	**MQMD alanı alındı**	**MQRFH2 alanı şu adresten alındı:**
JMSXUserID	UserIdentifier	
JMSXAppID	PutApplAdı	
JMSXDeliveryCount	BackoutCount	
JMSXGroupID	GroupId “1” sayfa 155	jms.Gid “1” sayfa 155
JMSXGroupSeq	MsgSeqSayı “1” sayfa 155	jms.Seq “1” sayfa 155

Not:

1. MQRFH2 ya da MQMD 'den değer alabilen özellikler için, her ikisi de kullanılabiliyorsa, MQRFH2 ' deki ayar kullanılır. MQMF_MSG_IN_GROUP ya da MQMF_LAST_MSG_IN_GROUP ileti işaretleri ayarlandıysa, özellikler MQMD değerlerinden ayarlanır.

Çizelge 30. Gelen ileti sağlayıcıya özgü JMS özellik eşlemesi		
JMS özellik adı	**MQMD alanı alındı**	**MQRFH2 alanı şu adresten alındı:**
JMS_IBM_Report_Exception	Rapor	
JMS_IBM_Report_Expiration	Rapor	
JMS_IBM_Report_COA	Rapor	

Çizelge 30. Gelen ileti sağlayıcıya özgü JMS özellik eşlemesi (devamı var)		
JMS özellik adı	**MQMD alanı alındı**	**MQRFH2 alanı şu adresten alındı:**
JMS_IBM_Report_COD	Rapor	
JMS_IBM_Report_PAN	Rapor	
JMS_IBM_Report_NAN	Rapor	
JMS_IBM_Report_ Pass_Msg_ID	Rapor	
JMS_IBM_Report_Pass_Correl_ID	Rapor	
JMS_IBM_Report_Discard_Msg	Rapor	
JMS_IBM_MsgType	MsgType	
JMS_IBM_Feedback	Geri bildirim	
JMS_IBM_Biçimi	Biçim	
JMS_IBM_PutApplTipi	PutApplTipi	
JMS_IBM_Kodlaması [“1” sayfa 156]	Kodlama	
JMS_IBM_Character_Set [“1” sayfa 156]	CodedCharacterSetId	
JMS_IBM_PutDate	PutDate	
JMS_IBM_PutTime	PutTime	
JMS_IBM_Last_Msg_In_Group	MsgFlags	

1. Yalnızca gelen ileti Byte İletisi ise ayarlanır.

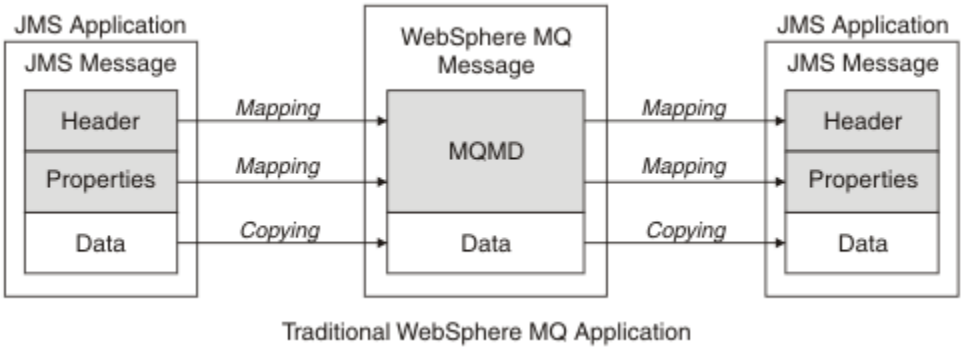
JMS uygulaması ile geleneksel IBM MQ uygulaması arasında ileti alışverişi
Bu konuda, bir JMS uygulaması MQRFH2 üstbilgisini işleyemeyen geleneksel bir IBM MQ uygulamasıyla ileti alışverişinde bulunduğunda ne olacağı açıklanmaktadır.

Şekil 11 sayfa 157 , eşlemeyi gösterir.

Yönetici, hedefin TARGCLIENT özelliğini *MQ*olarak ayarlayarak JMS uygulamasının geleneksel bir IBM MQ uygulamasıyla iletişim kurduğunu belirtir. Bu, MQRFH2 üstbilgisinin üretilmeyeceğini gösterir. Bu yapılmazsa, alan uygulamanın MQRFH2 üstbilgisini işleyebilmesi gerekir.

Geleneksel bir IBM MQ uygulamasını hedefleyen JMS -MQMD eşlemesi, bir JMS uygulamasını hedefleyen JMS ile MQMD eşlemesi ile aynıdır. IBM MQ classes for JMS , MQMD *Format* alanı MQFMT_RFH2dışında bir değere ayarlanmış bir IBM MQ iletisi alırsa, verilerJMS dışı bir uygulamadan alınır. Biçim MQFMT_STRING ise, ileti JMS metin iletisi olarak alınır. Tersi durumda, bir JMS byte iletisi olarak alınır. MQRFH2olmadığı için yalnızca MQMD ' de iletilen JMS özellikleri geri yüklenebilir.

IBM MQ classes for JMS , MQRFH2 üstbilgisi olmayan bir ileti alırsa, iletinin JMSReplyTo üstbilgi alanından türetilen Kuyruk ya da Konu nesnesinin TARGCLIENT özelliği varsayılan olarak MQ olarak ayarlanır. Bu, kuyruğa ya da konuya gönderilen bir yanıt iletisinin MQRFH2 üstbilgisine sahip olmadığı anlamına gelir. Bağlantı üreticisinin TARGCLIENTMATCHING özelliğini NO olarak ayarlayarak, özgün iletinin MQRFH2 üstbilgisi varsa, yanıt iletisine MQRFH2 üstbilgisi ekleme davranışını kapatabilirsiniz.



Şekil 11. JMS iletilerinin MQRFH2 üstbilgisi olmadan IBM MQ iletilerine dönüştürülmesi

JMS ileti gövdesi
Bu konu, ileti gövdesinin kodlamasına ilişkin bilgileri içerir. Kodlama, JMS iletisinin tipine bağlıdır.

ObjectMessage
ObjectMessage , Java Runtime tarafından normal şekilde diziselleştirilmiş bir nesnedir.

TextMessage
TextMessage , kodlanmış bir dizedir. Giden bir ileti için, dizgi hedef nesne tarafından belirtilen karakter kümesinde kodlanır. Bu, varsayılan olarak UTF8 kodlamasına ayarlanır (UTF8 kodlaması, iletinin ilk karakteriyle başlar; başlangıçta uzunluk alanı yoktur). Ancak, IBM MQ classes for JMStarafından desteklenen başka bir karakter kümesi belirtilabilir. Bu tür karakter kümeleri,JMS dışı bir uygulamaya ileti gönderdiğinizde kullanılır.

Karakter takımı çift baytlık bir kümese (UTF16dahil), hedef nesnenin tamsayı kodlama belirtimi baytların sırasını belirler.

Gelen bir ileti, iletinin kendisinde belirtilen karakter kümesi ve kodlama kullanılarak yorumlanır. Bu belirtimler son IBM MQ üstbilgisinde (ya da üstbilgi yoksa MQMD) bulunur. JMS iletileri için son üstbilgi genellikle MQRFH2' dir.

BytesMessage
BytesMessage varsayılan olarak, JMS 1.0.2 belirtimi ve ilişkili Java belgelerinde tanımlandığı şekilde bir bayt dizisidir.

Uygulamanın kendisi tarafından derlenen bir giden ileti için, hedef nesnenin kodlama özelliği, iletide bulunan tamsayı ve kayan noktalı alanların kodlamasını geçersiz kılmak için kullanılabilir. Örneğin, kayan nokta değerlerinin IEEE biçimi yerine S/390 biçiminde saklanmasını isteyebilirsiniz.

Gelen bir ileti, iletinin kendisinde belirtilen sayısal kodlama kullanılarak yorumlanır. Bu belirtim son IBM MQ üstbilgisinde (ya da üstbilgi yoksa MQMD) bulunur. JMS iletileri için son üstbilgi genellikle MQRFH2' dir.

Bir BytesMessage alınırsa ve değiştirilmeden yeniden gönderilirse, gövdesi alındığı gibi bayt için iletilir. Hedef nesnenin kodlama özelliğinin gövde üzerinde bir etkisi yoktur. BytesMessage içinde belirtik olarak gönderilebilen tek dizgi benzeri varlık UTF8 dizgisidir. Bu, Java UTF8 biçiminde kodlanır ve 2 baytlık bir uzunluk alanıyla başlar. Hedef nesnenin karakter kümesi özelliği, giden bir BytesMessagekodlamasını etkilemez. Gelen IBM MQ iletisindeki karakter kümesi değeri, o iletinin JMS BytesMessageolarak yorumlanmasında etkili olmaz.

Java dışı uygulamaların Java UTF8 kodlamasını tanıması olası değildir. Bu nedenle, bir JMS uygulamasının metin verileri içeren bir BytesMessage göndermesi için uygulamanın dizgilerini bayt dizilerine dönüştürmesi ve bu bayt dizilerini BytesMessageiçine yazması gerekir.

MapMessage
MapMessage , şu şekilde kodlanmış XML ad/tip/değer üçlülerini içeren bir dizedir:

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
```

```
  ...
</map>
```

Burada `datatype` , Çizelge 21 sayfa 147içinde listelenen veri tiplerinden biridir. Varsayılan veri tipi `string`olduğundan dizgi öğeleri için `dt="string"` özniteliği atlandı.

Bir eşlem iletisinin gövdesini oluşturan XML dizgisini kodlamak ya da yorumlamak için kullanılan karakter kümesi, metin iletisine uygulanan kurallara göre belirlenir.

5.3 öncesindeki IBM MQ classes for JMS sürümleri, bir eşlem iletisinin gövdesini aşağıdaki biçimde kodladı:

```
<map>
  <elementname1 dt="datatype1">value1</elementname1>
  <elementname2 dt="datatype2">value2</elementname2>
  ...
</map>
```

IBM MQ classes for JMS 5.3 ve sonraki sürümleri her iki biçimi de yorumlayabilir, ancak 5.3 sürümünden önceki IBM MQ classes for JMS sürümleri geçerli biçimi yorumlayamaz.

Bir uygulamanın IBM MQ classes for JMS 5.3sürümünden önceki bir sürümü kullanan başka bir uygulamaya eşlem iletileri göndermesi gerekiyorsa, gönderen uygulamanın, eşlem iletilerinin önceki biçimde gönderileceğini belirtmek için bağlantı üreticisi yöntemini `setMapNameStyle(WMQConstants.WMQ_MAP_NAME_STYLE_COMPATIBLE)` çağırması gerekir. Varsayılan olarak, tüm harita iletileri geçerli biçimde gönderilir.

StreamMessage
StreamMessage , bir eşlem iletisine benzer, ancak öğe adları yok:

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
  ...
</stream>
```

Burada `datatype` , Çizelge 21 sayfa 147içinde listelenen veri tiplerinden biridir. Varsayılan veri tipi `string`olduğundan dizgi öğeleri için `dt="string"` özniteliği atlandı.

StreamMessage gövdesini oluşturacak XML dizgisini kodlamak ya da yorumlamak için kullanılan karakter kümesi, bir TextMessageiçin geçerli olan kurallara göre belirlenir.

MQRFH2.format alanı aşağıdaki gibi ayarlanır:

MQFMT_NONE
ObjectMessage, BytesMessageya da gövdesi olmayan iletiler için.

MQFMT_STRING
TextMessage, StreamMessageya da MapMessageiçin.

JMS ileti dönüştürme
JMS içinde ileti verileri dönüşümü, ileti gönderilirken ve alınırken gerçekleştirilir. IBM MQ çoğu veri dönüştürmeyi otomatik olarak gerçekleştirir. JMS uygulamaları arasında bir ileti aktarırken metin ve sayısal verileri dönüştürür. Bir JMS uygulaması ile IBM MQ uygulaması arasında `JMSTextMessage` değiş tokuş edilirken metin dönüştürülür.

Daha karmaşık ileti alışverişi yapmayı planlıyorsanız, aşağıdaki konular ilginizi çeker. Karmaşık ileti değiş tokuşları şunlardır:

- Metin dışı iletilerin IBM MQ uygulaması ile JMS uygulaması arasında aktarılması.
- Metin verilerinin bayt biçiminde değiş tokuş edilmesiyle ilgili.
- Uygulamanızdaki metin dönüştürülüyor.

JMS İleti Verileri

İki JMS uygulaması arasında bile uygulamalar arasında metin ve sayısal veri alışverişi için veri dönüştürme gerekir. Bir iletide aktarılabilmeleri için metin ve sayıların iç gösterimi kodlanmalıdır. Kodlama, sayıların ve metnin nasıl gösterildiğine ilişkin bir kararı zorlar. IBM MQ , `JMSObjectMessage`dışında JMS iletilerinde metin ve sayı kodlamasını yönetir, bkz. "JMSObjectMessage" sayfa 165. Üç ileti özniteliği kullanır. Üç öznitelik: `CodedCharacterSetId`, `Encoding`ve `Format`.

Bu üç ileti özniteliği genellikle JMS iletisinin JMS üstbilgisinde (`MQRFH2`) saklanır. İleti tipi, `JMS` ileti tipi yerine `MQ`ise, öznitelikler ileti tanımlayıcısında (`MQMD`) saklanır. Öznitelikler, JMS ileti verilerini dönüştürmek için kullanılır. JMS ileti verileri, bir IBM MQ iletisinin ileti verileri bölümüne aktarılır.

JMS İleti Özellikleri

JMS ileti özellikleri (`JMS_IBM_CHARACTER_SET`gibi), ileti bir `MQRFH2`olmadan gönderilmedikçe, JMS iletisinin `MQRFH2` üstbilgi kısmında değiş tokuş edilir. `MQRFH2`olmadan yalnızca `JMSTextMessage` ve `JMSBytesMessage` gönderilebilir. Bir JMS özelliği, `MQMD`ileti tanımlayıcısında bir IBM MQ ileti özelliği olarak saklandıysa, `MQMD` dönüşümünün bir parçası olarak dönüştürülür. Bir JMS özelliği `MQRFH2`içinde saklanıyorsa, bu özellik `MQRFH2.NameValueCCSID`ile belirtilen karakter kümesinde saklanır. Bir ileti gönderildiğinde ya da alındığında, ileti özellikleri JVM ' deki iç gösterimlerine ve iç gösterimlerinden dönüştürülür. Dönüştürme, ileti tanımlayıcısının ya da `MQRFH2.NameValueCCSID`' in karakter takımındaki ya da karakter takımındaki dönüşümdür. Sayısal veriler metne dönüştürülür.

JMS ileti dönüştürme

Aşağıdaki konular, dönüştürme gerektiren daha karmaşık iletileri değiş tokuş etmeyi planlıyorsanız yararlı olan örnekleri ve görevleri içerir.

JMS ileti dönüştürme yaklaşımları
JMS uygulama tasarımcılarına bir dizi veri dönüştürme yaklaşımı açıktır. Bu yaklaşımlar münhasır değildir; bazı uygulamaların bu yaklaşımların bir kombinasyonunu kullanması muhtemeldir. Uygulamanız yalnızca metin alışverişi yapıyorsa ya da yalnızca diğer JMS uygulamalarıyla ileti alışverişi yapıyorsa, normalde veri dönüştürmeyi düşünmezsiniz. Veri dönüştürme sizin için IBM MQtarafından otomatik olarak gerçekleştirilir.

İleti dönüştürmeye nasıl yaklaşılacağına ilişkin birkaç soru sorabilirsiniz:

Mesajın dönüştürülmesi hakkında düşünmek gerekli mi?
JMS to JMS ileti aktarımları ve metin iletilerinin IBM MQ programlarıyla değiş tokuş edilmesi gibi bazı durumlarda, IBM MQ sizin için gerekli dönüştürmeleri otomatik olarak gerçekleştirir. Performans nedenleriyle veri dönüştürmeyi denetlemek isteyebilir ya da önceden tanımlanmış biçimi olan karmaşık iletileri değiş tokuş ediyor olabilirsiniz. Bu tür durumlarda, ileti dönüştürmeyi anlamanız ve aşağıdaki konuları okumanız gerekir.

Ne tür bir dönüşüm var?
Aşağıdaki bölümlerde açıklanan dört ana dönüştürme türü vardır:

1. "JMS istemcisi veri dönüştürme" sayfa 160
2. "Uygulama verilerini dönüştürme" sayfa 160
3. "Kuyruk yöneticisi veri dönüştürme" sayfa 161
4. "İleti kanalı veri dönüştürme" sayfa 161

Dönüştürme nerede gerçekleştirilmelidir?
"İleti dönüştürmeye ilişkin bir yaklaşım seçilmesi: alıcı iyi" sayfa 162bölümünde, "alıcının iyi yaptığı"olağan yaklaşımı açıklanmaktadır. "Günlük nesnesi" ' yi iyi yapar, JMS veri dönüştürmesi için de geçerlidir.

JMS istemcisi veri dönüştürme

JMS istemci[1]Veri dönüştürme, bir hedefe gönderilirken JMS iletisinde Java temel öğelerinin ve nesnelerinin bayta dönüştürülmesi ve alındığında yeniden dönüştürülmesi işlemidir. JMS istemcisi veri dönüştürme işlemi `JMSMessage` sınıflarının yöntemlerini kullanır. Yöntemler, Çizelge 31 sayfa 163içindeki `JMSMessage` sınıf tipine göre listelenir.

Okuma, alma, ayarlama ve yazma yöntemleri için sayıların ve metnin iç JVM gösterimine/gösteriminden dönüştürme gerçekleştirilir. Dönüştürme, ileti gönderildiğinde ve alınan bir iletide okuma ya da alma yöntemlerinden herhangi biri çağrıldığında gerçekleştirilir.

Bir iletinin içeriğini yazmak ya da ayarlamak için kullanılan kod sayfası ve sayısal kodlama, hedefin öznitelikleri olarak tanımlanır. Hedef kod sayfası ve sayısal kodlama yönetimsel olarak değiştirilebilir. Bir uygulama, ileti içeriğinin yazılmasını ya da ayarlanmasını denetleyen ileti özelliklerini ayarlayarak hedef kod sayfasını ve kodlamayı da geçersiz kılabilir.

Bir `JMSBytesMessage` iletisi `Native` kodlaması olarak tanımlanmamış bir hedefe gönderildiğinde sayı kodlamasını dönüştürmek istiyorsanız, iletiyi göndermeden önce `JMS_IBM_ENCODING` ileti özelliğini ayarlamanız gerekir. "Alıcı iyi" kalıbını izliyorsanız ya da JMS uygulamaları arasında ileti alışverişi yapıyorsanız, uygulamanın `JMS_IBM_ENCODING`ayarını tanımlamasına gerek yoktur. Çoğu durumda `Encoding` özelliğini `Native`olarak bırakabilirsiniz.

`JMSStreamMessage`, `JMSMapMessage`ve `JMSTextMessage` iletileri için, hedefin karakter kümesi tanıtıcısı özellikleri kullanılır. Sayılar metin biçiminde yazıldığı için, gönderirken kodlama yoksayılır. Uygulanacak hedef karakter kümesi özelliği varsa, iletiyi göndermeden önce JMS istemci uygulama programının `JMS_IBM_CHARACTER_SET` ayarını belirmesi gerekmez.

Bir iletideki verileri almak için bir uygulama JMS ileti okuma ya da alma yöntemlerini çağırır. Yöntemler, Java temel öğelerinin ve nesnelerinin doğru bir şekilde yaratılması için önceki ileti üstbilgisinde tanımlanan kod sayfasına ve kodlamaya başvuruda bulunur.

JMS istemci verileri dönüşümü, bir JMS istemcisi ile başka bir istemci arasında ileti alışverişi yapan JMS uygulamalarının çoğunun gereksinimlerini karşılar. Herhangi bir belirtik veri dönüştürmesini kodlayın. Genellikle bir dosyaya metin yazarken kullanılan `java.nio.charset.Charset` sınıfını kullanamazsınız. `writeString` ve `setString` yöntemleri sizin için dönüştürmeyi yapar.

JMS istemcisi veri dönüştürmesine ilişkin daha fazla ayrıntı için bkz. “JMS istemcisi ileti dönüştürme ve kodlama” sayfa 171.

Uygulama verilerini dönüştürme

Bir JMS istemci uygulaması, `java.nio.charset.Charset` sınıfını kullanarak belirtik karakter verileri dönüştürmesi gerçekleştirebilir; Şekil 14 sayfa 164 ve Şekil 15 sayfa 164içindeki örneklere bakın. Dizgi verileri, `getBytes` yöntemi kullanılarak bayta dönüştürülür ve bayt olarak gönderilir. Bayt dizisi ve `Charset`alan bir `String` oluşturucusu kullanılarak bayt yeniden metne dönüştürülür. Karakter verileri, `encode` ve `decode` `Charset` yöntemleri kullanılarak dönüştürülür. Genellikle ileti `JMSBytesMessage`olarak gönderilir ya da alınır; çünkü `JMSBytesMessage` ileti kısmı, uygulama tarafından yazılan verilerden başka bir şey içermez[2]. Baytları `JMSStreamMessage`, `JMSMapMessage`ya da `JMSObjectMessage`kullanarak da gönderebilir ve alabilirsiniz.

Farklı kodlama biçimlerinde gösterilen sayısal verileri içeren byte 'ları kodlamak ve kodlarını çözmek için Java yöntemi yoktur. Sayısal veriler, sayısal `JMSMessage` okuma ve yazma yöntemleri kullanılarak otomatik olarak kodlanır ve kodu çözülür. Okuma ve yazma yöntemleri, ileti verilerinin `JMS_IBM_ENCODING` özniteliğinin değerini kullanır.

Uygulama verileri dönüşümü için tipik bir kullanım, JMS istemcisininJMS dışı bir uygulamadan biçimlendirilmiş bir ileti göndermesi ya da alması durumunda olur. Biçimlenmiş bir ileti, veri alanlarının uzunluğuna göre düzenlenmiş metin, sayısal ve bayt verilerini içerir. JMS dışı uygulama ileti biçimini

[1] "JMS İstemci" , istemcide ya da bağ tanımlama kipinde çalışan JMS arabirimini gerçekleştiren IBM MQ classes for JMS ' yi ifade eder.

[2] Bir özel durum: `writeUTF` kullanılarak yazılan veriler 2 baytlık bir alanla başlar

"MQSTR "olarak belirtmediyse, ileti JMSBytesMessageolarak oluşturulur. JMSBytesMessage içinde biçimlendirilmiş ileti verilerini almak için bir yöntem dizisi çağırmanız gerekir. Yöntemlerin, alanların iletiye yazıldığı sırayla çağrılması gerekir. Alanlar sayısalsa, sayısal verilerin kodlamasını ve uzunluğunu bilmeniz gerekir. Alanlardan herhangi biri bayt ya da metin verisi içeriyorsa, iletideki bayt verilerinin uzunluğunu bilmeniz gerekir. Biçimlendirilmiş bir iletiyi kullanımı kolay bir Java nesnesine dönüştürmenin iki yolu vardır.

1. İletiyi okumak ve yazmak için kayıtla ilgili bir Java sınıfı oluşturun. Kayıttaki verilere erişim, sınıfın get (alma) ve set (ayarlama) yöntemleriyle olur.
2. `com.ibm.mq.headers` sınıfını genişleterek kayda karşılık gelen bir Java sınıfı oluşturun. Sınıftaki verilere erişim, formun tipe özgü erişimcileriyle (`getStringValue(`*`fieldname`*`);`) olur.

Bkz. "JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi" sayfa 179.

Kuyruk yöneticisi veri dönüştürme

Bir JMS istemci programı bir ileti aldığında, kuyruk yöneticisi tarafından kod sayfası dönüştürme işlemi gerçekleştirilebilir. Dönüştürme, bir C programı için gerçekleştirilen dönüştürmeyle aynıdır. C programı MQGMO_CONVERT öğesini MQGET `GetMsgOpts` değiştirge seçeneği olarak ayarlar; bkz. Şekil 13 sayfa 164. Bir kuyruk yöneticisi, ileti alan bir JMS istemci programı için dönüştürme gerçekleştirir; WMQ_RECEIVE_CONVERSION hedef özelliği WMQ_RECEIVE_CONVERSION_QMGRolarak ayarlanırsa, JMS istemci programı hedef özelliği de ayarlayabilir; bkz. Şekil 12 sayfa 161.

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Ya da ...

```
((MQDestination)destination).setReceiveConversion
              (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Şekil 12. Kuyruk yöneticisi veri dönüştürmesini etkinleştir

Kuyruk yöneticisi dönüşümünün ana yararı,JMS dışı uygulamalarla ileti alışverişi sırasında gelir. İletideki `Format` alanı tanımlıysa ve hedef karakter kümesi ya da kodlama iletiden farklıysa, kuyruk yöneticisi hedef uygulama isterse, hedef uygulama için veri dönüştürme gerçekleştirir. Kuyruk yöneticisi, CICS bridge üstbilgisi (MQCIH) gibi önceden tanımlanmış IBM MQ ileti tiplerinden birine göre biçimlendirilen ileti verilerini dönüştürür. `Format` alanı kullanıcı tanımlı ise, kuyruk yöneticisi `Format` alanında sağlanan adla bir veri dönüştürme çıkışı arar.

Kuyruk yöneticisi veri dönüştürmesi, "alıcısı" tasarım örüntüsünü en iyi şekilde etkilemek için kullanılır. Gönderen JMS istemcisinin dönüştürme gerçekleştirmesi gerekmez. JMS dışı bir alan program, iletinin gerekli kod sayfasında ve kodlamada teslim edildiğinden emin olmak için dönüştürme çıkışına dayanır. Gönderen bir JMS istemcisi veJMS olmayan bir alıcıyla, bu örnek IBM MQiçin geçerlidir.

Kuyruk yöneticisinin kendi kayıt biçimlenmiş verilerinizi dönüştürmesini sağlamak için, veri dönüştürme çıkış yardımcı programını (**`crtmqcvx`**) kullanarak bir veri dönüştürme çıkışı yaratabilirsiniz. Kendi kayıt biçiminizi oluşturabilir, bir Java sınıfı olarak erişmek için `com.ibm.mq.headers` ' i kullanabilir ve bunu dönüştürmek için kendi dönüştürme çıkışınızı kullanabilirsiniz. z/OS üzerinde yardımcı program **CSQUCVX**ve IBM iüzerinde **CVTMQMDTA**olarak adlandırılır. Bkz. "JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi" sayfa 179.

İleti kanalı veri dönüştürme

IBM MQ Gönderen, Sunucu, Küme Alıcısı ve Küme Gönderen kanallarının bir ileti dönüştürme seçeneği vardır, CONVERT. Bir iletinin içeriği, isteğe bağlı olarak bir ileti gönderildiğinde dönüştürülebilir.

Dönüştürme, kanalın gönderme sonunda gerçekleşir. Küme alıcı tanımlaması, karşılık gelen küme gönderen kanalını otomatik olarak tanımlamak için kullanılır.

İleti kanalları tarafından veri dönüştürme, genellikle diğer dönüştürme biçimlerini kullanmak mümkün değilse kullanılır.

İleti dönüştürmeye ilişkin bir yaklaşım seçilmesi: "alıcı iyi"

IBM MQ uygulama tasarımında kod dönüştürmeye ilişkin olağan yaklaşım "alıcı iyi yapıyor". "Günlük nesnesi iyi" , ileti dönüştürme sayısını azaltır. İleti aktarımı sırasında bazı aracı kuyruk yöneticisinde ileti dönüştürme başarısız olursa, beklenmeyen kanal hataları oluşmasını da önler. "Günlük nesnesi iyi" kuralı yalnızca, alıcının iyi olamamasının bir nedeni varsa bozulur. Örneğin, alıcı altyapıda doğru karakter kümesi olmayabilir.

"Günlük nesnesi iyi yapar" , JMS istemci uygulamaları için de iyi bir genel kılavuzdur. Ancak belirli durumlarda, kaynakta doğru karakter kümesine dönüştürme daha verimli olabilir. Metin ya da sayısal tipler içeren bir ileti gönderildiğinde, JVM iç gösteriminden dönüştürme gerçekleştirilmelidir. Günlük nesnesi bir JMS istemcisi değilse, alıcının gerektirdiği karakter kümesine dönüştürme işlemi,JMS dışı alıcının dönüştürme gerçekleştirme gereksinimini ortadan kaldırabilir. Alıcı bir JMS istemcisiyse, ileti verilerinin kodunu çözmek ve Java temel öğeleri ve nesneleri oluşturmak için yeniden dönüştürülür.

JMS istemci uygulamaları ile C gibi bir dilde yazılan uygulamalar arasındaki fark, Java ' in veri dönüştürmesi gerçekleştirmesi gerekmesi gerekmesi. Java uygulaması, sayıları ve metni iç gösterimlerinden iletilerde kullanılan kodlanmış biçime dönüştürmelidir.

Hedef ya da ileti özelliklerini ayarlayarak, iletilerdeki sayıları ve metni kodlamak için IBM MQ tarafından kullanılan karakter kümesini ve kodlamayı ayarlayabilirsiniz. Normalde, karakter kümesini `1208` ve kodlamayı `Native`olarak bırakırsınız.

IBM MQ , bayt dizilerini dönüştürmez. Dizgileri ve karakter dizilerini bayt dizilerine kodlamak için `java.nio.charset` paketini kullanın. `Charset` , bir dizgiyi ya da karakter dizisini bayt dizisine dönüştürmek için kullanılan karakter takımını belirtir. Bir `Charset`kullanarak bir bayt dizisinin kodunu bir dizgiye ya da karakter dizisine çözebilirsiniz. Dizgiler ve karakter dizileri kodlanırken `java.nio.charset.Charset.defaultCodePage` ' e güvenmek iyi bir uygulama değildir. Varsayılan `Charset` , genellikle Windowsüzerinde `windows-1252` ve AIX and Linuxüzerinde `UTF-8` ' dir. `windows-1252` , tek baytlık bir karakter takıbudur ve `UTF-8` çok baytlık bir karakter takısıdır.

Genellikle, diğer JMS uygulamalarıyla ileti alışverişi yaparken hedef karakter kümesi ve kodlama özelliklerini varsayılan `UTF-8` ve `Native` değerlerinde bırakın. Bir JMS uygulamasıyla sayı ya da metin içeren iletileri değiştiriyorsanız, amacınıza uygun `JMSTextMessage`, `JMSStreamMessage`, `JMSMapMessage`eya da `JMSObjectMessage` ileti tiplerinden birini seçin. Yapılacak başka dönüştürme görevi yok.

Kayıt biçimi kullananJMS dışı uygulamalarla ileti alışverişi daha karmaşıktır. Tüm kayıt metin içermediği ve `JMSTextMessage`olarak aktarılamadığı sürece, uygulamadaki metni kodlamanız ve metnin kodunu çözmeniz gerekir. Hedef ileti tipini `MQ`olarak ayarlayın ve IBM MQ classes for JMS ' in ileti verilerine ek üstbilgi ve etiketleme bilgileri eklemesini önlemek için `JMSBytesMessage` komutunu kullanın. Sayı ve bayt yazmak için `JMSBytesMessage` yöntemlerini ve `Charset` sınıfı metni bayt dizilerine belirtik olarak dönüştürür. Bazı etkenler, karakter kümesi seçiminizi etkileyebilir:

- Performans: Metni, en çok sayıda sunucuda kullanılan bir karakter kümesine dönüştürerek dönüştürme sayısını azaltabilir misiniz?
- Tekdüzelik: Aynı karakter takımındaki tüm iletileri aktar.
- Zenginlik: Hangi karakter kümeleri, uygulamaların kullanması gereken tüm kod noktalarına sahip?
- Basitlik: Tek baytlık karakter kümelerinin kullanımı değişken uzunluklu ve çok baytlık karakter kümelerinden daha basittir.

Bkz. “JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi” sayfa 179. JMS dışı uygulamalarla değiş tokuş edilen iletileri dönüştürme örnekleri için.

Örnekler

İleti tipleri ve dönüştürme tipleri tablosu

Çizelge 31. İleti tipleri ve dönüştürme tipleri

	Dönüştürme tipi			
İleti tipi	**Metin**	**Sayısal**	**Diğer**	**Yok**
JMSObjectMessage				getObject setObject
JMSTextMessage	getText setText			
JMSBytesMessage	readUTF writeUTF	readDouble readFloat readInt readLong readShort readUnsignedShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readUnsignedByte readBytes readChar writeByte writeBytes writeChar
JMSStreamMessage	readString writeString	readDouble readFloat readInt readLong readShort writeDouble writeFloat writeInt writeLong writeShort	readBoolean readObject writeBoolean writeObject	readByte readBytes readChar writeByte writeBytes writeChar
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

```
gmo.Options = MQGMO_WAIT         /* wait for new messages           */
            | MQGMO_NO_SYNCPOINT /* no transaction                  */
            | MQGMO_CONVERT;     /* convert if necessary            */

     while (CompCode != MQCC_FAILED) {
     buflen = sizeof(buffer) - 1; /* buffer size available for GET   */
     memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
     memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
     md.Encoding       = MQENC_NATIVE;
     md.CodedCharSetId = MQCCSI_Q_MGR;

           MQGET(Hcon,          /* connection handle               */
           Hobj,                /* object handle                   */
           &md,                 /* message descriptor              */
           &gmo,                /* get message options             */
           buflen,              /* buffer length                   */
           buffer,              /* message buffer                  */
           &messlen,            /* message length                  */
           &CompCode,           /* completion code                 */
           &Reason);            /* reason code                     */
```

Şekil 13. amqsget0.c kod parçacığı

JMSBytesMessage içinde metin gönderme ve alma

Şekil 14 sayfa 164 içindeki kod, BytesMessageiçinde bir dizgi gönderir. Basitlik için örnek, JMSTextMessage ' in daha uygun olduğu tek bir dizgi gönderir. Bayt cinsinden bir metin dizesi almak için, Şekil 15 sayfa 164içinde *TEXT_LENGTH* olarak adlandırılan bayt cinsinden dizginin uzunluğunu bilmeniz gerekir. Sabit sayıda karakter içeren bir dizgi için bile, bayt gösteriminin uzunluğu daha uzun olabilir.

```
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);
```

Şekil 14. JMSBytesMessage içinde String gönderilmesi

```
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);
```

Şekil 15. JMSBytesMessage olanağından String alınması

İlgili kavramlar

JMS istemcisi ileti dönüştürme ve kodlama
JMS istemcisi ileti dönüştürme ve kodlamasını gerçekleştirmek için kullandığınız yöntemler, her dönüştürme tipine ilişkin kod örnekleriyle birlikte listelenir.

Kuyruk yöneticisi veri dönüştürme
Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alanJMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

İlgili görevler

JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi
Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları ve JMSBytesMessagekullanarakJMS dışı bir uygulamayla ileti değiş tokuşu yapabilen bir JMS istemci uygulamasını izleyin. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değiş tokuşu, veri dönüştürme çıkışı çağrılmasıyla ya da çağrılmadan gerçekleştirilebilir.

İlgili başvurular

JMS ileti tipleri ve dönüştürmesi
İleti tipi seçimi, ileti dönüştürme yaklaşımınızı etkiler. İleti dönüştürme ve ileti tipi etkileşimi, JMS ileti tipleri JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessageve JMSBytesMessageiçin açıklanır.

JMS ileti tipleri ve dönüştürmesi
İleti tipi seçimi, ileti dönüştürme yaklaşımınızı etkiler. İleti dönüştürme ve ileti tipi etkileşimi, JMS ileti tipleri JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessageve JMSBytesMessageiçin açıklanır.

JMSObjectMessage

JMSObjectMessage , JVM tarafından bayt akımına diziselleştirilmiş bir nesne ve başvuruda bulunduğu nesneleri içerir. Metin UTF-8olarak diziselleştirilir ve 65534 baytı geçmeyecek dizgilerle ya da karakter dizileriyle sınırlıdır. JMSObjectMessage ' in bir yararı, yalnızca nesnenin yöntemlerini ve özniteliklerini kullandıkları sürece uygulamaların herhangi bir veri dönüştürme sorununa dahil olmamasıdır. JMSObjectMessage , bir nesnenin iletide nasıl kodlanacağını göz önünde bulundurarak, uygulama programcısı olmadan karmaşık nesneler için veri dönüştürme sağlar. JMSObjectMessage ' in kullanılmasının dezavantajı, yalnızca diğer JMS uygulamalarıyla değiş tokuş edilebilmesidir. Diğer JMS ileti tiplerinden birini seçerek, JMS iletileriniJMS dışı uygulamalarla değiş tokuş edebilirsiniz.

“JMSObjectMessage gönderme ve alma” sayfa 167 , bir iletide değiş tokuş edilen String nesnesini gösterir.

JMS istemci uygulaması, JMSObjectMessage ' yi yalnızca JMSstili gövdesi olan bir iletide alabilir. Hedef bir JMS biçem gövdesi belirtmelidir.

JMSTextMessage

JMSTextMessage tek bir metin dizgisi içerir. Bir metin iletisi gönderildiğinde, Format metni "MQSTR ", WMQConstants.MQFMT_STRINGolarak ayarlanır. Metnin CodedCharacterSetId değeri, hedefi için tanımlanan kodlanmış karakter takımı tanıtıcısına ayarlanır. Metin, IBM MQtarafından CodedCharacterSetId içine kodlanır. CodedCharacterSetId ve Format alanları, ileti tanımlayıcısında (MQMD) ya da MQRFH2içindeki JMS alanlarında ayarlanır. İleti, WMQ_MESSAGE_BODY_MQ ileti gövdesi stiline sahip olarak tanımlandıysa ya da gövde stili belirtilmediyse, ancak hedef hedef WMQ_TARGET_DEST_MQise, ileti tanımlayıcı alanları ayarlanır. Aksi takdirde, iletide bir JMS RFH2 bulunur ve alanlar MQRFH2' ın sabit bölümünde ayarlanır.

Bir uygulama, hedef için tanımlanan kodlanmış karakter takımı tanıtıcısını geçersiz kılabilir. JMS_IBM_CHARACTER_SET ileti özelliğini bir kodlanmış karakter kümesi tanıtıcısı olarak ayarlamalıdır; örneğe bakın: “JMSTextmessage gönderme ve alma” sayfa 168.

JMS istemcisi consumer.receive yöntem kuyruk yöneticisi dönüştürmesini çağırdığında isteğe bağlıdır. WMQ_RECEIVE_CONVERSION hedef özelliği WMQ_RECEIVE_CONVERSION_QMGRolarak ayarlanarak kuyruk yöneticisi dönüşümü etkinleştirilir. Kuyruk yöneticisi, iletiyi JMS istemcisine aktarmadan önce, ileti için belirtilen JMS_IBM_CHARACTER_SET öğesinden metin iletisini dönüştürür. Hedefin farklı bir WMQ_RECEIVE_CCSIDkarakteri yoksa, dönüştürülen iletinin karakter kümesi 1208, UTF-8olur. JMSTextMessage öğesine başvuran iletideki CodedCharacterSetId , hedef karakter kümesi tanıtıcısına güncellenir. Metnin kodu, getText yöntemiyle hedef karakter kümesinden Unicode 'a çözülür; “JMSTextmessage gönderme ve alma” sayfa 168içindeki örneğe bakın.

JMSTextMessage , JMS MQRFH2 üstbilgisi olmadan MQstilinde bir ileti gövdesinde gönderilebilir. Hedef özniteliklerin değeri olan WMQ_MESSAGE_BODY ve

WMQ_TARGET_DEST , uygulama tarafından geçersiz kılınmadıkça, ileti gövdesi stilini belirler. Uygulama, destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) ya da destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ) yöntemini çağırarak hedefte ayarlanan değerleri geçersiz kılabilir.

Bir JMSTextMessage öğesini WMQ_MESSAGE_BODY ayarı WMQ_MESSAGE_BODY_MQolan bir hedefe göndererek MQ biçem gövdesiyle gönderirseniz, bunu aynı hedeften JMSTextMessage olarak alamazsınız. WMQ_MESSAGE_BODY ayarı WMQ_MESSAGE_BODY_MQ olan bir hedeften alınan tüm iletiler JMSBytesMessageolarak alınır. İletiyi JMSTextMessage olarak almayı denerseniz, bu bir kural dışı duruma neden olur ClassCastException: com.ibm.jms.JMSBytesMessage cannot be cast to jakarta (or javax).jms.TextMessage.

Not: JMSBytesMessage içindeki metin JMS istemcisi tarafından dönüştürülmez. İstemci, iletideki metni yalnızca bayt dizisi olarak alabilir. Kuyruk yöneticisi dönüşümü etkinleştirildiyse, metin kuyruk yöneticisi tarafından dönüştürülür, ancak JMS istemcisi bunu bir JMSBytesMessageiçinde bayt dizisi olarak almalıdır.

Bir JMSTextMessage ' in MQ ya da JMS gövde stiliyle gönderilip gönderilmeyeceğini denetlemek için WMQ_TARGET_DEST özelliğini kullanmak genellikle daha iyidir. Daha sonra iletiyi, WMQ_TARGET_DEST ayarı WMQ_TARGET_DEST_MQ ya da WMQ_TARGET_DEST_JMSolan bir hedeften alabilirsiniz. WMQ_TARGET_DEST alıcıyı etkilemez.

JMSMapMessage ve JMSStreamMessage

Bu iki JMS ileti tipi benzerdir. DataInputStream ve DataOutputStream arabirimlerine dayalı yöntemleri kullanarak iletileri okuyabilir ve iletilere temel tipler yazabilirsiniz; bkz. “İleti tipleri ve dönüştürme tipleri tablosu” sayfa 170. Ayrıntılar için bkz. “JMS istemcisi ileti dönüştürme ve kodlama” sayfa 171. Her temel öğe etiketlidir; bkz. “JMS ileti gövdesi” sayfa 157.

Sayısal veriler, XML metni olarak kodlanmış iletiye okunur ve yazılır. JMS_IBM_ENCODINGhedef özelliğine başvuru yapılmadı. Metin verileri, JMSTextMessageiçindeki metinle aynı şekilde işlenir. Şekil 20 sayfa 168örneğinde yaratılan ileti içeriğine bakarsanız, tüm ileti verileri 37 karakter takımı değeriyle gönderildiği gibi EBCDIC ' de olur.

Bir JMSMapMessage ya da JMSStreamMessageiçinde birden çok öğe gönderebilirsiniz.

Tek tek veri öğelerini bir JMSMapMessage'den adlarına göre ya da bir JMSStreamMessage' den konumlarına göre alabilirsiniz. İletide saklanan CodedCharacterSetId değeri kullanılarak bir get (alma) ya da read (okuma) yöntemi çağrıldığında her öğenin kodu çözülür. Öğeyi almak için kullanılan yöntem, gönderilen tipten farklı bir tip döndürürse, tip dönüştürülür. Tip dönüştürülemezse kural dışı durum yayınlanır. Ayrıntılar için bkz. Sınıf JMSStreamMessage . “JMSStreamMessage ve JMSMapMessage içinde veri gönderilmesi” sayfa 168 içindeki örnek, tip dönüşümünü ve JMSMapMessage içeriğinin sıradan çıkarılmasını gösterir.

JMSMapMessage ve JMSStreamMessage için MQRFH2.format alanı "MQSTR "olarak ayarlanır. WMQ_RECEIVE_CONVERSION hedef özelliği WMQ_RECEIVE_CONVERSION_QMGRolarak ayarlanırsa, ileti verileri JMS istemcisine gönderilmeden önce kuyruk yöneticisi tarafından dönüştürülür. İletinin MQRFH2.CodedCharacterSetId 'i hedefin WMQ_RECEIVE_CCSID ' idir. MQRFH2.Encoding , Native' dir. WMQ_RECEIVE_CONVERSION WMQ_RECEIVE_CONVERSION_CLIENT_MSG ise, MQRFH2 CodedCharacterSetId ve Encoding gönderen tarafından ayarlanan değerdir.

JMS istemci uygulaması, bir JMSMapMessage ya da JMSStreamMessage öğesini yalnızca JMSstili gövdesi olan bir iletide ve MQ biçemi gövdesini belirtmeyen bir hedeften alabilir.

JMSBytesMessage

JMSBytesMessage birden çok ilkel tip içerebilir. DataInputStream ve DataOutputStream arabirimlerine dayalı yöntemleri kullanarak iletileri okuyabilir ve iletilere temel tipler yazabilirsiniz; bkz. “İleti tipleri ve dönüştürme tipleri tablosu” sayfa 170. Ayrıntılar için bkz. “JMS ileti tipleri ve dönüştürmesi” sayfa 165.

İletideki sayısal verilerin kodlaması, sayısal verileri JMSBytesMessageeiçine yazmadan önce ayarlanan JMS_IBM_ENCODING değeriyle denetlenir. Bir uygulama, JMS_IBM_ENCODINGileti özelliğini ayarlayarak JMSBytesMessage için tanımlanan varsayılan Native kodlamasını geçersiz kılabilir.

Metin verileri, readUTF ve writeUTFkullanılarak UTF-8 içinde ya da readChar ve writeChar yöntemleri kullanılarak Unicode 'da okunabilir ve yazılabilir. CodedCharacterSetIdkullanan bir yöntem yoktur. Diğer bir seçenek olarak, JMS istemcisi Charset sınıfını kullanarak metni kodlayabilir ve byte olarak çözebilir. IBM MQ classes for JMS herhangi bir dönüştürme gerçekleştirmeden JVM ve ileti arasındaki baytları aktarır; bkz. “JMSBytesMessage içinde metin gönderme ve alma” sayfa 168.

Bir MQ uygulamasına gönderilen JMSBytesMessage , genellikle JMS MQRFH2 üstbilgisi olmadan MQstilinde bir ileti gövdesinde gönderilir. Bir JMS uygulamasına gönderilirse, ileti gövdesi stili genellikle JMSolur. Hedef özniteliklerin değeri olan WMQ_MESSAGE_BODY ve WMQ_TARGET_DEST , uygulama tarafından geçersiz kılınmadıkça, ileti gövdesi stilini belirler. Uygulama, destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ) ya da destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ)yöntemini çağırarak hedefte ayarlanan değerleri geçersiz kılabilir.

Bir JMSBytesMessage öğesini MQ biçemi gövdesiyle gönderirseniz, iletiyi MQ ya da JMS ileti gövdesi biçemini tanımlayan bir hedeften alabilirsiniz. JMS biçemi gövdesiyle bir JMSBytesMessage gönderirseniz, iletiyi JMS ileti gövdesi biçemini tanımlayan bir hedeften almanız gerekir. Bunu yapmazsanız, MQRFH2 kullanıcı iletisi verilerinin bir parçası olarak işlenir; bu, beklediğiniz gibi olmayabilir.

Bir iletinin MQ ya da JMS gövde stiline sahip olması, WMQ_TARGET_DESTayarından etkilenmez.

İleti verileri için bir Format belirtilirse ve kuyruk yöneticisi veri dönüştürmesi etkinleştirilirse, kuyruk yöneticisi tarafından ileti daha sonra dönüştürülebilir. Biçim alanını, ileti verilerinin biçimini belirtmekten başka bir şey için kullanmayın ya da alanı boş bırakın, MQConstants.MQFMT_NONE

Bir JMSBytesMessageeiçinde birden çok öğe gönderebilirsiniz. Her sayısal öğe, ileti için tanımlanan kodlama kullanılarak gönderildiğinde dönüştürülür.

JMSBytesMessage' den tek tek veri öğelerini alabilirsiniz. Okuma yöntemlerini, iletiyi yaratmak için çağrılan yazma yöntemleriyle aynı sırayla çağırın. İleti, iletide saklanan Encoding değeri kullanılarak çağrıldığında her sayısal öğe dönüştürülür.

JMSMapMessage ve JMSStreamMessage' in tersine, JMSBytesMessage yalnızca uygulama tarafından yazılan verileri içerir. JMSMapMessage ve JMSStreamMessageeiçindeki öğeleri tanımlamak için kullanılan XML etiketleri gibi ileti verilerinde ek veri saklanmaz. Bu nedenle, diğer uygulamalar için biçimlenmiş iletileri aktarmak üzere JMSBytesMessage seçeneğini kullanın.

JMSBytesMessage ve DataInputStream ve DataOutputStream arasında dönüştürme, bazı uygulamalarda yararlıdır. Örneğe dayalı kod (“DataInputStream ve DataOutputStream kullanarak iletileri okuma ve yazma” sayfa 169), JMSile com.ibm.mq.header paketini kullanmak için gereklidir.

Örnekler

JMSObjectMessage gönderme ve alma

```
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
```

Şekil 16. JMSObjectMessage gönderme ve alma

JMSTextmessage gönderme ve alma

Bir metin iletisi, farklı karakter kümelerinde metin içeremez. Örnek, iki farklı iletide gönderilen farklı karakter kümelerindeki metni gösterir.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Şekil 17. Hedefin tanımladığı karakter takımındaki metin iletisini gönder

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Şekil 18. `ccsid` 37 'de metin iletisi gönder

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Şekil 19. Metin iletisi al

JMSStreamMessage ve JMSMapMessage içinde veri gönderilmesi

```
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
```

Şekil 20. `JMSStreamMessage` ve `JMSMapMessage` içinde veri gönder

JMSBytesMessage içinde metin gönderme ve alma

Şekil 21 sayfa 169 içindeki kod, BytesMessageiçinde bir dizgi gönderir. Basitlik için örnek, `JMSTextMessage` ' in daha uygun olduğu tek bir dizgi gönderir. Bayt cinsinden bir metin dizesi almak için,

Şekil 22 sayfa 169içinde *TEXT_LENGTH* olarak adlandırılan bayt cinsinden dizginin uzunluğunu bilmeniz gerekir. Sabit sayıda karakter içeren bir dizgi için bile, bayt gösteriminin uzunluğu daha uzun olabilir.

```
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);
```

Şekil 21. JMSBytesMessage içinde String gönderilmesi

```
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);
```

Şekil 22. JMSBytesMessage olanağından String alınması

DataInputStream ve DataOutputStream kullanarak iletileri okuma ve yazma

Şekil 23 sayfa 169 içindeki kod, DataOutputStreamkullanarak bir JMSBytesMessage oluşturur.

```
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);
```

Şekil 23. DataOutputStream kullanarak JMSBytesMessage gönderme

JMS_IBM_ENCODING özelliğini ayarlayan deyim açıklama satırı yapıldı. Deyim, doğrudan bir JMSBytesMessage'a yazılırsa geçerlidir, ancak DataOutputStream' a yazılırken etkili olmaz. DataOutputStream kodlamasına yazılan sayılar Native kodlamasında kodlanır. JMS_IBM_ENCODING ayarının bir etkisi yoktur.

Şekil 24 sayfa 170 içindeki kod, DataInputStreamkullanarak bir JMSBytesMessage alır.

```
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)));
```

Şekil 24. `DataInputStream` kullanarak bir `JMSBytesMessage` alma

Kod sayfası, giriş iletisi verilerinin (JMS_IBM_CHARACTER_SET) kod sayfası özelliği kullanılarak yazılır. Girişte JMS_IBM_CHARACTER_SET , sayısal kodlanmış karakter takımı tanıtıcısı değil, bir Java kod sayfasıdır.

İleti tipleri ve dönüştürme tipleri tablosu

Çizelge 32. İleti tipleri ve dönüştürme tipleri

	Dönüştürme tipi			
İleti tipi	**Metin**	**Sayısal**	**Diğer**	**Yok**
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

Çizelge 32. İleti tipleri ve dönüştürme tipleri (devamı var)				
	Dönüştürme tipi			
İleti tipi	**Metin**	**Sayısal**	**Diğer**	**Yok**

`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`

İlgili kavramlar

JMS ileti dönüştürme yaklaşımları

JMS uygulama tasarımcılarına bir dizi veri dönüştürme yaklaşımı açıktır. Bu yaklaşımlar münhasır değildir; bazı uygulamaların bu yaklaşımların bir kombinasyonunu kullanması muhtemeldir. Uygulamanız yalnızca metin alışverişi yapıyorsa ya da yalnızca diğer JMS uygulamalarıyla ileti alışverişi yapıyorsa, normalde veri dönüştürmeyi düşünmezsiniz. Veri dönüştürme sizin için IBM MQtarafından otomatik olarak gerçekleştirilir.

JMS istemcisi ileti dönüştürme ve kodlama

JMS istemcisi ileti dönüştürme ve kodlamasını gerçekleştirmek için kullandığınız yöntemler, her dönüştürme tipine ilişkin kod örnekleriyle birlikte listelenir.

Kuyruk yöneticisi veri dönüştürme

Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alanJMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

İlgili görevler

JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi

Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları ve `JMSBytesMessage`kullanarakJMS dışı bir uygulamayla ileti değiş tokuşu yapabilen bir JMS istemci uygulamasını izleyin. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değiş tokuşu, veri dönüştürme çıkışı çağrılmasıyla ya da çağrılmadan gerçekleştirilebilir.

JMS istemcisi ileti dönüştürme ve kodlama

JMS istemcisi ileti dönüştürme ve kodlamasını gerçekleştirmek için kullandığınız yöntemler, her dönüştürme tipine ilişkin kod örnekleriyle birlikte listelenir.

Dönüştürme ve kodlama, Java temel öğeleri ya da nesneleri JMS iletilerine okunurken ya da bu iletilerden yazıldığında oluşur. Dönüştürme, kuyruk yöneticisi veri dönüştürme ve uygulama verileri dönüştürme işlemlerinden ayırt etmek için JMS istemci veri dönüştürme olarak adlandırılır. Dönüştürme,

JMS iletisinden veri okunduğunda ya da bir iletiye veri yazıldığında gerçekleşir. Metin, iç 16 bit Unicode gösterimine/gösteriminden dönüştürülür[3]iletilerdeki metin için kullanılan karakter kümesine. Sayısal veriler, ileti için tanımlanan kodlamaya ve Java temel sayısal tiplerine dönüştürülür. Dönüştürmenin gerçekleştirilip gerçekleştirilmeyeceği ve hangi dönüştürme tipinin gerçekleştirileceği, JMS ileti tipine ve okuma ya da yazma işlemine bağlıdır.

Çizelge 33 sayfa 172 , farklı JMS ileti tipleri için gerçekleştirilen dönüştürme tipine göre okuma ve yazma yöntemlerini kategorilere ayırır. Dönüştürme tipleri, çizelgeyi izleyen metinde açıklanır.

Çizelge 33. İleti tipleri ve dönüştürme tipleri

	Dönüştürme tipi			
İleti tipi	**Metin**	**Sayısal**	**Diğer**	**Yok**
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

[3] Bazı Unicode gösterimi 16 bitten fazla gerektirir. Java SE başvurusuna bakın.

Çizelge 33. İleti tipleri ve dönüştürme tipleri (devamı var)				
	Dönüştürme tipi			
İleti tipi	**Metin**	**Sayısal**	**Diğer**	**Yok**
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getByte getBytes readChar setByte setBytes setChar

Metin

Bir hedef için varsayılan CodedCharacterSetId , 1208, UTF-8değeridir. Varsayılan olarak, metin Unicode 'dan dönüştürülür ve UTF-8 metin dizgisi olarak gönderilir. Alma işlemi üzerinde, metin istemci tarafından alınan iletideki kodlanmış karakter kümesinden Unicode 'a dönüştürülür.

setText ve writeString yöntemleri, metni Unicode 'dan hedef için tanımlanan karakter kümesine dönüştürür. Bir uygulama, JMS_IBM_CHARACTER_SETileti özelliğini ayarlayarak hedef karakter kümesini geçersiz kılabilir. JMS_IBM_CHARACTER_SET, ileti gönderilirken sayısal bir kodlanmış karakter kümesi tanıtıcısı olmalıdır.[4].

"JMSTextmessage gönderme ve alma" sayfa 175 içindeki kod parçacıkları iki ileti gönderir. Biri hedef için tanımlanan karakter kümesinde, diğeri uygulama tarafından tanımlanan 37 karakter kümesinde gönderilir.

getText ve readString yöntemleri, iletideki metni iletide tanımlanan karakter kümesinden Unicode 'a dönüştürür. Yöntemler, JMS_IBM_CHARACTER_SETileti özelliğinde tanımlanan kod sayfasını kullanır. İleti MQtipinde bir ileti olmadığı ve MQRFH2olmadığı sürece, kod sayfası MQRFH2.CodedCharacterSetId ile eşlenir. İleti, MQRFH2olmadan MQtipinde bir iletiyse, kod sayfası MQMD.CodedCharacterSetIdile eşlenir.

Şekil 29 sayfa 176 içindeki kod parçacığı, hedefe gönderilen iletiyi alır. İletideki metin IBM037 kod sayfasından Unicode 'a geri dönüştürülür.

Not: Metnin 37 numaralı kodlanmış karakter kümesine dönüştürüldüğünü denetlemenin basit bir yolu IBM MQ Explorer 'ı kullanmaktır. Kuyruğa göz atın ve alınmadan önce iletinin özelliklerini gösterin.

Şekil 28 sayfa 175 içindeki kod parçacığını, Şekil 25 sayfa 173içindeki yanlış kod parçacığı ile karşılaştırın. Yanlış kod parçacığında, metin dizgisi biri uygulama tarafından, diğeri IBM MQtarafından olmak üzere iki kez dönüştürülür.

```
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);
```

Şekil 25. Yanlış kod sayfası dönüşümü

writeUTF yöntemi, metni Unicode 'dan 1208 'e (UTF-8) dönüştürür. Metin diziliminin başına 2 baytlık bir uzunluk konmuştur. Metin diziliminin uzunluk üst sınırı 65534 bayttır. readUTF yöntemi,

[4] JMS_IBM_CHARACTER_SET iletisi alınırken bir Java Charset kod sayfası adıdır.

writeUTF yöntemiyle yazılan bir iletide bir öğeyi okur. Tam olarak writeUTF yöntemiyle yazılan bayt sayısını okur.

Sayısal

Bir hedef için varsayılan sayısal kodlama şudur: Native. Java için Native kodlama değişmezi, tüm platformlar için aynı olan 273 x'00000111' değerine sahiptir. Alma işlemi üzerinde, iletideki sayılar doğru olarak sayısal Java ilkellerine dönüştürülür. Dönüştürme, iletide tanımlanan kodlamayı ve okuma yönteminin döndürdüğü tipi kullanır.

Gönderme yöntemi, set ve write tarafından bir iletiye eklenen sayıları hedef için tanımlanan sayısal kodlamaya dönüştürür. JMS_IBM_ENCODING ileti özelliğini ayarlayan bir uygulama tarafından bir ileti için hedef kodlama geçersiz kılınabilir; örneğin:

```
message.setIntProperty(WMQConstants.JMS_IBM_ENCODING,
WMQConstants.WMQ_ENCODING_INTEGER_REVERSED);
```

get ve read sayısal yöntemleri, iletideki sayıları iletide tanımlanan sayısal kodlamadan dönüştürür. Sayıları read ya da get yöntemiyle belirtilen tipe dönüştürür; bkz. ENCODING özelliği. Yöntemler, JMS_IBM_ENCODINGiçinde tanımlanan kodlamayı kullanır. İleti MQtipinde bir ileti değilse ve MQRFH2değilse, kodlama MQRFH2.Encoding ile eşlenir. İleti, MQRFH2olmadan MQtipinde bir iletiyse, yöntemler MQMD.Encodingiçinde tanımlanan kodlamayı kullanır.

Şekil 30 sayfa 176 örneğinde, hedef biçimde bir sayıyı kodlayan ve bir JMSStreamMessageiçinde gönderen bir uygulama gösterilmektedir. Şekil 30 sayfa 176 içindeki örneği, Şekil 31 sayfa 176içindeki örnekle karşılaştırın. Fark, JMS_IBM_ENCODING bir JMSBytesMessageiçinde ayarlanmalıdır.

Not: Sayının doğru şekilde kodlanıp kodlanmadığını kontrol etmenin basit bir yolu IBM MQ Explorer 'ı kullanmaktır. Kuyruğa göz atın ve kullanılmadan önce iletinin özelliklerini gösterin.

Diğer

boolean yöntemleri, bir JMSByteMessage, JMSStreamMessageve JMSMapMessageiçinde true ve false'yi x'01' ve x'00' olarak kodlar.

UTF yöntemleri, Unicode kodlarını UTF-8 metin dizgilerine kodlar ve kodlarını çözer. Dizgiler 65536 karakterden daha az karakterle sınırlıdır ve başında 2 baytlık uzunluk alanı bulunur.

Nesne yöntemleri, ilkel tipleri nesne olarak kaydırır. Sayısal ve metin tipleri, ilkel tipler sayısal ve metin yöntemleri kullanılarak okunmuş ya da yazılmış gibi kodlanır ya da dönüştürülür.

Yok

readByte, readBytes, readUnsignedByte, writeByteve writeBytes yöntemleri, uygulama ile ileti arasında dönüştürmeden tek bayt ya da bayt dizilerini alır ya da yerleştirir. readChar ve writeChar yöntemleri, uygulama ile ileti arasına dönüştürme olmadan 2 baytlık Unicode karakterleri alır ve koyar.

Uygulama, readBytes ve writeBytes yöntemlerini kullanarak “JMSBytesMessage içinde metin gönderme ve alma” sayfa 176içinde olduğu gibi kendi kod noktası dönüştürmesini gerçekleştirebilir.

IBM MQ , ileti bir JMSBytesMessageolduğundan ve readBytes ve writeBytes yöntemleri kullanıldığından istemcide herhangi bir kod sayfası dönüştürmesi gerçekleştirmez. Bununla birlikte, baytlar metni gösteriyorsa, uygulama tarafından kullanılan kod sayfasının hedefin kodlanmış karakter kümesiyle eşleştiğinden emin olun. İleti, bir kuyruk yöneticisi dönüştürme çıkışıyla yeniden dönüştürülebilir. Başka bir olasılık da, alan JMS istemci programının iletideki metni gösteren byte dizilerini, iletideki JMS_IBM_CHARACTER_SET özelliğini kullanarak dizgilere ya da karakterlere dönüştürme kuralına uymasıdır.

Bu örnekte istemci, dönüştürme işlemi için hedef kodlanmış karakter kümesini kullanır:

```
bytes.writeBytes("In the destination code page".getBytes(
CCSID.getCodepage(((MQDestination) destination)
.getIntProperty(WMQConstants.WMQ_CCSID))));
```

Diğer bir seçenek olarak, istemci bir kod sayfası seçip iletinin JMS_IBM_CHARACTER_SET özelliğinde ilgili kodlanmış karakter kümesini ayarlamış olabilir. IBM MQ classes for Java , MQRFH2içindeki JMS özelliklerinde ya da ileti tanımlayıcısında MQMD CodedCharacterSetId alanını ayarlamak için JMS_IBM_CHARACTER_SET komutunu kullanın:

```
String codePage = CCSID.getCodepage(37);
message.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage);
5
```

Bayt dizisi bir JMSStringMessage ya da JMSMapMessageiçine yazılırsa, baytlar JMSStringMessage ve JMSMapMessageiçinde metin olarak değil, onaltılı veri olarak yazıldığı için IBM MQ classes for JMS veri dönüştürme işlemi gerçekleştirmez.

Bayt 'lar uygulamanızdaki karakterleri gösteriyorsa, iletiyi okumak ve yazmak için hangi kod noktalarını dikkate almanız gerekir. Şekil 26 sayfa 175 içindeki kod, hedef kodlanmış karakter takımını kullanma kuralını izler. Dizgiyi JVM için varsayılan karakter kümesini kullanarak oluşturursanız, bayt içeriği altyapıya bağlıdır. Windows üzerindeki bir JVM genellikle windows-1252varsayılan Charset değerine sahiptir ve AIX and Linux UTF-8değerine sahiptir. Windowsve AIX and Linuxarasındaki değiş tokuş için, metni bayt olarak değiş tokuş etmek üzere belirtik bir kod sayfası seçmeniz gerekir.

```
StreamMessage smo = producer.session.createStreamMessage();
smo.writeBytes("123".getBytes(CCSID.getCodepage(((MQDestination) destination)
.getIntProperty(WMQConstants.WMQ_CCSID))));
```

Şekil 26. Hedef karakter kümesini kullanarak JMSStreamMessage içindeki bir dizgiyi gösteren bayt yazılması

Örnekler

JMSTextmessage gönderme ve alma

Bir metin iletisi, farklı karakter kümelerinde metin içeremez. Örnek, iki farklı iletide gönderilen farklı karakter kümelerindeki metni gösterir.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Şekil 27. Hedefin tanımladığı karakter takımındaki metin iletisini gönder

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText("Sent in EBCDIC character set 37");
producer.send(tmo);
```

Şekil 28. ccsid 37 'de metin iletisi gönder

[5] SetStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage) currently accepts only numeric character set identifiers.

```
TextMessage tmi = (TextMessage)consumer.receive();
System.out.println(tmi.getText());
...
Sent in the character set defined for the destination
```

Şekil 29. Metin iletisi al

Kodlama örnekleri

Kodlamada gönderilmekte olan bir sayıyı gösteren örnekler, bir hedef için tanımlar. Bir `JMSBytesMessage` öğesinin `JMS_IBM_ENCODING` özelliğini hedef için belirtilen değere ayarlamanız gerektiğine dikkat edin.

```
StreamMessage smo = session.createStreamMessage();
smo.writeInt(256);
producer.send(smo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println(smi.readInt());
...
256
```

Şekil 30. JMS`StreamMessage` içindeki hedef kodlamasını kullanarak sayı gönderme

```
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
```

Şekil 31. JMS`BytesMessage` içindeki hedef kodlamasını kullanarak sayı gönderme

JMSBytesMessage içinde metin gönderme ve alma

Şekil 32 sayfa 176 içindeki kod, BytesMessageiçinde bir dizgi gönderir. Basitlik için örnek, `JMSTextMessage` ' in daha uygun olduğu tek bir dizgi gönderir. Bayt cinsinden bir metin dizesi almak için, Şekil 33 sayfa 177içinde *TEXT_LENGTH* olarak adlandırılan bayt cinsinden dizginin uzunluğunu bilmeniz gerekir. Sabit sayıda karakter içeren bir dizgi için bile, bayt gösteriminin uzunluğu daha uzun olabilir.

```
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);
```

Şekil 32. JMS`BytesMessage` içinde `String` gönderilmesi

```
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);
```

Şekil 33. JMSBytesMessage olanağından String alınması

İlgili kavramlar

JMS ileti dönüştürme yaklaşımları

JMS uygulama tasarımcılarına bir dizi veri dönüştürme yaklaşımı açıktır. Bu yaklaşımlar münhasır değildir; bazı uygulamaların bu yaklaşımların bir kombinasyonunu kullanması muhtemeldir. Uygulamanız yalnızca metin alışverişi yapıyorsa ya da yalnızca diğer JMS uygulamalarıyla ileti alışverişi yapıyorsa, normalde veri dönüştürmeyi düşünmezsiniz. Veri dönüştürme sizin için IBM MQtarafından otomatik olarak gerçekleştirilir.

Kuyruk yöneticisi veri dönüştürme

Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alanJMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

İlgili görevler

JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi

Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları ve JMSBytesMessagekullanarakJMS dışı bir uygulamayla ileti değiş tokuşu yapabilen bir JMS istemci uygulamasını izleyin. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değiş tokuşu, veri dönüştürme çıkışı çağrılmasıyla ya da çağrılmadan gerçekleştirilebilir.

İlgili başvurular

JMS ileti tipleri ve dönüştürmesi

İleti tipi seçimi, ileti dönüştürme yaklaşımınızı etkiler. İleti dönüştürme ve ileti tipi etkileşimi, JMS ileti tipleri JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessageve JMSBytesMessageiçin açıklanır.

Kuyruk yöneticisi veri dönüştürme

Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alanJMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

Kuyruk yöneticisi, ileti verileri için ayarlanan CodedCharacterSetId, Encodingve Format değerlerini kullanarak ileti verilerindeki karakter ve sayısal verileri dönüştürebilir. JMS dışı uygulamalar için dönüştürme yeteneği her zaman GetMessageOption(GMO_CONVERT) ayarlanarak kullanılabilir olmuştur.

Kuyruk yöneticisi, JMS istemcilerine gönderilen iletileri dönüştürebilir. Kuyruk yöneticisi dönüşümü, WMQ_RECEIVE_CONVERSIONhedef özelliği WMQ_RECEIVE_CONVERSION_QMGRya da WMQ_RECEIVE_CONVERSION_CLIENT_MSGolarak ayarlanarak denetlenir. Uygulama hedef ayarını değiştirebilir:

```
((MQDestination)destination).setIntProperty(
              WMQConstants.WMQ_RECEIVE_CONVERSION,
              WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Ya da ...

```
((MQDestination)destination).setReceiveConversion
              (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

Şekil 34. Kuyruk yöneticisi veri dönüştürmesini etkinleştir

JMS istemcisi için kuyruk yöneticisi veri dönüşümü, istemci `consumer.receive` yöntemini çağırdığında gerçekleşir. Metin verileri varsayılan olarak UTF-8 ' e (1208) dönüştürülür. Sonraki okuma ve alma yöntemleri, iç Unicode kodlamasında Java metin temel öğeleri oluşturarak UTF-8kodlamasından alınan verilerdeki metni çözdü. UTF-8 , kuyruk yöneticisi veri dönüşümündeki tek hedef karakter kümesi değil. `WMQ_RECEIVE_CCSID` hedef özelliğini ayarlayarak farklı bir CCSID seçebilirsiniz.

Bir uygulama hedef ayarı da değiştirebilir; örneğin, 437, DOS-US olarak ayarlanabilir:

```
((MQDestination)destination).setIntProperty
(WMQConstants.WMQ_RECEIVE_CCSID, 437);
```

Ya da ...

```
((MQDestination)destination).setReceiveCCSID(437);
```

Şekil 35. Kuyruk yöneticisi dönüştürmesi için hedef kodlanmış karakter kümesini ayarla

`WMQ_RECEIVE_CCSID` değiştirmenin nedeni özelleştirilmiştir; seçilen CCSID, JVM ' de yaratılan metin nesneleri için fark yaratmaz. Ancak, bazı platformlardaki bazı JVM ' ler iletideki metnin CCSID 'sinden Unicode 'a dönüştürme işlemini işleyemeyebilir. Bu seçenek, iletide istemciye teslim edilen metinler için bir CCSID seçeneği sunar. Bazı JMS istemci altyapılarında, ileti metninin UTF-8biçiminde teslim edilmesiyle ilgili sorunlar ortaya çıktı.

JMS kodu, Şekil 36 sayfa 179içindeki C kodundaki kalın metne eşdeğerdir.

```
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                     */
```

Şekil 36. amqsget0.c kod parçacığı

Not:

Kuyruk yöneticisi dönüşümü yalnızca, bilinen bir IBM MQ biçimine sahip ileti verilerinde gerçekleştirilir. MQSTR, ya da MQCIH önceden tanımlanmış bilinen biçimlere örnektir. Bilinen bir biçim, veri dönüştürme çıkışı sağladığınız sürece, kullanıcı tanımlı bir biçim de olabilir.

JMSTextMessage, JMSMapMessage ve JMSStreamMessageolarak oluşturulan iletilerin MQSTR biçimi vardır ve kuyruk yöneticisi tarafından dönüştürülebilir.

İlgili kavramlar

JMS ileti dönüştürme yaklaşımları
JMS uygulama tasarımcılarına bir dizi veri dönüştürme yaklaşımı açıktır. Bu yaklaşımlar münhasır değildir; bazı uygulamaların bu yaklaşımların bir kombinasyonunu kullanması muhtemeldir. Uygulamanız yalnızca metin alışverişi yapıyorsa ya da yalnızca diğer JMS uygulamalarıyla ileti alışverişi yapıyorsa, normalde veri dönüştürmeyi düşünmezsiniz. Veri dönüştürme sizin için IBM MQtarafından otomatik olarak gerçekleştirilir.

JMS istemcisi ileti dönüştürme ve kodlama
JMS istemcisi ileti dönüştürme ve kodlamasını gerçekleştirmek için kullandığınız yöntemler, her dönüştürme tipine ilişkin kod örnekleriyle birlikte listelenir.

"Veri dönüştürme çıkışının çağrılması" sayfa 942
Veri dönüştürme çıkışı, bir MQGET çağrısının işlenmesi sırasında denetimi alan, kullanıcı tarafından yazılan bir çıkıştır.

İlgili görevler

JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi
Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları ve JMSBytesMessagekullanarakJMS dışı bir uygulamayla ileti değiş tokuşu yapabilen bir JMS istemci uygulamasını izleyin. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değiş tokuşu, veri dönüştürme çıkışı çağrılmasıyla ya da çağrılmadan gerçekleştirilebilir.

İlgili başvurular

JMS ileti tipleri ve dönüştürmesi
İleti tipi seçimi, ileti dönüştürme yaklaşımınızı etkiler. İleti dönüştürme ve ileti tipi etkileşimi, JMS ileti tipleri JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessageve JMSBytesMessageiçin açıklanır.

JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi
Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları ve JMSBytesMessagekullanarakJMS dışı bir uygulamayla ileti değiş tokuşu yapabilen bir JMS istemci

uygulamasını izleyin. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değiş tokuşu, veri dönüştürme çıkışı çağrılmasıyla ya da çağrılmadan gerçekleştirilebilir.

Başlamadan önce

`JMSTextMessage`kullanarakJMS olmayan bir uygulamayla ileti alışverişi için daha basit bir çözüm tasarlayabilirsiniz. Bu görevdeki adımları izlemeden önce bu olasılığı ortadan kaldırın.

Bu görev hakkında

JMS istemcisi, diğer JMS istemcileriyle değiş tokuş edilen JMS iletilerinin biçimlendirilmesine ilişkin ayrıntılarda yer almıyorsa, yazılması daha kolay olur. İleti tipi `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`ya da `JMSObjectMessage`olduğu sürece, IBM MQ iletiyi biçimlendirme ayrıntılarını görür. IBM MQ , farklı platformlardaki kod sayfalarındaki ve sayısal kodlamadaki farklılıklarla ilgilenir.

JMS dışı uygulamalarla ileti alışverişi yapmak için bu ileti tiplerini kullanabilirsiniz. Bunu yapmak için, bu iletilerin IBM MQ classes for JMStarafından nasıl oluşturulduğunu anlamanız gerekir. İletileri yorumlamak içinJMS dışı uygulamayı değiştirebilirsiniz; bkz. “JMS iletilerinin IBM MQ iletileriyle eşlenmesi” sayfa 142.

Bu ileti tiplerinden birini kullanmanın bir yararı, JMS istemci programlamasının ileti alışverişi yaptığı uygulamanın tipine bağlı olmamasıdır. Dezavantajı, başka bir programda değişiklik yapılmasını gerektirebilir ve diğer programı değiştiremeyebilirsiniz.

Diğer bir yaklaşım, var olan ileti biçimleriyle başa çıkabilen bir JMS istemci uygulaması yazmaktır. Genellikle var olan iletiler sabit biçimdedir ve biçimlendirilmemiş veri, metin ve sayıların bir karışımını içerir. JMS dışı uygulamalarla biçimlendirilmiş kayıtları değiş tokuş edebilen bir JMS istemcisi oluşturmak için başlangıç noktası olarak bu görevdeki adımları ve “JMSBytesMessage içindeki bir kayıt düzenini sarmalamak için sınıflar yazılması” sayfa 183içindeki örnek JMS istemcisini kullanın.

Yordam

1. Kayıt düzenini tanımlayın ya da önceden tanımlı IBM MQ üstbilgi sınıflarından birini kullanın.

   Önceden tanımlanmış IBM MQ üstbilgilerini işlemek için IBM MQ ileti üstbilgilerini işlemebaşlıklı konuya bakın.

   Şekil 37 sayfa 181 , veri dönüştürme yardımcı programı tarafından işlenebilen, kullanıcı tanımlı, değişmez uzunluklu kayıt düzeni örneğidir.
2. Veri dönüştürme çıkışını oluşturun.

   Bir veri dönüştürme çıkışı yazmak için Writing a data-conversion exit program başlıklı konudaki yönergeleri izleyin.

   Örneği “JMSBytesMessage içindeki bir kayıt düzenini sarmalamak için sınıflar yazılması” sayfa 183içinde denemek için veri dönüştürme çıkışını `MYRECORD`olarak adlandırın.
3. Kayıt düzenini sarmak ve kayıt göndermek ve almak için Java sınıflarını yazın. Alabildiğiniz iki yaklaşım şunlardır:
   - Kaydı içeren `JMSBytesMessage` öğesini okuyan ve yazan bir sınıf yazın; bkz. “JMSBytesMessage içindeki bir kayıt düzenini sarmalamak için sınıflar yazılması” sayfa 183.
   - Kaydın veri yapısını tanımlamak için `com.ibm.mq.header.Header` uzantılı bir sınıf yazın; bkz. Yeni üstbilgi tipleri için sınıf yaratılması.
4. Hangi kodlanmış karakter kümesinin ileti değiş tokuşu yapacağına karar verin.

   Bkz. İleti dönüştürmeye bir yaklaşım seçilmesi: Alıcı iyi yapar.
5. Hedefi, JMS `MQRFH2` üstbilgisi olmadan MQtipi iletileri değiş tokuş etmek üzere yapılandırın.

   Hem gönderme hem de alma hedefi, MQtipi iletileri değiş tokuş etmek üzere yapılandırılmalıdır. Gönderme ve alma için aynı hedefi kullanabilirsiniz.

   Uygulama, hedef ileti gövdesi özelliğini geçersiz kılabilir:

```
((MQDestination)destination).setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

"JMSBytesMessage içindeki bir kayıt düzenini sarmalamak için sınıflar yazılması" sayfa 183 içindeki örnek, MQstili bir iletinin gönderilmesini sağlayarak hedef ileti gövdesi özelliğini geçersiz kılar.

6. Çözümü JMS veJMS dışı uygulamalarla test edin

   Bir veri dönüştürme çıkışını sınamak için yararlı araçlar şunlardır:

   - `amqsgetc0.c` örnek programı, bir JMS istemcisi tarafından gönderilen bir iletiyi almayı sınamak için kullanışlıdır. Şekil 38 sayfa 182içindeki örnek üstbilgiyi (`RECORD.h`) kullanmak için önerilen değişikliklere bakın. Bu değişikliklerle, `amqsgetc0.c` örnek JMS istemcisi tarafından gönderilen bir ileti alır, `TryMyRecord.java` ; bkz. "JMSBytesMessage içindeki bir kayıt düzenini sarmalamak için sınıflar yazılması" sayfa 183.
   - Örnek IBM MQ göz atma programı `amqsbcg0.c`, ileti üstbilgisinin, JMS üstbilgisinin `MQRFH2`ve ileti içeriğinin incelenmesi için yararlıdır.
   - Daha önce SupportPac IH03' te bulunan **`rfhutil`** programı, sınama iletilerinin yakalanmasına ve dosyalarda saklanmasına ve daha sonra, İleti Akışlarını kullanmak için kullanılmasına olanak tanır. Çıkış iletileri çeşitli biçimlerde okunabilir ve görüntülenebilir. Biçimler, iki tip XML ' in yanı sıra bir COBOL copybook ile eşleştirmeyi içerir. Veriler EBCDIC ya da ASCII olabilir. İleti gönderilmeden önce iletiye RFH2 üstbilgisi eklenebilir.

   Değiştirilen `amqsgetc0.c` örnek programını kullanarak ileti almayı denerseniz ve `2080`neden koduyla bir hata alırsanız, iletinin bir `MQRFH2`olup olmadığını denetleyin. Bu değişiklikler, iletinin `MQRFH2`belirtilmeyen bir hedefe gönderildiğini varsayar.

Örnekler

```
struct RECORD { MQCHAR StrucID[4];
                MQLONG Version;
                               MQLONG StructLength;
                               MQLONG Encoding;
                               MQLONG CodeCharSetId;
                               MQCHAR Format[8];
                               MQLONG Flags;
                               MQCHAR RecordData[32];
        };
```

Şekil 37. RECORD.h

- RECORD.h veri yapısını bildir

```
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);
```

- MQGET çağrıyı kullanılacak şekilde değiştir RECORD,

  1. Değişiklikten önce:

```
    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. Değişiklikten sonra:

```
    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           */
```

- Yazdırma deyimini değiştirin,

  1. Kimden:

```
buffer[messlen] = '\0';            /* add terminator          */
printf("message <%s>\n", buffer);
```

  2. Kime:

```
/* buffer[messlen] = '\0';            add terminator          */
printf("ccsid <%d>, flags <%d>, message <%32.32s>\n \0",
        md.CodedCharSetId, record.Flags, record.RecordData);
```

Şekil 38. amqsget0.c değerini değiştir

İlgili kavramlar

JMS ileti dönüştürme yaklaşımları

JMS uygulama tasarımcılarına bir dizi veri dönüştürme yaklaşımı açıktır. Bu yaklaşımlar münhasır değildir; bazı uygulamaların bu yaklaşımların bir kombinasyonunu kullanması muhtemeldir. Uygulamanız yalnızca metin alışverişi yapıyorsa ya da yalnızca diğer JMS uygulamalarıyla ileti alışverişi yapıyorsa, normalde veri dönüştürmeyi düşünmezsiniz. Veri dönüştürme sizin için IBM MQtarafından otomatik olarak gerçekleştirilir.

JMS istemcisi ileti dönüştürme ve kodlama
JMS istemcisi ileti dönüştürme ve kodlamasını gerçekleştirmek için kullandığınız yöntemler, her dönüştürme tipine ilişkin kod örnekleriyle birlikte listelenir.

Kuyruk yöneticisi veri dönüştürme
Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alanJMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

Dönüştürme çıkış kodu yaratmak için kullanılan yardımcı program

İlgili başvurular

JMS ileti tipleri ve dönüştürmesi
İleti tipi seçimi, ileti dönüştürme yaklaşımınızı etkiler. İleti dönüştürme ve ileti tipi etkileşimi, JMS ileti tipleri `JMSObjectMessage`, `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`eve `JMSBytesMessage`eiçin açıklanır.

`JMSBytesMessage` içindeki bir kayıt düzenini sarmalamak için sınıflar yazılması
Bu görevin amacı, örneğin, bir `JMSBytesMessage`eiçinde veri dönüştürme ile sabit bir kayıt düzeninin nasıl birleştirileceğini araştırmaktır. Görevde, `JMSBytesMessage`eiçinde örnek bir kayıt yapısını değiş tokuş etmek için bazı Java sınıfları yaratırsınız. Diğer kayıt yapılarını değiş tokuş etmek üzere sınıfları yazmak için örneği değiştirebilirsiniz.

`JMSBytesMessage` ,JMS dışı programlarla karışık veri tipi kayıtlarını değiş tokuş etmek için JMS ileti tipinin en iyi seçimidir. JMS sağlayıcısı tarafından ileti gövdesine eklenen ek veri yoktur. Bu nedenle, bir JMS istemci programı var olan bir IBM MQ programıyla birlikte çalışıyorsa, bu ileti tipinin en iyi seçimidir. `JMSBytesMessage` kullanımındaki ana zorluk, diğer program tarafından beklenen kodlama ve karakter kümesini eşleştirmeyle birlikte gelir. Çözüm, kaydı çevreleyen bir sınıf oluşturmaktır. Belirli bir kayıt tipi için `JMSBytesMessage`' in okunmasını ve yazılmasını içeren bir sınıf, JMS programında sabit biçimli kayıtların gönderilmesini ve alınmasını kolaylaştırır. Soyut bir sınıfta arabirimin genel yönlerini yakalayarak, çözümün çoğu farklı kayıt biçimleri için yeniden kullanılabilir. Soyut soysal sınıfı genişleten sınıflarda farklı kayıt biçimleri uygulanabilir.

Alternatif bir yaklaşım, `com.ibm.mq.headers.Header` sınıfını genişletmektir. `Header` sınıfının, daha bildirimsel bir şekilde kayıt biçimi oluşturmak için `addMQLONG`gibi yöntemleri vardır. `Header` sınıfının kullanılmasının dezavantajı, öznitelikleri almanın ve ayarlamanın daha karmaşık bir yorumlama arabirimi kullanmaktır. Her iki yaklaşım da aynı miktarda uygulama koduyla sonuçlanır.

`JMSBytesMessage` , her kayıt aynı biçimi, kodlanmış karakter kümesini ve kodlamayı kullanmadığı sürece, `MQRFH2`' e ek olarak tek bir biçimi de tek bir iletide kapsayabilir. Bir `JMSBytesMessage` 'in biçimi, kodlaması ve karakter kümesi, `MQRFH2`' den sonraki tüm iletilerin özellikleridir. Bu örnek, `JMSBytesMessage` ' in tek bir kullanıcı kaydı içerdiği varsayımıyla yazılır.

Başlamadan önce

1. Beceri düzeyiniz: Java programlama ve JMShakkında bilgi sahibi olmanız gerekir. Java geliştirme ortamının ayarlanmasıyla ilgili yönergeler sağlanmaz. `JMSTextMessage`, `JMSStreamMessage`eya da `JMSMapMessage`dedeğiştokuşu için bir program yazmanız avantajlıdır. Daha sonra bir `JMSBytesMessage`kullanarak ileti alışverişindeki farklılıkları görebilirsiniz.
2. Örnek için IBM WebSphere MQ 7.0gereklidir.
3. Bu örnek, Eclipse çalışma ortamının Java perspektifi kullanılarak yaratılmıştır. JRE 6.0 ya da üstünü gerektirir. Java sınıflarını geliştirmek ve çalıştırmak için IBM MQ Explorer 'da Java perspektifini kullanabilirsiniz. Diğer bir seçenek olarak, kendi Java geliştirme ortamınızı kullanın.
4. IBM MQ Explorer 'ın kullanılması, test ortamını ve hata ayıklamayı komut satırı yardımcı programlarını kullanmaktan daha kolay hale getirir.

Bu görev hakkında

İki sınıf yaratma adımları boyunca size yol gösterecektir: RECORD ve MyRecord. Bu iki sınıf birlikte sabit biçimli bir kaydı kapsıyor. Öznitelikleri almak ve ayarlamak için yöntemleri vardır. get yöntemi kaydı JMSBytesMessage içinden okur ve koyma yöntemi bir kaydı JMSBytesMessageiçine yazar.

Görevin amacı, yeniden kullanabileceğiniz bir üretim kalitesi sınıfı yaratmaktır. Kendi sınıflarınıza başlamak için görevdeki örnekleri kullanmayı seçebilirsiniz. Görevin amacı, öncelikle bir JMSBytesMessagekullanırken karakter kümeleri, biçimleri ve kodlamaları kullanmayla ilgili olarak size yol gösterici notlar sağlamaktır. Sınıfların yaratılmasında her adım açıklanır ve bazen göz ardı edilen JMSBytesMessagekullanımı yönleri açıklanır.

RECORD sınıfı soyuttur ve bir kullanıcı kaydı için bazı ortak alanları tanımlar. Ortak alanlar, bir göz alıcı, bir sürüm ve bir uzunluk alanına sahip olmanın standart IBM MQ üstbilgi düzeninde modellenir. Birçok IBM MQ üstbilgisinde bulunan kodlama, karakter kümesi ve biçim alanları atlanır. Başka bir üstbilgi, kullanıcı tanımlı bir biçimi izleyemez. RECORD sınıfını genişleten MyRecord sınıfı, kaydı tam anlamıyla ek kullanıcı alanlarıyla genişleterek bunu yapar. Sınıflar tarafından yaratılan bir JMSBytesMessage, kuyruk yöneticisi veri dönüştürme çıkışı tarafından işlenebilir.

"Örneği çalıştırmak için kullanılan sınıflar" sayfa 190 , RECORD ve MyRecord' in tam listesini içerir. Ayrıca, RECORD ve MyRecord' yi test etmek için ek "iskele" sınıflarının listelerini de içerir. Ek sınıflar şunlardır:

TryMyRecord
RECORD ve MyRecordsınaması için ana program.

EndPoint
Tek bir sınıfta JMS bağlantısını, hedefini ve oturumunu saran soyut sınıf. Arabirimi yalnızca RECORD ve MyRecord sınıflarının sınanması gereksinimlerini karşılar. Bu, JMS uygulamalarının yazılması için oluşturulmuş bir tasarım kalıbı değildir.

Not: Endpoint sınıfı, bir hedef oluşturduktan sonra bu kod satırını içerir:

```
((MQDestination)destination).setReceiveConversion
                            (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

V7.0' da, V7.0.1.5sürümünden kuyruk yöneticisi dönüşümünü açmanız gerekir. Varsayılan olarak devre dışıdır. V7.0' da, varsayılan olarak en çok V7.0.1.4 kuyruk yöneticisi dönüşümü etkinleştirilir ve bu kod satırı hataya neden olur.

MyProducer ve MyConsumer
EndPoint' yi genişleten ve bir MessageConsumer ve MessageProduceroluşturan, bağlı ve kabul etmeye hazır sınıflar.

Tüm sınıflar bir JMSBytesMessageürününde veri dönüştürmeyi nasıl kullanacağınızı anlamak için oluşturabileceğiniz ve deneyleyebileceğiniz eksiksiz bir uygulama oluşturur.

Yordam

1. Varsayılan oluşturucuyla bir IBM MQ üstbilgisindeki standart alanları kapsüllemek için bir soyut sınıf yaratın. Daha sonra, üstbilgiyi gereksinimlerinize göre uyarlamak için sınıfı genişletebilirsiniz.

```
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();
    }
```

Not:

a. `structID` to `nextFormat`öznitelikleri, standart IBM MQ ileti üstbilgisine yerleştirildikleri sırayla listelenir.

b. `format`, `messageEncoding`ve `messageCharset`öznitelikleri üstbilginin kendisini açıklar ve üstbilginin bir parçası değildir.

c. Kaydın kodlanmış karakter takımı tanıtıcısının mı, yoksa karakter takımının mı saklanacağına karar vermeniz gerekir. Java , karakter takımlarını kullanır ve IBM MQ iletileri kodlanmış karakter takımı tanıtıcılarını kullanır. Örnek kod karakter kümelerini kullanır.

d. `int` , IBM MQtarafından `MQLONG` olarak diziselleştirilmiştir. `MQLONG` 4 bayttır.

2. Özel öznitelikler için alıcıları ve ayarlayıcıları yaratın.

a) Alıcıları yarat ya da oluştur:

```
     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) Ayarlayıcıları oluşturun ya da oluşturun:

```
     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. Bir `JMSBytesMessage`eiçinden RECORD yönetim ortamı yaratmak için bir oluşturucu yaratın.

```
  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());
  }
```

Not:

a. `messageCharset` ve `messageEncoding`, hedef için ayarlanan değerleri geçersiz kıldıkları için ileti özelliklerinden yakalanır. `format` güncellenmez. Örnek, hata denetimi yapmaz. `Record(BytesMessage)` oluşturucusu çağrılırsa, `JMSBytesMessage` öğesinin RECORD tipinde bir ileti olduğu varsayılır. Çizgi "`setStructID(new String(structID, getMessageCharset()))`" , göz alıcıyı ayarlar.

b. İletideki alanları dizisel biçimden geri çevirme yöntemini tamamlayan ve RECORD eşgörünümünde ayarlanan varsayılan değerleri güncelleyen kod satırları.

4. Üstbilgi alanlarını JMSBytesMessageiçine yazmak için bir koyma yöntemi oluşturun.

```
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;
}
```

Not:

a. MyProducer , JMS Connection, Destination, Sessionve MessageProducer öğelerini tek bir sınıfta birleştirir. Daha sonra kullanılan MyConsumer, JMS Connection, Destination, Sessionve MessageConsumer öğelerini tek bir sınıfta birleştirir.

b. Bir JMSBytesMessageiçin kodlama Nativedışında bir kodlamaysa, kodlamanın iletide ayarlanması gerekir. Hedef kodlama, ileti kodlama özniteliğine (JMS_IBM_CHARACTER_SET) kopyalanır ve RECORD sınıfının özniteliği olarak kaydedilir.

   i) "setMessageEncoding(myProducer.getEncoding());" , hedef kodlamasını almak için "(((MQDestination) destination).getIntProperty(WMQConstants.WMQ_ENCODING));" arar.

   ii) "Bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getMessageEncoding());" , ileti kodlamasını ayarlar.

c. Metni bayta dönüştürmek için kullanılan karakter kümesi hedeften alınır ve RECORD sınıfının özniteliği olarak kaydedilir. Bir JMSBytesMessageyazarken IBM MQ classes for JMS tarafından kullanılmadığından, iletide ayarlanmaz.

   "messageCharset = myProducer.getCharset();" çağrıları

```
public String getCharset() throws UnsupportedEncodingException,
        JMSException {
    return CCSID.getCodepage(getCCSID());
}
```

   Kodlanmış karakter takımı tanıtıcısından Java karakter kümesini alır.

   " CCSID.getCodepage(ccsid) " , pakette com.ibm.mq.headers. ccsid , hedefi sorgulayan MyProduceriçindeki başka bir yöntemden alınır:

```
public int getCCSID() throws JMSException {
    return (((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_CCSID));
}
```

d. "myProducer.setMQClient(true);" , istemci tipine ilişkin hedef ayarı geçersiz kılar ve IBM MQ MQI clienttipine zorlar. Bir yönetim yapılandırma hatasını gizlediği için bu kod satırını atlamak isteyebilirsiniz.

"myProducer.setMQClient(true);" aramaları:

```
((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ); }
if (!getMQDest()) setMQBody();
```

Kod, bir JMSayarını geçersiz kılması gerekiyorsa, IBM MQ gövde stilini belirtilmemiş olarak ayarlamanın yan etkisine sahiptir.

Not:

IBM MQ classes for JMS , iletinin biçimini, kodlamasını ve karakter kümesi tanıtıcısını ileti tanımlayıcısına (MQMD) ya da JMS üstbilgisine (MQRFH2) yazar. İletinin IBM MQ stili gövdesine sahip olup olmadığına bağlıdır. MQMD alanlarını el ile ayarlamayın.

İleti tanımlayıcı özelliklerini el ile ayarlamak için bir yöntem vardır. JMS_IBM_MQMD_* özelliklerini kullanır. JMS_IBM_MQMD_* özelliklerini ayarlamak için WMQ_MQMD_WRITE_ENABLED hedef özelliğini ayarlamanız gerekir:

```
((MQDestination)destination).setMQMDWriteEnabled(true);
```

Özellikleri okumak için hedef özelliği (WMQ_MQMD_READ_ENABLED) ayarlamanız gerekir.

JMS_IBM_MQMD_* ' i yalnızca tüm ileti bilgi yükü üzerinde tam denetim alırsanız kullanın. JMS_IBM_* özelliklerinden farklı olarak, JMS_IBM_MQMD_* özellikleri IBM MQ classes for JMS ' in bir JMS iletisini nasıl oluşturduğunu denetleyemez. JMS iletisinin özellikleriyle çakışan ileti tanımlayıcı özellikleri yaratılabilir.

e. Yöntemi tamamlayan kod satırları, sınıftaki öznitelikleri iletideki alanlar olarak serileştirir.

Dizilim öznitelikleri boşluklarla doldurulur. Dizgiler, kayıt için tanımlanan karakter kümesi kullanılarak baytlara dönüştürülür ve ileti alanlarının uzunluğuna kesilir.

5. İçe aktarmaları ekleyerek sınıfı tamamlayın.

```
 package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import jakarta.jms.BytesMessage;
import jakarta.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;
```

6. RECORD sınıfını ek alanlar içerecek şekilde genişletmek için bir sınıf oluşturun. Varsayılan bir oluşturucu ekleyin.

```
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);
   }
```

Not:

a. RECORD Alt sınıfı (MyRecord), üstbilginin göz alıcısını, biçimini ve uzunluğunu özelleştirir.

7. Alıcıları ve ayarlayıcıları oluşturun ya da oluşturun.
   a) Alıcıları yarat:

```
    public int getFlags() { return flags; }
   public String getRecordData() { return recordData; } .
```

   b) Ayarlayıcıları yarat:

```
    public void setFlags(int flags) {
       this.flags = flags; }
   public void setRecordData(String recordData) {
       this.recordData = recordData; }
}
```

8. Bir JMSBytesMessageiçinden MyRecord yönetim ortamı yaratmak için bir oluşturucu yaratın.

```
 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()));
}
```

   Not:

   a. Standart ileti şablonunu oluşturan alanlar önce RECORD sınıfı tarafından okunur.

   b. recordData metni, iletinin karakter kümesi özelliği kullanılarak String biçimine dönüştürülür.

9. Bir tüketiciden ileti almak ve yeni bir MyRecord eşgörünümü yaratmak için durağan bir yöntem yaratın.

```
 public static MyRecord get(MyConsumer myConsumer) throws JMSException,
        MQDataException, IOException {
    BytesMessage message = (BytesMessage) myConsumer.receive();
    return new MyRecord(message);
}
```

   Not:

   a. Örnekte, MyRecord(BytesMessage) oluşturucusu durağan alma yönteminden çağrılır. Genellikle, iletiyi almayı yeni bir MyRecord yönetim ortamı yaratmaktan ayırabilirsiniz.

10. İleti üstbilgisi içeren bir JMSBytesMessage içine müşteri alanlarını eklemek için bir koyma yöntemi oluşturun.

```
 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;
}
```

   Not:

   a. Bu yöntem, kod içindeki MyRecord sınıfındaki öznitelikleri iletideki alanlar olarak serileştirir.

- `recordData` `String` özniteliği boşluklarla doldurulur, kayıt için tanımlanan karakter kümesi kullanılarak baytlara dönüştürülür ve `RecordData` alanlarının uzunluğuna kısaltılır.

11. İçerme deyimlerini ekleyerek sınıfı tamamlayın.

```
 package com.ibm.mq.id;
import java.io.IOException;
import jakarta.jms.BytesMessage;
import jakarta.jms.JMSException;
import com.ibm.mq.headers.MQDataException;
```

Sonuçlar

- `TryMyRecord` sınıfını çalıştırmanın sonuçları:
  - İleti 37 numaralı kodlanmış karakter takımıyla ve bir kuyruk yöneticisi dönüştürme çıkışını kullanarak gönderiliyor:

```
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
```

  - İleti 37 numaralı kodlanmış karakter takımıyla gönderiliyor ve kuyruk yöneticisi dönüştürme çıkışı kullanılmıyor:

```
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
```

- `TryMyRecord` sınıfının değiştirilmesinin sonuçları, iletiyi almayacak ve bunun yerine değiştirilen `amqsget0.c` örneği kullanılarak alınacak. Değiştirilen örnek biçimlendirilmiş bir kaydı kabul eder; bkz. “JMS dışı bir uygulamayla biçimlendirilmiş bir kaydın değiştirilmesi” sayfa 179içinde Şekil 38 sayfa 182 .
  - İleti 37 numaralı kodlanmış karakter takımıyla ve bir kuyruk yöneticisi dönüştürme çıkışını kullanarak gönderiliyor:

```
Sample AMQSGET0 start
ccsid <850>, flags <1>, message <ABCDEFGHIJKLMNOPQRSTUVWXYZ012345>
no more messages
Sample AMQSGET0 end
```

  - İleti 37 numaralı kodlanmış karakter takımıyla gönderiliyor ve kuyruk yöneticisi dönüştürme çıkışı kullanılmıyor:

```
Sample AMQSGET0 start
MQGET ended with reason code 2110
ccsid <37>, flags <1>, message <--+-+ãÃ++ÐÊËÈiÐÎÐ+ÔÒõõµþÞÚ-±=¾¶§>
no more messages
Sample AMQSGET0 end
```

Örneği denemek ve farklı kod sayfaları ve bir veri dönüştürme çıkışıyla denemek için. Java sınıflarını oluşturun, IBM MQ' ı yapılandırın ve ana programı çalıştırın, `TryMyRecord` ; bkz. “#unique_196/unique_196_Connect_42_Try” sayfa 190.

1. Örneği çalıştırmak için IBM MQ ve JMS ' i yapılandırın. Yönergeler, örneğin Windowsüzerinde çalıştırılmasına ilişkin yönergelerdir.
   a. Kuyruk yöneticisi yarat

```
crtmqm -sa -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1
```

b. Kuyruk yarat

```
echo DEFINE QL('Q1') REPLACE | runmqsc QM1
```

c. JNDI dizini yarat

```
cd c:\
md JNDI-Directory
```

d. JMS bin dizinine geç

JMS Denetim programı buradan çalıştırılmalıdır. Yol: *MQ_INSTALLATION_PATH*\java\bin.

e. JMSQM1Q1.txt adlı bir dosyada aşağıdaki JMS tanımlamalarını oluşturun

```
DEF CF(QM1) PROVIDERVERSION(7) QMANAGER(QM1)
DEF Q(Q1) CCSID(37) ENCODING(RRR) MSGBODY(MQ) QMANAGER(QM1) QUEUE(Q1) TARGCLIENT(MQ)
VERSION(7)
END
```

f. JMS kaynaklarını yaratmak için JMSAdmin programını çalıştırın.

```
JMSAdmin < JMSQM1Q1.txt
```

2. IBM MQ Gezgini 'ni kullanarak yarattığınız tanımlamaları yaratabilir, değiştirebilir ve bunlara göz atabilirsiniz.
3. TryMyRecordkomutunu çalıştırın.

Örneği çalıştırmak için kullanılan sınıflar

Aşağıdaki kod öbeklerinde listelenen sınıflar sıkıştırılmış bir dosyada da bulunur. jm25529_.zip ya da jm25529_.tar.gzdosyasını karşıdan yükleyin.

TryMyRecord

```
package com.ibm.mq.id;
public class TryMyRecord {
    public static void main(String[] args) throws Exception {
        MyProducer producer = new MyProducer();
        MyRecord outrec = new MyRecord();
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        outrec.put(producer);
        System.out.println("Out flags " + outrec.getFlags() + " text "
                + outrec.getRecordData() + " Encoding "
                + producer.getEncoding() + " CCSID " + producer.getCCSID()
                + " MQ " + producer.getMQDest());
        MyRecord inrec = MyRecord.get(new MyConsumer());
        System.out.println("In  flags " + inrec.getFlags() + " text "
                + inrec.getRecordData() + " Encoding "
                + inrec.getMessageEncoding() + " CCSID "
                + inrec.getMessageCharset());
    }
}
```

RECORD

V 9.3.0 V 9.3.0 JM 3.0

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import jakarta.jms.BytesMessage;
import jakarta.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
```

```
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;

public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }

    public RECORD(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super();
        setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
        setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
        byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
        message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
        setStructID(new String(structID, getMessageCharset()));
        setVersion(message.readInt());
        setStructLength(message.readInt());
    }

    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }


    protected BytesMessage put(MyProducer myProducer) throws IOException,
            JMSException, UnsupportedEncodingException {
        setHeaderEncoding(myProducer.getEncoding());
        setHeaderCharset(myProducer.getCharset());
        myProducer.setMQClient(true);
        BytesMessage bytes = myProducer.session.createBytesMessage();
        bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
        bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
        bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                             myProducer.getCCSID());
        bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                         + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                         .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
        bytes.writeInt(getVersion());
        bytes.writeInt(getStructLength());
        return bytes;
    }

    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

JMS 2.0

```
package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
```

```
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.constants.MQConstants;
import com.ibm.mq.headers.MQDataException;
import com.ibm.msg.client.wmq.WMQConstants;
public abstract class RECORD implements Serializable {
    private static final long serialVersionUID = -1616617232750561712L;
    protected final static int UTF8 = 1208;
    protected final static int MQLONG_LENGTH = 4;
    protected final static int RECORD_STRUCT_ID_LENGTH = 4;
    protected final static int RECORD_VERSION_1 = 1;
    protected final String RECORD_STRUCT_ID = "BLNK";
    protected final String RECORD_TYPE = "BLANK   ";
    private String structID = RECORD_STRUCT_ID;
    private int version = RECORD_VERSION_1;
    private int structLength = RECORD_STRUCT_ID_LENGTH + MQLONG_LENGTH * 2;
    private int headerEncoding = WMQConstants.WMQ_ENCODING_NATIVE;
    private String headerCharset = "UTF-8";
    private String headerFormat = RECORD_TYPE;

    public RECORD() {
        super();
    }
    public RECORD(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super();
        setHeaderCharset(message.getStringProperty(WMQConstants.JMS_IBM_CHARACTER_SET));
        setHeaderEncoding(message.getIntProperty(WMQConstants.JMS_IBM_ENCODING));
        byte[] structID = new byte[RECORD_STRUCT_ID_LENGTH];
        message.readBytes(structID, RECORD_STRUCT_ID_LENGTH);
        setStructID(new String(structID, getMessageCharset()));
        setVersion(message.readInt());
        setStructLength(message.readInt());
    }

    public String getHeaderFormat() { return headerFormat; }
    public int getHeaderEncoding() { return headerEncoding;    }
    public String getMessageCharset() { return headerCharset; }
    public int getMessageEncoding() { return headerEncoding; }
    public String getStructID() { return structID; }
    public int getStructLength() { return structLength; }
    public int getVersion() { return version; }

    protected BytesMessage put(MyProducer myProducer) throws IOException,
            JMSException, UnsupportedEncodingException {
        setHeaderEncoding(myProducer.getEncoding());
        setHeaderCharset(myProducer.getCharset());
        myProducer.setMQClient(true);
        BytesMessage bytes = myProducer.session.createBytesMessage();
        bytes.setStringProperty(WMQConstants.JMS_IBM_FORMAT, getHeaderFormat());
        bytes.setIntProperty(WMQConstants.JMS_IBM_ENCODING, getHeaderEncoding());
        bytes.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET,
                             myProducer.getCCSID());
        bytes.writeBytes(String.format("%1$-" + RECORD_STRUCT_ID_LENGTH + "."
                         + RECORD_STRUCT_ID_LENGTH + "s", getStructID())
                         .getBytes(getMessageCharset()), 0, RECORD_STRUCT_ID_LENGTH);
        bytes.writeInt(getVersion());
        bytes.writeInt(getStructLength());
        return bytes;
    }

    public void setHeaderCharset(String charset) {
        this.headerCharset = charset; }
    public void setHeaderEncoding(int encoding) {
        this.headerEncoding = encoding; }
    public void setHeaderFormat(String headerFormat) {
        this.headerFormat = headerFormat; }
    public void setStructID(String structID) {
        this.structID = structID; }
    public void setStructLength(int structLength) {
        this.structLength = structLength; }
    public void setVersion(int version) {
        this.version = version; }
}
```

MyRecord

V 9.3.0 V 9.3.0 JM 3.0

```
package com.ibm.mq.id;
import java.io.IOException;
```

```
import jakarta.jms.BytesMessage;
import jakarta.jms.JMSException;
import com.ibm.mq.headers.MQDataException;

public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }

    public MyRecord(BytesMessage message) throws JMSException, IOException,
            MQDataException {
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }

    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }

    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; } .


    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                        + DATA_LENGTH + "s",getRecordData())
                        .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }

    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

JMS 2.0

```
package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSException;
import com.ibm.mq.headers.MQDataException;
public class MyRecord extends RECORD {
    private static final long serialVersionUID = -370551723162299429L;
    private final static int FLAGS = 1;
    private final static String STRUCT_ID = "MYRD";
    private final static int DATA_LENGTH = 32;
    private final static String FORMAT = "MYRECORD";
    private int flags = FLAGS;
    private String recordData = "ABCDEFGHIJKLMNOPQRSTUVWXYZ012345";

    public MyRecord() {
        super();
        super.setStructID(STRUCT_ID);
        super.setHeaderFormat(FORMAT);
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH
                + DATA_LENGTH);
    }
    public MyRecord(BytesMessage message) throws JMSException, IOException,
```

```
            MQDataException {
        super(message);
        setFlags(message.readInt());
        byte[] recordData = new byte[DATA_LENGTH];
        message.readBytes(recordData, DATA_LENGTH);
        setRecordData(new String(recordData, super.getMessageCharset()));
    }
    public static MyRecord get(MyConsumer myConsumer) throws JMSException,
            MQDataException, IOException {
        BytesMessage message = (BytesMessage) myConsumer.receive();
        return new MyRecord(message);
    }
    public int getFlags() { return flags; }
    public String getRecordData() { return recordData; } .

    public BytesMessage put(MyProducer myProducer) throws JMSException,
            IOException {
        BytesMessage bytes = super.put(myProducer);
        bytes.writeInt(getFlags());
        bytes.writeBytes(String.format("%1$-" + DATA_LENGTH + "."
                        + DATA_LENGTH + "s",getRecordData())
                        .getBytes(super.getMessageCharset()), 0, DATA_LENGTH);
        myProducer.send(bytes);
        return bytes;
    }
    public void setFlags(int flags) {
        this.flags = flags; }
    public void setRecordData(String recordData) {
        this.recordData = recordData; }
}
```

EndPoint

V 9.3.0 V 9.3.0 JM 3.0

```
package com.ibm.mq.id;
import java.io.UnsupportedEncodingException;
import jakarta.jms.Connection;
import jakarta.jms.ConnectionFactory;
import jakarta.jms.Destination;
import jakarta.jms.JMSException;
import jakarta.jms.Session;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import com.ibm.mq.headers.CCSID;
import com.ibm.mq.jms.MQDestination;
import com.ibm.msg.client.wmq.WMQConstants;
public abstract class EndPoint {
    public Context ctx;
    public ConnectionFactory cf;
    public Connection connection;
    public Destination destination;
    public Session session;
    protected EndPoint() throws NamingException, JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup("QM1");
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup("Q1");
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); }
    protected EndPoint(String cFactory, String dest) throws NamingException,
            JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup(cFactory);
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup(dest);
        ((MQDestination)destination).setReceiveConversion
                                    (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);  }
    public int getCCSID() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_CCSID));  }
    public String getCharset() throws UnsupportedEncodingException,
```

```
            JMSException {
        return CCSID.getCodepage(getCCSID());  }
    public int getEncoding() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_ENCODING));  }
    public boolean getMQDest() throws JMSException {
        if ((((MQDestination) destination).getMessageBodyStyle()
               == WMQConstants.WMQ_MESSAGE_BODY_MQ)
            || ((((MQDestination) destination).getMessageBodyStyle()
                   == WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED)
               && (((MQDestination) destination).getTargetClient()
                   == WMQConstants.WMQ_TARGET_DEST_MQ)))
            return true;
        else
            return false; }
    public void setCCSID(int ccsid) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_CCSID,
                ccsid); }
    public void setEncoding(int encoding) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_ENCODING,
                encoding); }
    public void setMQBody() throws JMSException {
        ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);  }
    public void setMQBody(boolean mqbody) throws JMSException {
        if (mqbody)  ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
        else         ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_JMS);  }
    public void setMQClient(boolean mqclient) throws JMSException {
       if (mqclient){
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
            if (!getMQDest()) setMQBody();
        }
       else
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_JMS); }
    }
```

JMS 2.0

```
package com.ibm.mq.id;
import java.io.UnsupportedEncodingException;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Session;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import com.ibm.mq.headers.CCSID;
import com.ibm.mq.jms.MQDestination;
import com.ibm.msg.client.wmq.WMQConstants;
public abstract class EndPoint {
    public Context ctx;
    public ConnectionFactory cf;
    public Connection connection;
    public Destination destination;
    public Session session;
    protected EndPoint() throws NamingException, JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup("QM1");
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup("Q1");
        ((MQDestination)destination).setReceiveConversion
                                     (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); }
    protected EndPoint(String cFactory, String dest) throws NamingException,
            JMSException {
        System.setProperty("java.naming.provider.url", "file:/C:/JNDI-Directory");
        System.setProperty("java.naming.factory.initial",
                           "com.sun.jndi.fscontext.RefFSContextFactory");
        ctx = new InitialContext();
        cf = (ConnectionFactory) ctx.lookup(cFactory);
        connection = cf.createConnection();
        destination = (Destination) ctx.lookup(dest);
        ((MQDestination)destination).setReceiveConversion
                                     (WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

```
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);  }
    public int getCCSID() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_CCSID));  }
    public String getCharset() throws UnsupportedEncodingException,
            JMSException {
        return CCSID.getCodepage(getCCSID());  }
    public int getEncoding() throws JMSException {
        return (((MQDestination) destination)
                .getIntProperty(WMQConstants.WMQ_ENCODING));  }
    public boolean getMQDest() throws JMSException {
        if ((((MQDestination) destination).getMessageBodyStyle()
               == WMQConstants.WMQ_MESSAGE_BODY_MQ)
            || ((((MQDestination) destination).getMessageBodyStyle()
                   == WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED)
               && (((MQDestination) destination).getTargetClient()
                   == WMQConstants.WMQ_TARGET_DEST_MQ)))
            return true;
        else
            return false; }
    public void setCCSID(int ccsid) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_CCSID,
                ccsid); }
    public void setEncoding(int encoding) throws JMSException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_ENCODING,
                encoding); }
    public void setMQBody() throws JMSException {
        ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);  }
    public void setMQBody(boolean mqbody) throws JMSException {
        if (mqbody)  ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
        else         ((MQDestination) destination)
                    .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_JMS);  }
    public void setMQClient(boolean mqclient) throws JMSException {
       if (mqclient){
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
            if (!getMQDest()) setMQBody();
        }
       else
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_JMS); }
    }
```

MyProducer

V 9.3.0 V 9.3.0 JM 3.0

```
package com.ibm.mq.id;
import jakarta.jms.JMSException;
import jakarta.jms.Message;
import jakarta.jms.MessageProducer;
import javax.naming.NamingException;
public class MyProducer extends EndPoint {
    public MessageProducer producer;
    public MyProducer() throws NamingException, JMSException {
        super();
        producer = session.createProducer(destination); }
    public MyProducer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        producer = session.createProducer(destination); }
    public void send(Message message) throws JMSException {
        producer.send(message); }
}
```

JMS 2.0

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageProducer;
import javax.naming.NamingException;
public class MyProducer extends EndPoint {
    public MessageProducer producer;
    public MyProducer() throws NamingException, JMSException {
        super();
        producer = session.createProducer(destination); }
    public MyProducer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
```

```
        producer = session.createProducer(destination); }
    public void send(Message message) throws JMSException {
        producer.send(message); }
}
```

MyConsumer

V 9.3.0 V 9.3.0 JM 3.0

```
package com.ibm.mq.id;
import jakarta.jms.JMSException;
import jakarta.jms.Message;
import jakarta.jms.MessageConsumer;
import javax.naming.NamingException;
public class MyConsumer extends EndPoint {
    public MessageConsumer consumer;
    public MyConsumer() throws NamingException, JMSException {
        super();
        consumer = session.createConsumer(destination);
        connection.start();  }
    public MyConsumer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        consumer = session.createConsumer(destination);
        connection.start();  }
    public Message receive() throws JMSException {
        return consumer.receive();  }
}
```

JMS 2.0

```
package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.naming.NamingException;
public class MyConsumer extends EndPoint {
    public MessageConsumer consumer;
    public MyConsumer() throws NamingException, JMSException {
        super();
        consumer = session.createConsumer(destination);
        connection.start();  }
    public MyConsumer(String cFactory, String dest) throws NamingException,
            JMSException {
        super(cFactory, dest);
        consumer = session.createConsumer(destination);
        connection.start();  }
    public Message receive() throws JMSException {
        return consumer.receive();  }
}
```

Bağlantı üreticileri ve hedefleri oluşturma ve yapılandırma

Bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging uygulaması, bunları bir Java Naming and Directory Interface (JNDI) ad alanından yönetilen nesneler olarak alarak, IBM JMS uzantılarını kullanarak ya da IBM MQ JMS uzantılarını kullanarak bağlantı üreticileri ve hedefleri yaratabilir. Bir uygulama, bağlantı üreticileri ve hedeflerinin özelliklerini ayarlamak için IBM JMS uzantılarını ya da IBM MQ JMS uzantılarını da kullanabilir.

Bağlantı üreticileri ve hedefler, JMS ya da Jakarta Messaging uygulamasının mantık akışında başlangıç noktalarıdır. Bir uygulama, bir ileti alışverişi sunucusuna bağlantı yaratmak için ConnectionFactory nesnesini kullanır ve ileti göndermek için hedef ya da ileti almak için kaynak olarak bir Kuyruk ya da Konu nesnesini kullanır. Bu nedenle bir uygulamanın en az bir bağlantı üreticisi ve bir ya da daha çok hedef yaratması gerekir. Bir bağlantı üreticisi ya da hedefi yarattıktan sonra, uygulamanın bir ya da daha çok özelliğini ayarlayarak nesneyi yapılandırması gerekebilir.

Özet olarak, bir uygulama aşağıdaki şekillerde bağlantı üreticilerini ve hedeflerini oluşturabilir ve yapılandırabilir:

Yönetilen nesneleri almak için JNDI ' yi kullanma

Bir denetimci, bir JNDI ad alanında denetlenen nesneler olarak bağlantı üreticileri ve hedefleri yaratmak ve yapılandırmak için IBM MQ JMS denetim aracını Yönetim araçlarını kullanarak JMS ve Jakarta Messaging nesnelerini yapılandırmaya da IBM MQ Explorer başlıklı konuda açıklandığı

gibi kullanabilir. JMS 2.0 nesnelerini IBM MQ Explorerkullanarak yapılandırma ve yapılandırma. Bir uygulama, JNDI ad alanından denetlenen nesneleri alabilir. Yönetilen bir nesneyi alan uygulama, gerekiyorsa, IBM JMS uzantılarını ya da IBM MQ JMS uzantılarını kullanarak bir ya da daha fazla özelliğini ayarlayabilir ya da değiştirebilir.

Not: V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging 3.0için, IBM MQ Explorerkullanarak JNDI ' yi yönetemezsiniz. JNDI yönetimi, **JMSAdmin**' ın **JMS30Admin**olan Jakarta Messaging 3.0 çeşitlemesi tarafından desteklenir.

IBM JMS uzantılarının kullanılması

Bir uygulama, çalıştırma zamanında dinamik olarak bağlantı üreticileri ve hedefler oluşturmak için IBM JMS uzantılarını kullanabilir. Uygulama önce bir JmsFactoryFactory nesnesi yaratır ve bağlantı üreticileri ve hedefler yaratmak için bu nesnenin yöntemlerini kullanır. Bir bağlantı üreticisi ya da hedefi yaratmış olan uygulama, özelliklerini ayarlamak için JmsPropertybağlam arabiriminden edinilen yöntemleri kullanabilir. Diğer bir seçenek olarak, uygulama hedefi yaratırken hedefin bir ya da daha çok özelliğini belirtmek için tek tip kaynak tanıtıcısını (URI) kullanabilir.

IBM MQ JMS uzantılarının kullanılması

Bir uygulama, çalıştırma zamanında dinamik olarak bağlantı üreticileri ve hedefler oluşturmak için IBM MQ JMS uzantılarını da kullanabilir. Uygulama, bağlantı üreticileri ve hedefler yaratmak için sağlanan oluşturucuları kullanır. Bir bağlantı üreticisi ya da hedefi yaratmış olan uygulama, özelliklerini ayarlamak için nesnenin yöntemlerini kullanabilir. Diğer bir seçenek olarak, uygulama hedefi yaratırken hedefin bir ya da daha çok özelliğini belirtmek için bir URI kullanabilir.

İlgili görevler

JMS ve Jakarta Messaging kaynaklarının yapılandırılması

Bir JMS ya da Jakarta Messaging uygulamasında denetlenen nesneleri almak için JNDI ' yi kullanma

Java Adlandırma ve Dizin Arabirimi (JNDI) ad alanından yönetilen nesneleri almak için bir JMS ya da Jakarta Messaging uygulaması bir başlangıç bağlamı yaratmalı ve nesneleri almak için lookup () yöntemini kullanmalıdır.

Bir uygulamanın JNDI ad alanından denetlenen nesneleri alabilmesi için önce bir yöneticinin denetlenen nesneleri yaratması gerekir.

JMS 2.0 JMS 2.0için yönetici, bir JNDI ad alanında yönetilen nesneler yaratmak ve bunların bakımını yapmak için IBM MQ JMS yönetim aracını **JMSAmin**ya da IBM MQ Explorer kullanabilir. Daha fazla bilgi için JNDI ad alanında bağlantı üreticilerini ve hedefleri yapılandırmabaşlıklı konuya bakın.

V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging 3.0için, IBM MQ Explorerkullanarak JNDI ' yi yönetemezsiniz. JNDI yönetimi, **JMSAdmin**' ın **JMS30Admin**olan Jakarta Messaging 3.0 çeşitlemesi tarafından desteklenir.

Bir uygulama sunucusu, genellikle yönetilen nesneler için kendi havuzunu ve nesneleri oluşturmak ve bakımını yapmak için kendi araçlarını sağlar.

JNDI ad alanından denetlenen nesneleri almak için, aşağıdaki örnekte gösterildiği gibi, bir uygulamanın ilk bağlamı yaratması gerekir:

V 9.3.0 V 9.3.0 JM 3.0

```
import jakarta.jms.*;
import javax.naming.*;
import javax.naming.directory.*;
.
.
.
String url = "ldap://server.company.com/o=company_us,c=us";
String icf = "com.sun.jndi.ldap.LdapCtxFactory";
.
java.util.Hashtable environment = new java.util.Hashtable();
environment.put(Context.PROVIDER_URL, url);
environment.put(Context.INITIAL_CONTEXT_FACTORY, icf);
Context ctx = new InitialDirContext(environment);
```

JMS 2.0

```
import javax.jms.*;
import javax.naming.*;
import javax.naming.directory.*;
.
.
.
String url = "ldap://server.company.com/o=company_us,c=us";
String icf = "com.sun.jndi.ldap.LdapCtxFactory";
.
java.util.Hashtable environment = new java.util.Hashtable();
environment.put(Context.PROVIDER_URL, url);
environment.put(Context.INITIAL_CONTEXT_FACTORY, icf);
Context ctx = new InitialDirContext(environment);
```

Bu kodda, Dizgi değişkenleri `url` ve `icf` aşağıdaki anlamları taşır:

url
Dizin hizmetinin birörnek kaynak konum belirleyicisi (URL). URL aşağıdaki biçimlerden birine sahip olabilir:

- `ldap://`*`hostname`*`/`*`contextName`* , LDAP sunucusuna dayalı bir dizin hizmeti için
- `file:/`*`directoryPath`* , yerel dosya sistemine dayalı bir dizin hizmeti için

icf
Aşağıdaki değerlerden biri olabilen ilk bağlam üreticisinin sınıf adı:

- `com.sun.jndi.ldap.LdapCtxFactory`, LDAP sunucusuna dayalı bir dizin hizmeti için
- `com.sun.jndi.fscontext.RefFSContextFactory`, yerel dosya sistemine dayalı bir dizin hizmeti için

JNDI paketinin ve LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) hizmet sağlayıcısının bazı birleşimlerinin LDAP 84 hatasının oluşmasına neden olabileceğini unutmayın. Bu sorunu çözmek için, InitialDirContext () çağrıdan önce aşağıdaki kod satırını ekleyin:

```
environment.put(Context.REFERRAL, "throw");
```

İlk bağlam elde edildikten sonra uygulama, aşağıdaki örnekte gösterildiği gibi, lookup () yöntemini kullanarak JNDI ad alanından denetlenen nesneleri alabilir:

```
ConnectionFactory factory;
Queue queue;
Topic topic;
.
.
.
factory = (ConnectionFactory)ctx.lookup("cn=myCF");
queue = (Queue)ctx.lookup("cn=myQ");
topic = (Topic)ctx.lookup("cn=myT");
```

Bu kod, LDAP tabanlı bir ad alanından aşağıdaki nesneleri alır:

- myCF adıyla bağlı bir ConnectionFactory nesnesi
- myQ adıyla bağlı bir kuyruk nesnesi
- myT adıyla bağlı bir Konu nesnesi

JNDI kullanımıyla ilgili daha fazla bilgi için Oracle Corporation tarafından sağlanan JNDI belgelerine bakın.

İlgili görevler
IBM MQ Explorer 'ı kullanarak JMS 2.0 nesnelerini yapılandırma
Yönetim araçlarını kullanarak JMS ve Jakarta Messaging nesnelerini yapılandırma
WebSphere Application Server ' da JMS 2.0 kaynaklarının yapılandırılması

IBM JMS uzantılarının kullanılması
IBM MQ classes for JMS (JMS 2.0) ve IBM MQ classes for Jakarta Messaging (Jakarta Messaging 3.0) her biri, IBM JMS uzantıları adı verilen JMS API ile işlevsel olarak özdeş bir uzantı kümesi içerir. Bir uygulama, çalıştırma zamanında bağlantı üreticilerini ve hedefleri dinamik olarak yaratmak ve IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnelerinin özelliklerini ayarlamak için bu uzantıları kullanabilir. Uzantılar herhangi bir ileti sistemi sağlayıcısıyla kullanılabilir.

IBM JMS uzantıları, aşağıdaki paketlerdeki bir arabirim ve sınıf kümeleridir:

- com.ibm.msg.client.jms
- com.ibm.msg.client.services

Jakarta Messaging 3.0için, bu paketler `com.ibm.jakarta.client.jar`içinde yer alan paketlerdir.

JMS 2.0 JMS 2.0için bu paketler `com.ibm.mqjms.jar` ya da `com.ibm.mq.allclient.jar`içinde yer alan paketlerdir.

Bu uzantılar aşağıdaki işlevi sağlar:

- Java Naming and Directory Interface (JNDI) ad alanından yönetilen nesneler olarak almak yerine, çalıştırma zamanında dinamik olarak bağlantı üreticileri ve hedefleri oluşturmak için fabrikaya dayalı bir mekanizma
- IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnelerinin özelliklerini ayarlamak için bir yöntem kümesi
- Bir sorunla ilgili ayrıntılı bilgi almak için yöntemler içeren kural dışı durum sınıfları kümesi
- İzlemeyi denetlemek için bir yöntem kümesi
- IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging ile ilgili sürüm bilgilerini almak için bir yöntem kümesi

IBM JMS uzantıları, çalıştırma zamanında dinamik olarak bağlantı üreticileri ve hedefleri oluşturmak ve özelliklerini ayarlamak ve almak için IBM MQ JMS uzantılarına alternatif bir arabirim kümesi sağlar. Ancak, IBM MQ JMS uzantıları IBM MQ ileti sistemi sağlayıcısına özgüdür, ancak IBM JMS uzantıları IBM MQ ' e özgü değildir ve IBM MQ sınıflarında JMS mimarisibaşlıklı konuda açıklanan katmanlı mimari içinde herhangi bir ileti sistemi sağlayıcısıyla birlikte kullanılabilir.

com.ibm.msg.client.wmq.WMQConstants (JMS 2.0) ya da com.ibm.msg.jakarta.client.wmq.WMQConstants (Jakarta Messaging 3.0) arabirimi, bir uygulamanın IBM JMS uzantılarını kullanarak IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnelerinin özelliklerini ayarlarken kullanabileceği değişmezlerin tanımlamalarını içerir. Arabirim, IBM MQ ileti alışverişi sağlayıcısına ve herhangi bir ileti alışverişi sağlayıcısından bağımsız JMS değişmezlerine ilişkin değişmezleri içerir.

Aşağıdaki kod örnekleri, Java sınıfında aşağıdaki içe aktarma deyimlerini içerdiğini varsayar:

V 9.3.0 V 9.3.0 JM 3.0
```
import com.ibm.msg.jakarta.client.jms.*;
import com.ibm.msg.jakarta.client.services.*;
import com.ibm.msg.jakarta.client.wmq.WMQConstants;
```

JMS 2.0
```
import com.ibm.msg.client.jms.*;
import com.ibm.msg.client.services.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Bağlantı üreticileri ve hedefleri yaratılması

Bir uygulamanın IBM JMS uzantılarını kullanarak bağlantı üreticileri ve hedefler oluşturabilmesi için önce bir JmsFactoryFactory nesnesi oluşturması gerekir. Bir JmsFactoryFactory nesnesi yaratmak için, uygulama aşağıdaki örnekte gösterildiği gibi, JmsFactoryFactory sınıfının getInstance() yöntemini çağırır:

V 9.3.0 V 9.3.0 JM 3.0

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.JAKARTA_WMQ_PROVIDER);
```

JMS 2.0

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
```

getInstance() çağrısının değiştirgesi, IBM MQ ileti alışverişi sağlayıcısını seçilen ileti alışverişi sağlayıcısı olarak tanımlayan bir değişmez. Daha sonra uygulama, bağlantı üreticileri ve hedefler yaratmak için JmsFactoryFactory nesnesini kullanabilir.

Bir bağlantı üreticisi yaratmak için, uygulama aşağıdaki örnekte gösterildiği gibi, JmsFactoryFactory nesnesinin createConnectionFactory () yöntemini çağırır:

```
JmsConnectionFactory factory = ff.createConnectionFactory();
```

Bu deyim, tüm özellikleri için varsayılan değerleri olan bir JmsConnectionFactory nesnesi yaratır; bu, uygulamanın bağ tanımlama kipinde varsayılan kuyruk yöneticisine bağlanacağı anlamına gelir. Bir uygulamanın istemci kipinde bağlanmasını ya da varsayılan kuyruk yöneticisinden başka bir kuyruk yöneticisine bağlanmasını istiyorsanız, uygulama bağlantıyı yaratmadan önce JmsConnectionFactory nesnesinin uygun özelliklerini ayarlamalıdır. Bunun nasıl yapılacağını öğrenmek için bkz. “IBM MQ classes for JMS nesnelerinin özelliklerini ayarlama” sayfa 202.

JmsFactoryFactory sınıfı, aşağıdaki tiplerde bağlantı üreticileri yaratmaya ilişkin yöntemler de içerir:

- JmsQueueConnectionFactory
- JmsTopicConnectionFactory
- JmsXAConnectionÜreticisi
- JmsXAQueueConnectionFactory
- JmsXATopicConnectionFactory

Bir Kuyruk nesnesi yaratmak için uygulama, aşağıdaki örnekte gösterildiği gibi JmsFactoryFactory nesnesinin createQueue() yöntemini çağırır:

```
JmsQueue q1 = ff.createQueue("Q1");
```

Bu deyim, tüm özellikleri için varsayılan değerleri olan bir JmsQueue nesnesi yaratır. Nesne, yerel kuyruk yöneticisine ait olan Q1 adlı bir IBM MQ kuyruğunu temsil eder. Bu kuyruk bir yerel kuyruk, diğer ad kuyruğu ya da uzak kuyruk tanımlaması olabilir.

createQueue() yöntemi, parametre olarak bir kuyruk birörnek kaynak tanıtıcısını (URI) da kabul edebilir. Kuyruk URI 'si, bir IBM MQ kuyruğunun adını ve isteğe bağlı olarak, kuyruğun sahibi olan kuyruk yöneticisinin adını ve JmsQueue nesnesinin bir ya da daha çok özelliğini belirten bir dizedir. Aşağıdaki deyim bir kuyruk URI 'si örneği içeriyor:

```
JmsQueue q2 = ff.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Bu deyim tarafından yaratılan JmsQueue nesnesi, QM2kuyruk yöneticisine ait Q2 adlı bir IBM MQ kuyruğunu gösterir ve bu hedefe gönderilen tüm iletiler kalıcıdır ve 5 önceliğine sahiptir. Kuyruk URI ' leri hakkında daha fazla bilgi için bkz. “Birörnek kaynak tanıtıcıları (URI ' ler)” sayfa 214. JmsQueue nesnesinin özelliklerini ayarlamanın alternatif bir yolu için bkz. “IBM MQ classes for JMS nesnelerinin özelliklerini ayarlama” sayfa 202.

Konu nesnesi yaratmak için, bir uygulama aşağıdaki örnekte gösterildiği gibi JmsFactoryFactory nesnesinin createTopic() yöntemini kullanabilir:

```
JmsTopic t1 = ff.createTopic("Sport/Football/Results");
```

Bu deyim, tüm özellikleri için varsayılan değerlerle bir JmsTopic nesnesi yaratır. Nesne, Spor/Futbol/Sonuçlar adlı bir konuyu temsil eder.

createTopic() yöntemi, bir konu URI 'sini parametre olarak da kabul edebilir. Konu URI 'si, bir konunun adını ve isteğe bağlı olarak JmsTopic nesnesinin bir ya da daha fazla özelliğini belirten bir dizedir. Aşağıdaki deyimler bir konu URI 'si örneği içerir:

```
String s1 = "topic://Sport/Tennis/Results?persistence=1&priority=0";
JmsTopic t2 = ff.createTopic(s1);
```

Bu deyimler tarafından oluşturulan JmsTopic nesnesi Spor/Tennis/Sonuçlar adlı bir konuyu temsil eder ve bu hedefe gönderilen tüm iletiler kalıcı değildir ve 0 önceliğine sahiptir. Konu URI ' leri hakkında daha fazla bilgi için bkz. “Birörnek kaynak tanıtıcıları (URI ' ler)” sayfa 214. JmsTopic nesnesinin özelliklerini ayarlamanın alternatif bir yolu için bkz. “IBM MQ classes for JMS nesnelerinin özelliklerini ayarlama” sayfa 202.

Bir uygulama bir bağlantı üreticisi ya da hedefi yarattıktan sonra, o nesne yalnızca seçilen ileti alışverişi sağlayıcısıyla kullanılabilir.

IBM MQ classes for JMS nesnelerinin özelliklerini ayarlama

IBM MQ classes for JMS nesnelerinin özelliklerini IBM JMS uzantılarını kullanarak ayarlamak için bir uygulama com.ibm.msg.client.JmsPropertyContext arabiriminin yöntemlerini kullanır. Benzer şekilde, bir uygulama IBM JMS uzantılarını kullanarak IBM MQ classes for Jakarta Messaging nesnelerinin özelliklerini ayarlamak için com.ibm.msg.jakarta.client.JmsPropertyContext arabiriminin yöntemlerini kullanır.

Her Java veri tipi için JmsPropertyBağlam arabirimi, o veri tipine sahip bir özelliğin değerini ayarlamak için bir yöntem ve o veri tipine sahip bir özelliğin değerini almak için bir yöntem içerir. Örneğin, bir uygulama tamsayı değeriyle bir özellik ayarlamak için setIntProperty () yöntemini çağırır ve tamsayı değerine sahip bir özellik almak için getIntProperty () yöntemini çağırır.

com.ibm.mq.jms ve com.ibm.mq.jakarta.jms paketlerindeki sınıfların eşgörünümleri, ilgili JmsPropertyBağlam arabirimlerinin yöntemlerini devralır. Bu nedenle, bir uygulama MQConnectionFactory, MQQueue ve MQTopic nesnelerinin özelliklerini ayarlamak için bu yöntemleri kullanabilir.

Bir uygulama bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnesi oluşturduğunda, varsayılan değerleri olan özellikler otomatik olarak ayarlanır. Bir uygulama bir özellik ayarladığında, yeni değer, özelliğin sahip olduğu önceki değerlerin yerine geçer. Bir özellik ayarlandıktan sonra silinemez, ancak değeri değiştirilebilir.

Bir uygulama bir özelliği özellik için geçerli olmayan bir değere ayarlamaya çalışırsa, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging bir JMSException kural dışı durumu verir. Bir uygulama ayarlanmamış bir özelliği alma girişiminde bulunursa, davranış JMS belirtiminde açıklanmıştır. IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , ilkel veri tipleri için NumberFormatkural dışı durumu yayınlıyor ve başvurulan veri tipleri için boş değer döndürüyor.

Bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnesinin önceden tanımlanmış özelliklerine ek olarak, bir uygulama kendi özelliklerini ayarlayabilir. Bu uygulama tanımlı özellikler IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingtarafından yoksayılır.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging nesnelerinin özellikleri hakkında daha fazla bilgi için bkz. IBM MQ classes for JMS nesnelerinin özellikleri.

Aşağıdaki kod, IBM JMS uzantılarını kullanarak özelliklerin nasıl ayarlanacağına ilişkin bir örnektir. Kod, bir bağlantı üreticisinin beş özelliğini ayarlar.

```
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");
```

Bu özelliklerin ayarlanmasının etkisi, uygulamanın QM1.SVRadlı bir MQI kanalı kullanarak istemci kipinde QM1 kuyruk yöneticisine bağlanmasıdır. Kuyruk yöneticisi HOST1anasistem adını taşıyan bir sistemde çalışıyor ve kuyruk yöneticisine ilişkin dinleyici 1415 numaralı kapıda dinliyor. Bu bağlantı ve altındaki oturumlarla ilişkilendirilmiş diğer kuyruk yöneticisi bağlantıları, bunlarla ilişkilendirilmiş "Uygulamam" uygulama adını içerir.

Not: z/OS altyapılarında çalışan kuyruk yöneticileri uygulama adlarının ayarlanmasını desteklemez ve bu nedenle bu ayar yoksayılır.

JmsPropertyBağlam arabirimi, bir uygulamanın özellikleri ayarlamak için kullanabileceği setObjectProperty () yöntemini de içerir. Yöntemin ikinci parametresi, özelliğin değerini saran bir nesnedir. Örneğin, aşağıdaki kod 1415 tamsayısını çevreleyen bir Tamsayı nesnesi yaratır ve daha sonra, bir bağlantı üreticisinin PORT özelliğini 1415 değerine ayarlamak için setObjectProperty () yöntemini çağırır:

```
Integer port = new Integer(1415);
factory.setObjectProperty(WMQConstants.WMQ_PORT, port);
```

Bu nedenle, bu kod aşağıdaki deyimin eşdeğeridir:

```
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
```

Tersi durumda, getObjectProperty () yöntemi bir özelliğin değerini saran bir nesne döndürür.

Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi

Bir uygulama bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnesinin özelliğini ayarlamak ya da almak için JmsPropertyBağlam arabirimi yöntemini kullandığında, özelliğin değeri örtük olarak bir veri tipinden diğerine dönüştürülebilir.

Örneğin, aşağıdaki deyim JmsQueue nesnesinin PRIORITY özelliğini q1:

```
q1.setStringProperty(WMQConstants.WMQ_PRIORITY, "5");
```

PRIORITY özelliğinin bir tamsayı değeri var ve setStringProperty () çağrısı, "5" dizgisini (kaynak değer) örtük olarak 5 numaralı tamsayıya (hedef değer) dönüştürür; bu da PRIORITY özelliğinin değeri olur.

Tersi durumda, aşağıdaki deyim JmsQueue nesnesinin (q1:

```
String s1 = q1.getStringProperty(WMQConstants.WMQ_PRIORITY);
```

PRIORITY özelliğinin değeri olan 5 numaralı tamsayı (kaynak değer), getStringProperty () çağrısıyla örtük olarak "5" dizgisine (hedef değer) dönüştürülür.

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging tarafından desteklenen dönüştürmeler Çizelge 34 sayfa 203içinde gösterilmektedir.

Çizelge 34. Bir veri tipinden diğerine desteklenen dönüştürmeler

Kaynak veri tipi	**Desteklenen hedef veri tipleri**
boole	Dizgi

Çizelge 34. Bir veri tipinden diğerine desteklenen dönüştürmeler (devamı var)	
Kaynak veri tipi	**Desteklenen hedef veri tipleri**
Byte	int, long, short, String
DAMGA	Dizgi
çift	Dizgi
kayan nokta	çift, Dizgi
int	uzun, Dizgi
uzun	Dizgi
kısa	int, long, String
Dizgi	boole, byte, double, float, int, long, short

Desteklenen dönüştürmeleri düzenleyen genel kurallar şunlardır:

- Dönüştürme sırasında veri kaybına uğramadan, sayısal değerler bir veri tipinden diğerine dönüştürülebilir. Örneğin, `int` veri tipine sahip bir değer, `long`veri tipine sahip bir değere dönüştürülebilir, ancak `short`veri tipine sahip bir değere dönüştürülemez.
- Herhangi bir veri tipindeki bir değer dizgiye dönüştürülebilir.
- Bir dizgi başka bir veri tipine (`char` dışında) dönüştürülebilir dizginin dönüştürme için doğru biçimde olması koşuluyla. Bir uygulama doğru biçimde olmayan bir dizgiyi dönüştürmeyi denerse, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging bir NumberFormatKural Dışı Durum kural dışı durumu yayınlar.
- Bir uygulama desteklenmeyen bir dönüştürme girişiminde bulunursa, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging bir MessageFormatkural dışı durumu yayınlar.

Bir değeri bir veri tipinden diğerine dönüştürmeye ilişkin belirli kurallar şunlardır:

- Bir Boole değeri dizgiye dönüştürülürken, `true` değeri "true"dizgisine dönüştürülür ve `false` değeri "false"dizgisine dönüştürülür.
- Bir dizgi Boole değerine dönüştürülürken, "true" dizgisi (büyük ve küçük harfe duyarlı değil) `true`değerine ve "false" dizgisi (büyük ve küçük harfe duyarlı değil) `false`değerine dönüştürülür. Diğer dizgiler `false`olarak dönüştürülür.
- Bir dizgi `byte`, `int`, `long`ya da `short`veri tipli bir değere dönüştürülürken, dizgi aşağıdaki biçimde olmalıdır:

  [*boşluklar*] [*işaret*] *rakamlar*

  Dizginin bileşenlerinin anlamları aşağıdaki gibidir:

  boşluklar
  İsteğe bağlı baştaki boş karakterler.

  İşaret
  İsteğe bağlı artı işareti (+) ya da eksi işareti (-).

  Rakamlar
  Bitişik basamak sırası (0-9). En az bir basamak var olmalıdır.

  Sayı dizgisinden sonra, dizilim basamak olmayan başka karakterler içerebilir, ancak bu karakterlerin ilkine ulaşılır ulaşılmaz dönüştürme durur. Dizginin bir ondalık tamsayıyı temsil ettiği varsayılır.

  Dizgi doğru biçimde değilse, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging bir NumberFormatkural dışı durumu yayınlar.
- Bir dizgi `double` ya da `float`veri tipindeki bir değere dönüştürülürken, dizgi aşağıdaki biçimde olmalıdır:

  [*boşluklar*] [*işaret*] *rakamlar* [*e_char* [*e_sign*] *e_digits*]

Dizginin bileşenlerinin anlamları aşağıdaki gibidir:

boşluklar
İsteğe bağlı baştaki boş karakterler.

İşaret
İsteğe bağlı artı işareti (+) ya da eksi işareti (-).

Rakamlar
Bitişik basamak sırası (0-9). En az bir basamak var olmalıdır.

e_char
E ya da *e*olan üstel bir karakter.

e_işareti
Üstel için isteğe bağlı artı işareti (+) ya da eksi işareti (-).

e_basamaklar
Üs için bitişik basamak sırası (0-9). Dizgi bir üstel karakter içeriyorsa, en az bir basamak bulunmalıdır.

Basamak sırasından ya da bir üstel karakteri gösteren isteğe bağlı karakterlerden sonra, dizgi rakam olmayan diğer karakterleri içerebilir, ancak bu karakterlerin ilkine ulaşıldığında dönüştürme durur. Dizginin, 10 üslü bir ondalık kayan nokta sayısını temsil ettiği varsayılır.

Dizgi doğru biçimde değilse, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging bir NumberFormatkural dışı durumu yayınlar.

- Sayısal bir değer dönüştürülürken (veri tipi `byte` olan bir değer de içinde olmak üzere) bir dizgiye, değer, değerin ASCII karakterini içeren dizgiye değil, ondalık sayı olarak dizgi gösterimine dönüştürülür. Örneğin, 65 tamsayısı "A"dizgisine değil, "65"dizgisine dönüştürülür.

Tek bir aramada birden çok özellik ayarlama

JmsPropertyBağlam arabirimi, bir uygulamanın tek bir çağrıda birden çok özelliği ayarlamak için kullanabileceği setBatchProperties () yöntemini de içerir. Yöntemin parametresi, bir özellik ad-değer çiftleri kümesini saran bir Eşleme nesnesidir.

Örneğin, aşağıdaki kod, “IBM MQ classes for JMS nesnelerinin özelliklerini ayarlama” sayfa 202içinde gösterildiği gibi bir bağlantı üreticisinin aynı beş özelliğini ayarlamak için setBatchProperties () yöntemini kullanır. Kod, Eşlem arabirimini uygulayan HashMap sınıfının bir eşgörünümünü yaratır.

```
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);
```

Map.put() yönteminin ikinci parametresinin bir nesne olması gerektiğini unutmayın. Bu nedenle, ilkel veri tipine sahip bir özellik değeri, örnekte gösterildiği gibi, bir nesne içinde ya da bir dizgiyle gösterilmelidir.

setBatchProperties () yöntemi her bir özelliği doğrular. Örneğin, değeri geçerli olmadığı için setBatchProperties () yöntemi bir özelliği ayarlayamazsa, belirtilen özelliklerin hiçbiri ayarlanmaz.

Özellik adları ve değerleri

Bir uygulama IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnelerinin özelliklerini ayarlamak ve almak için uygun JmsPropertyBağlam arabiriminin yöntemlerini kullanıyorsa, özelliklerin adlarını ve değerlerini aşağıdaki yollardan biriyle belirtebilir. Eşlik eden örneklerin her biri, kuyruğa gönderilen bir iletinin gönderme () çağrısında belirtilen önceliğe sahip olması için JmsQueue nesnesinin q1 PRIORITY özelliğinin nasıl ayarlanacağını gösterir.

com.ibm.msg.client.wmq.WMQConstants arabiriminde değişmez olarak tanımlanan özellik adlarının ve değerlerin kullanılması

Aşağıdaki deyim, özelliklerin adlarının ve değerlerinin bu şekilde nasıl belirtileceğini gösteren bir örnektir:

```
q1.setIntProperty(WMQConstants.WMQ_PRIORITY, WMQConstants.WMQ_PRI_APP);
```

Kuyrukta ve konu birörnek kaynak tanıtıcılarında (URI ' ler) kullanılabilecek özellik adlarını ve değerlerini kullanma

Aşağıdaki deyim, özelliklerin adlarının ve değerlerinin bu şekilde nasıl belirtileceğini gösteren bir örnektir:

```
q1.setIntProperty("priority", -2);
```

Bu şekilde yalnızca hedeflerin özelliklerinin adları ve değerleri belirtilebilir.

IBM MQ JMS yönetim aracı tarafından tanınan özellik adlarını ve değerlerini kullanma

Aşağıdaki deyim, özelliklerin adlarının ve değerlerinin bu şekilde nasıl belirtileceğini gösteren bir örnektir:

```
q1.setStringProperty("PRIORITY", "APP");
```

Aşağıdaki deyimde gösterildiği gibi, özellik adının kısa biçimi de kabul edilebilir:

```
q1.setStringProperty("PRI", "APP");
```

Bir uygulama bir özellik aldığında, döndürülen değer, uygulamanın özelliğin adını nasıl belirttiğine bağlıdır. Örneğin, bir uygulama özellik adı olarak WMQConstants.WMQ_PRIORITY değişmezini belirtiyorsa, döndürülen değer -2 tamsayısıdır:

```
int n1 = getIntProperty(WMQConstants.WMQ_PRIORITY);
```

Uygulama özellik adı olarak "priority" dizgisini belirtiyorsa aynı değer döndürülür:

```
int n2 = getIntProperty("priority");
```

Ancak, uygulama özellik adı olarak "PRIORITY" ya da "PRI" dizgisini belirtiyorsa, döndürülen değer "APP" dizgisidir:

```
String s1 = getStringProperty("PRI");
```

İç olarak, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging özellik adlarını ve değerlerini, eşleşen WMQConstants arabiriminde tanımlı hazır bilgi değerleri olarak saklar. Bu, özellik adları ve değerleri için tanımlanan kurallı biçimdir. Genel bir kural olarak, bir uygulama özellik adlarını ve değerlerini belirtmenin diğer iki yönteminden birini kullanarak özellikleri ayarlarsa, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging adları ve değerleri belirtilen giriş biçiminden kurallı biçime dönüştürmelidir. Benzer şekilde, bir uygulama özellik adlarını ve değerlerini belirtmenin diğer iki yönteminden birini kullanarak özellikleri alırsa, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging adları belirtilen giriş biçiminden kurallı biçime dönüştürmeli ve değerleri kurallı biçimden istenen çıkış biçimine dönüştürmelidir. Bu dönüştürmeleri gerçekleştirmek zorunda kalmanın başarımla ilgili etkileri olabilir.

İzleme dosyalarında ya da IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging günlüğünde kural dışı durumlar tarafından döndürülen özellik adları ve değerleri her zaman kurallı biçimdedir.

Harita arabiriminin kullanılması

JmsPropertyBağlam arabirimi java.util.Map arabirimini genişletir. Bu nedenle bir uygulama, bir IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnesinin özelliklerine erişmek için Harita arabiriminin yöntemlerini kullanabilir.

Örneğin, aşağıdaki kod, bir bağlantı üreticisinin tüm özelliklerinin adlarını ve değerlerini yazdırır. Kod, özelliklerin adlarını ve değerlerini almak için yalnızca Harita arabiriminin yöntemlerini kullanır.

```
// 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));
}
```

Eşlem arabiriminin yöntemlerinin kullanılması, herhangi bir özellik doğrulamayı ya da dönüştürmeyi atlamaz.

IBM MQ JMS uzantılarının kullanılması
IBM MQ classes for JMS , IBM MQ JMS uzantıları adı verilen JMS API için bir uzantı kümesi içerir. Bir uygulama, çalıştırma zamanında dinamik olarak bağlantı üreticileri ve hedefler oluşturmak ve bağlantı üreticileri ve hedeflerinin özelliklerini ayarlamak için bu uzantıları kullanabilir.

IBM MQ classes for JMS , com.ibm.jms ve com.ibm.mq.jmspaketlerinde bir sınıf kümesi içerir. Bu sınıflar JMS arabirimlerini uygular ve IBM MQ JMS uzantılarını içerir. Aşağıdaki kod örnekleri, bu paketlerin aşağıdaki deyimler tarafından içe aktarıldığını varsayar:

```
import com.ibm.jms.*;
import com.ibm.mq.jms.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

Bir uygulama, aşağıdaki işlevleri gerçekleştirmek için IBM MQ JMS uzantılarını kullanabilir:

- Java Naming and Directory Interface (JNDI) ad alanından yönetilen nesneler olarak almak yerine, çalıştırma zamanında bağlantı üreticilerini ve hedefleri dinamik olarak oluşturun
- Bağlantı üreticileri ve hedeflerinin özelliklerini ayarlayın

Bağlantı üreticileri yaratılıyor

Bir bağlantı üreticisi yaratmak için, aşağıdaki örnekte gösterildiği gibi, bir uygulama MQConnectionFactory oluşturucusunu kullanabilir:

```
MQConnectionFactory factory = new MQConnectionFactory();
```

Bu deyim, tüm özellikleri için varsayılan değerleri olan bir MQConnectionFactory nesnesi yaratır; bu, uygulamanın bağ tanımlama kipinde varsayılan kuyruk yöneticisine bağlanacağı anlamına gelir. Bir uygulamanın istemci kipinde bağlanmasını ya da varsayılan kuyruk yöneticisinden başka bir kuyruk yöneticisine bağlanmasını istiyorsanız, uygulama bağlantıyı yaratmadan önce MQConnectionFactory nesnesinin uygun özelliklerini ayarlamalıdır. Bunun nasıl yapılacağını öğrenmek için bkz. “Bağlantı üreticileri özelliklerinin ayarlanması” sayfa 208.

Bir uygulama, benzer bir şekilde aşağıdaki tiplerde bağlantı üreticileri yaratabilir:

- MQQueueConnectionÜreticisi
- MQTopicConnectionÜreticisi
- MQXAConnectionFactory
- MQXAQueueConnectionÜreticisi

- MQXATopicConnectionÜreticisi

Bağlantı üreticileri özelliklerinin ayarlanması

Bir uygulama, bağlantı üreticisinin uygun yöntemlerini çağırarak bağlantı üreticisinin özelliklerini ayarlayabilir. Bağlantı üreticisi, yürütme sırasında devingen olarak yaratılan bir nesne ya da denetlenen bir nesne olabilir.

Örneğin, aşağıdaki kodu göz önünde bulundurun:

```
MQConnectionFactory factory = new MQConnectionFactory();
.
factory.setTransportType(WMQConstants.WMQ_CM_CLIENT);
factory.setQueueManager("QM1");
factory.setHostName("HOST1");
factory.setPort(1415);
factory.setChannel("QM1.SVR");
```

Bu kod bir MQConnectionFactory nesnesi yaratır ve nesnenin beş özelliğini ayarlar. Bu özelliklerin ayarlanmasının etkisi, uygulamanın QM1.SVRadlı bir MQI kanalını kullanarak istemci kipinde QM1 kuyruk yöneticisine bağlanmasıdır. Kuyruk yöneticisi HOST1anasistem adını taşıyan bir sistemde çalışıyor ve kuyruk yöneticisine ilişkin dinleyici 1415 numaralı kapıda dinliyor.

Bir aracıya gerçek zamanlı bağlantı kullanan bir uygulama yalnızca ileti alışverişi yayınlama/abone olma biçemini kullanabilir. İleti sisteminin noktadan noktaya iletişim biçemini kullanamaz.

Yalnızca bir bağlantı üreticisinin bazı özellik birleşimleri geçerlidir. Hangi birleşimlerin geçerli olduğu hakkında bilgi için bkz. IBM MQ classes for JMS nesnelerinin özellikleri arasındaki bağımlılıklar.

Bir bağlantı üreticisinin özellikleri ve özelliklerini ayarlamak için kullanılan yöntemler hakkında daha fazla bilgi için IBM MQ classes for JMS nesnelerinin özellikleribaşlıklı konuya bakın.

Hedef oluşturma

Bir kuyruk nesnesi yaratmak için, uygulama aşağıdaki örnekte gösterildiği gibi MQQueue oluşturucusunu kullanabilir:

```
MQQueue q1 = new MQQueue("Q1");
```

Bu deyim, tüm özellikleri için varsayılan değerleri olan bir MQQueue nesnesi yaratır. Nesne, yerel kuyruk yöneticisine ait olan Q1 adlı bir IBM MQ kuyruğunu temsil eder. Bu kuyruk bir yerel kuyruk, diğer ad kuyruğu ya da uzak kuyruk tanımlaması olabilir.

Aşağıdaki örnekte gösterildiği gibi, MQQueue oluşturucusunun diğer bir biçiminin iki değiştirgesi vardır:

```
MQQueue q2 = new MQQueue("QM2", "Q2");
```

Bu deyim tarafından yaratılan MQQueue nesnesi, QM2kuyruk yöneticisine ait Q2 adlı bir IBM MQ kuyruğunu gösterir. Bu şekilde tanıtılan kuyruk yöneticisi yerel kuyruk yöneticisi ya da uzak kuyruk yöneticisi olabilir. Uzak bir kuyruk yöneticisiyse, IBM MQ yapılandırılmalıdır; böylece, uygulama bu hedefe bir ileti gönderdiğinde, WebSphere MQ iletiyi yerel kuyruk yöneticisinden uzak kuyruk yöneticisine yöneltir.

MQQueue oluşturucusu, bir kuyruk birörnek kaynak tanıtıcısını (URI) tek bir değiştirge olarak da kabul edebilir. Kuyruk URI 'si, bir IBM MQ kuyruğunun adını ve isteğe bağlı olarak, kuyruğun sahibi olan kuyruk yöneticisinin adını ve MQQueue nesnesinin bir ya da daha çok özelliğini belirten bir dizedir. Aşağıdaki deyim bir kuyruk URI 'si örneği içeriyor:

```
MQQueue q3 = new MQQueue("queue://QM3/Q3?persistence=2&priority=5");
```

Bu deyim tarafından yaratılan MQQueue nesnesi, QM3kuyruk yöneticisine ait Q3 adlı bir IBM MQ kuyruğunu temsil eder ve bu hedefe gönderilen tüm iletiler kalıcı ve 5 önceliğine sahiptir. Kuyruk URI

' leri hakkında daha fazla bilgi için bkz. “Birörnek kaynak tanıtıcıları (URI ' ler)” sayfa 214. Bir MQQueue nesnesinin özelliklerini ayarlamanın diğer bir yolu için bkz. “Hedeflerin özelliklerini ayarlama” sayfa 209.

Konu nesnesi yaratmak için, aşağıdaki örnekte gösterildiği gibi, bir uygulama MQTopic oluşturucusunu kullanabilir:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
```

Bu deyim, tüm özellikleri için varsayılan değerleri olan bir MQTopic nesnesi yaratır. Nesne, Spor/Futbol/Sonuçlar adlı bir konuyu temsil eder.

MQTopic oluşturucusu bir konu URI 'sini parametre olarak da kabul edebilir. Konu URI 'si, bir konunun adını ve isteğe bağlı olarak MQTopic nesnesinin bir ya da daha fazla özelliğini belirten bir dizedir. Aşağıdaki deyim bir konu URI 'si örneğini içerir:

```
MQTopic t2 = new MQTopic("topic://Sport/Tennis/Results?persistence=1&priority=0");
```

Bu deyim tarafından yaratılan MQTopic nesnesi, Spor/Tennis/Sonuçlar adlı bir konuyu temsil eder ve bu hedefe gönderilen tüm iletiler kalıcı değildir ve 0 önceliğine sahiptir. Konu URI ' leri hakkında daha fazla bilgi için bkz. “Birörnek kaynak tanıtıcıları (URI ' ler)” sayfa 214. Bir MQTopic nesnesinin özelliklerini ayarlamanın diğer bir yolu için bkz. “Hedeflerin özelliklerini ayarlama” sayfa 209.

Hedeflerin özelliklerini ayarlama

Bir uygulama, hedefin uygun yöntemlerini çağırarak bir hedefin özelliklerini ayarlayabilir. Hedef, yönetilen bir nesne ya da yürütme sırasında dinamik olarak yaratılan bir nesne olabilir.

Örneğin, aşağıdaki kodu göz önünde bulundurun:

```
MQQueue q1 = new MQQueue("Q1");
.
q1.setPersistence(WMQConstants.WMQ_PER_PER);
q1.setPriority(5);
```

Bu kod bir MQQueue nesnesi yaratır ve nesnenin iki özelliğini ayarlar. Bu özelliklerin ayarlanmasının etkisi, hedefe gönderilen tüm iletilerin kalıcı olması ve 5 önceliğine sahip olması şeklindedir.

Bir uygulama, aşağıdaki örnekte gösterildiği gibi, MQTopic nesnesinin özelliklerini benzer bir şekilde ayarlayabilir:

```
MQTopic t1 = new MQTopic("Sport/Football/Results");
.
t1.setPersistence(WMQConstants.WMQ_PER_NON);
t1.setPriority(0);
```

Bu kod bir MQTopic nesnesi yaratır ve nesnenin iki özelliğini ayarlar. Bu özelliklerin ayarlanmasının etkisi, hedefe gönderilen tüm iletilerin kalıcı olmamasına ve 0 önceliğine sahip olmamasına neden olur.

Bir hedefin özellikleri ve özelliklerini ayarlamak için kullanılan yöntemler hakkında daha fazla bilgi için IBM MQ classes for JMS nesnelerinin özellikleribaşlıklı konuya bakın.

Linux AIX *JMS uygulamasından IBM MQ ' ya bağlanma*

Bir bağlantı oluşturmak için JMS uygulaması, **Connection** nesnesi oluşturmak üzere bir **ConnectionFactory** nesnesi kullanır ve bağlantıyı başlatır.

JMS 2.0 ve üstü için, uygulamalar genellikle bir **ConnectionFactory** nesnesini ve `createContext()` yöntemini kullanarak ileti alışverişi sağlayıcısına bağlanır.

JMS ' nin önceki sürümlerinde, önce bir **Connection** nesnesi yaratmak için `createConnection` kullanmanız, daha sonra ileti alışverişi işlemlerini gerçekleştirebilecek bir **Session** nesnesi yaratmak için `getSession()` bağlantı çağrısını başlatmanız gerekiyordu.

JMSContext nesnesi, hem **Connection** hem de **Session** nesnelerini etkili bir şekilde kaplar. Geleneksel yaklaşımı kullanmak ve bağlantı ve oturum nesnelerini doğrudan oluşturmak istiyorsanız bkz. "JMS uygulamasında bağlantı oluşturma" sayfa 210 ve "JMS uygulamasında oturum oluşturma" sayfa 211.

Bir **JMSContext** nesnesi oluşturmak için uygulama, aşağıdaki örnekte gösterildiği gibi **ConnectionFactory** nesnesinin `createContext()` yöntemini kullanır:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createContext();
```

Bir JMS bağlantısı yaratıldığında, IBM MQ classes for JMS bir bağlantı tanıtıcısı (Hconn) yaratır ve kuyruk yöneticisiyle bir etkileşim başlatır.

Not: Uygulama işlemi tanıtıcısının, kuyruk yöneticisine geçirilecek varsayılan kullanıcı kimliği olarak kullanıldığını unutmayın. Uygulama istemci iletim kipinde çalışıyorsa, sunucuda ilgili yetkileriyle birlikte bu süreç tanıtıcısının var olması gerekir. Farklı bir kimlik kullanılmasını istiyorsanız, `createConnection(username,password)` yöntemini kullanın.

V 9.3.5 Bu düzenek, bir kimlik doğrulama simgesi sağlamak için de kullanılabilir; bkz. Seçilen belirteç düzenleyicinizden kimlik doğrulama simgesi alınması.

JMS 1.0 *JMS uygulamasında bağlantı oluşturma*

Bir bağlantı oluşturmak için JMS 1.0 içinde, JMS uygulaması bir ConnectionFactory nesnesini kullanarak bir Bağlantı nesnesi yaratır ve bağlantıyı başlatır.

Bir Connection nesnesi yaratmak için, bir uygulama aşağıdaki örnekte gösterildiği gibi bir ConnectionFactory nesnesinin createConnection() yöntemini kullanır:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createConnection();
```

Bir JMS bağlantısı yaratıldığında, IBM MQ classes for JMS bir bağlantı tanıtıcısı (Hconn) yaratır ve kuyruk yöneticisiyle bir etkileşim başlatır.

QueueConnectionFactory arabirimi ve TopicConnectionFactory arabirimi, ConnectionFactory arabiriminden createConnection() yöntemini devralır. Bu nedenle, aşağıdaki örnekte gösterildiği gibi etki alanına özgü bir nesne yaratmak için createConnection() yöntemini kullanabilirsiniz:

```
QueueConnectionFactory qcf;
Connection connection;
.
.
.
connection = qcf.createConnection();
```

Bu kod parçası bir QueueConnection nesnesi yaratır. Bir uygulama artık bu nesne üzerinde etki alanından bağımsız bir işlem ya da yalnızca noktadan noktaya iletişim etki alanı için geçerli bir işlem gerçekleştirebilir. Ancak, uygulama yalnızca yayınlama/abone olma etki alanı için geçerli olan bir işlemi gerçekleştirme girişiminde bulunursa, aşağıdaki iletiyle birlikte bir IllegalStatekural dışı durum kural dışı durumu yayınlanır:

```
JMSMQ1112: Operation for a domain specific object was not valid.
           Operation createProducer() is not valid for type com.ibm.mq.jms.MQTopic
```

Bunun nedeni, bağlantının etki alanına özgü bir bağlantı üreticisinden yaratılmasıdır.

Not: Uygulama işlemi tanıtıcısının, kuyruk yöneticisine geçirilecek varsayılan kullanıcı kimliği olarak kullanıldığını unutmayın. Uygulama istemci iletim kipinde çalışıyorsa, sunucuda ilgili yetkileriyle birlikte bu süreç tanıtıcısının var olması gerekir. Farklı bir kimlik kullanılmasını istiyorsanız, createConnection(kullanıcı adı, parola) yöntemini kullanın.

JMS belirtimi, `stopped` durumunda bir bağlantının oluşturulduğunu belirtir. Bir bağlantı başlatılıncaya kadar, bağlantıyla ilişkilendirilmiş bir ileti tüketicisi ileti alamaz. Bir bağlantıyı başlatmak için uygulama, aşağıdaki örnekte gösterildiği gibi bir Connection nesnesinin start () yöntemini kullanır:

```
connection.start();
```

V 9.3.5 Bu düzenek, bir kimlik doğrulama simgesi sağlamak için de kullanılabilir; bkz. Seçilen belirteç düzenleyicinizden kimlik doğrulama simgesi alınması.

JMS 1.0 *JMS uygulamasında oturum oluşturma*

Bir oturum JMS 1.0 içindeyaratmak için JMS uygulaması, Connection nesnesinin createSession() yöntemini kullanır.

createSession() yönteminin iki parametresi vardır:

1. Oturumun hareket edilip edilmediğini belirten bir parametre.
2. Oturum için alındı bildirimi kipini belirten bir parametre

Örneğin, aşağıdaki kod işlem yapılmamış bir oturum yaratır ve alındı bildirimi kipi AUTO_ALINDI bildirimi olan bir oturum yaratır:

```
Session session;
.
boolean transacted = false;
session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);
```

Bir JMS oturumu yaratıldığında, IBM MQ classes for JMS bir bağlantı tanıtıcısı (Hconn) yaratır ve kuyruk yöneticisiyle bir etkileşim başlatır.

Bir Oturum nesnesi ve bundan yaratılan herhangi bir MessageProducer ya da MessageConsumer nesnesi, çok iş parçacıklı bir uygulamanın farklı iş parçacıkları tarafından eşzamanlı olarak kullanılamaz. Bu nesnelerin eşzamanlı olarak kullanılmamasını sağlamanın en basit yolu, her iş parçacığı için ayrı bir Oturum nesnesi yaratmaktır.

V 9.3.5 Bu düzenek, bir kimlik doğrulama simgesi sağlamak için de kullanılabilir; bkz. Seçilen belirteç düzenleyicinizden kimlik doğrulama simgesi alınması.

JMS uygulamalarında işlemli oturumlar

JMS uygulamaları önce işlemli bir oturum yaratarak yerel hareketleri çalıştırabilir. Bir uygulama bir hareketi kesinleştirebilir ya da geri alabilir.

JMS uygulamaları yerel hareketleri çalıştırabilir. Yerel hareket, yalnızca uygulamanın bağlı olduğu kuyruk yöneticisinin kaynaklarında yapılan değişiklikleri içeren bir harekettir. Yerel hareketleri çalıştırmak için, bir uygulamanın öncelikle bir Connection nesnesinin createSession() yöntemini çağırarak, oturumun hareket halinde olduğunu parametre olarak belirtmesi gerekir. Daha sonra, oturum içinde gönderilen ve alınan tüm iletiler bir işlem dizisi içinde gruplanır. Hareket, uygulama, hareket başladığından bu yana gönderdiği ve aldığı iletileri kesinleştirdiğinde ya da geri aldığında sona erer.

Bir hareketi kesinleştirmek için, uygulama Oturum nesnesinin commit () yöntemini çağırır. Bir işlem kesinleştirildiğinde, hareket içinde gönderilen tüm iletiler diğer uygulamalara teslim edilmek üzere kullanılabilir olur ve hareket içinde alınan tüm iletiler, ileti sistemi sunucusunun bunları uygulamaya yeniden teslim etmeye çalışmaması için onaylanır. Noktadan noktaya iletişim etki alanında, ileti sistemi sunucusu alınan iletileri kuyruklarından da kaldırır.

Bir hareketi geriye işlemek için, bir uygulama Oturum nesnesinin rollback () yöntemini çağırır. Bir hareket geriye işlendiğinde, hareket içinde gönderilen tüm iletiler ileti sistemi sunucusu tarafından atılır ve hareket içinde alınan tüm iletiler yeniden teslim için kullanılabilir duruma gelir. Noktadan noktaya iletişim etki alanında, alınan iletiler kuyruklarına geri konur ve diğer uygulamalar tarafından yeniden görünür hale gelir.

Yeni bir hareket, bir uygulama hareket içeren bir oturum yarattığında ya da commit () ya da rollback () yöntemini çağırdığında otomatik olarak başlar. Bu nedenle, işlem yapılan bir oturumun her zaman etkin bir hareketi vardır.

Bir uygulama işlemli bir oturumu kapattığında, örtük bir geri alma gerçekleşir. Bir uygulama bir bağlantıyı kapattığında, bağlantının işlem yapılan tüm oturumları için örtük bir geri alma oluşur.

Bir uygulama bir bağlantıyı kapatmadan sona erdirilirse, bağlantının tüm hareketli oturumları için örtük bir geriye işleme de gerçekleşir.

Bir işlemin tamamı işlemli bir oturumda yer alır. Bir işlem oturumlara yayılamaz. Başka bir deyişle, bir uygulama iki ya da daha çok hareket içeren oturumda ileti gönderip alamaz ve tüm bu işlemleri tek bir hareket olarak kesinleştiremez ya da geriye işleyemez.

JMS oturumlarının alındı bildirimi kipleri

İşlem yapılmayan her oturumda, uygulama tarafından alınan iletilerin nasıl alındığını belirleyen bir alındı bildirimi kipi vardır. Üç alındı bildirimi kipi vardır ve alındı bildirimi kipinin seçimi, uygulamanın tasarımını etkiler.

Bir oturum işlemden geçirilmezse, uygulama tarafından alınan iletilerin alınma şekli, oturumun alındı bildirimi kipine göre belirlenir. Üç alındı bildirimi kipi aşağıdaki paragraflarda açıklanmıştır:

OTO_ALıNMA

Oturum, uygulama tarafından alınan her iletiyi otomatik olarak onaylar.

İletiler uygulamaya zamanuyumlu olarak teslim edilirse, oturum, bir Alma çağrısının başarıyla tamamlandığı her zaman bir iletinin alındığını onaylar. İletiler zamanuyumsuz olarak teslim edilirse, bir ileti dinleyicinin onMessage() yöntemine yapılan her çağrı başarıyla tamamlandığında oturum bir iletinin alındığını onaylar.

Uygulama başarılı bir şekilde bir ileti alırsa, ancak bir hata alındı bildiriminin oluşmasını engellerse, ileti yeniden teslim edilebilir duruma gelir. Bu nedenle uygulama, yeniden teslim edilen bir iletiyi işleyebilmelidir.

DUPS_OK_ONAY

Oturum, seçtiği zaman uygulama tarafından alınan iletileri onaylar.

Bu alındı bildirimi kipinin kullanılması, oturumun yapması gereken iş miktarını azaltır, ancak ileti onayını önleyen bir hata, yeniden teslim için birden çok iletinin kullanılabilir olmasına neden olabilir. Bu nedenle uygulama, yeniden teslim edilen iletileri işleyebilmelidir.

Sınırlama: AUTO_CEVAP ve DUPS_OK_CEVAP kiplerinde, JMS ileti dinleyicisinde işlenemeyen bir kural dışı durum yayınını desteklemez. Bu, iletilerin başarılı bir şekilde işlenip işlenmediğine bakılmaksızın ileti dinleyici döndüğünde her zaman kabul edileceği anlamına gelir (hatalar önemli değilse ve uygulamanın devam etmesini önlemezse). İleti onayının daha ince denetimine gereksinim duyarsanız, uygulamaya alındı bildirimi işlevlerinin tam denetimini veren CLIENT_ONAY ya da işlem kiplerini kullanın.

ISTEMCI_ONAY

Uygulama, İleti sınıfının Onay yöntemini çağırarak aldığı iletileri onaylar.

Uygulama, her bir iletinin ayrı ayrı alındığını onaylayabilir ya da bir ileti kümesi alabilir ve yalnızca aldığı son ileti için Onay yöntemini çağırabilir. Alındı bildirimi yöntemi çağrıldığında, yöntemin son çağrılmasından bu yana alınan tüm iletiler alınır.

Bu alındı bildirimi kiplerinin herhangi biriyle birlikte, bir uygulama Oturum sınıfının Kurtar yöntemini çağırarak bir oturumda ileti teslimini durdurabilir ve yeniden başlatabilir. Alınan, ancak önceden alınmamış iletiler yeniden teslim edilir. Ancak, bunlar daha önce teslim edildikleri sırayla teslim

edilmeyebilir. Bu arada, daha yüksek öncelikli iletiler gelmiş olabilir ve özgün iletilerin bazılarının süresi dolmuş olabilir. Noktadan noktaya iletişim etki alanında, özgün iletilerin bazıları başka bir uygulama tarafından tüketilmiş olabilir.

Bir uygulama, iletinin JMSRegönderilen üstbilgi alanının içeriğini inceleyerek iletinin yeniden teslim edilip edilmediğini belirleyebilir. Uygulama bunu, Message sınıfının getJMSRedelivered() yöntemini çağırarak yapar.

JMS uygulamasında hedef oluşturma

Bir JMS uygulaması, Java Naming and Directory Interface (JNDI) ad alanından yönetilen nesneler olarak hedefleri almak yerine, yürütme sırasında dinamik olarak hedef yaratmak için bir oturumu kullanabilir. Bir uygulama, IBM MQ kuyruğunu ya da konusunu tanımlamak ve isteğe bağlı olarak bir Kuyruk ya da Konu nesnesine ilişkin bir ya da daha çok özellik belirtmek için tek tip kaynak tanıtıcısını (URI) kullanabilir.

Kuyruk Nesneleri Yaratmak için Oturum Kullanılması

Bir Kuyruk nesnesi yaratmak için, aşağıdaki örnekte gösterildiği gibi, bir uygulama Oturum nesnesinin createQueue() yöntemini kullanabilir:

```
Session session;
.
Queue q1 = session.createQueue("Q1");
```

Bu kod, tüm özellikleri için varsayılan değerleri olan bir Kuyruk nesnesi yaratır. Nesne, yerel kuyruk yöneticisine ait olan Q1 adlı bir IBM MQ kuyruğunu temsil eder. Bu kuyruk bir yerel kuyruk, diğer ad kuyruğu ya da uzak kuyruk tanımlaması olabilir.

createQueue() yöntemi de bir kuyruk URI 'sini parametre olarak kabul eder. Kuyruk URI 'si, bir IBM MQ kuyruğunun adını ve isteğe bağlı olarak, kuyruğun sahibi olan kuyruk yöneticisinin adını ve Kuyruk nesnesinin bir ya da daha çok özelliğini belirten bir dizedir. Aşağıdaki deyim bir kuyruk URI 'si örneği içeriyor:

```
Queue q2 = session.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

Bu deyim tarafından oluşturulan Kuyruk nesnesi, QM2adlı bir kuyruk yöneticisinin sahip olduğu Q2 adlı bir IBM MQ kuyruğunu temsil eder ve bu hedefe gönderilen tüm iletiler kalıcıdır ve 5 önceliğine sahiptir. Bu şekilde tanıtılan kuyruk yöneticisi yerel kuyruk yöneticisi ya da uzak kuyruk yöneticisi olabilir. Uzak bir kuyruk yöneticisiyse, IBM MQ yapılandırılmalıdır; böylece, uygulama bu hedefe bir ileti gönderdiğinde, WebSphere MQ iletiyi yerel kuyruk yöneticisinden QM2kuyruk yöneticisine yönlendirebilir. URI ' ler hakkında daha fazla bilgi için bkz. “Birörnek kaynak tanıtıcıları (URI ' ler)” sayfa 214.

createQueue() yöntemindeki parametrenin sağlayıcıya özgü bilgiler içerdiğini unutmayın. Bu nedenle, JNDI ad alanından yönetilen bir nesne olarak bir Kuyruk nesnesini almak yerine bir Kuyruk nesnesi yaratmak için createQueue() yönteminin kullanılması, uygulamanızı daha az taşınabilir kılabilir.

Bir uygulama, aşağıdaki örnekte gösterildiği gibi, bir Oturum nesnesinin createTemporaryQueue () yöntemini kullanarak TemporaryQueue nesnesi yaratabilir:

```
TemporaryQueue q3 = session.createTemporaryQueue();
```

Geçici bir kuyruk yaratmak için oturum kullanılsa da, geçici bir kuyruğun kapsamı, oturumu yaratmak için kullanılan bağlantıdır. Bağlantının oturumlarından herhangi biri, geçici kuyruk için ileti üreticileri ve ileti tüketicileri yaratabilir. Geçici kuyruk, bağlantı sona erinceye ya da uygulama, hangisi daha önceyse, TemporaryQueue.delete () yöntemini kullanarak geçici kuyruğu belirtik olarak silinceye kadar kalır.

Bir uygulama geçici bir kuyruk yarattığında, IBM MQ classes for JMS uygulamanın bağlı olduğu kuyruk yöneticisinde dinamik bir kuyruk yaratır. Bağlantı üreticisinin TEMPMODEL özelliği, dinamik kuyruğu yaratmak için kullanılan model kuyruğunun adını belirtir ve bağlantı üreticisinin TEMPQPREFIX özelliği, dinamik kuyruğun adını oluşturmak için kullanılan öneki belirtir.

Konu nesneleri yaratmak için oturum kullanılması

Bir Konu nesnesi yaratmak için, aşağıdaki örnekte gösterildiği gibi, bir uygulama Oturum nesnesinin createTopic() yöntemini kullanabilir:

```
Session session;
.
Topic t1 = session.createTopic("Sport/Football/Results");
```

Bu kod, tüm özellikleri için varsayılan değerlerle bir Konu nesnesi oluşturur. Nesne, Spor/Futbol/Sonuçlar adlı bir konuyu temsil eder.

createTopic() yöntemi, parametre olarak bir konu URI 'sini de kabul eder. Konu URI 'si, bir konunun adını ve isteğe bağlı olarak Konu nesnesinin bir ya da daha fazla özelliğini belirten bir dizedir. Aşağıdaki kod bir konu URI 'si örneğini içerir:

```
String uri = "topic://Sport/Tennis/Results?persistence=1&priority=0";
Topic t2 = session.createTopic(uri);
```

Bu kod tarafından oluşturulan Konu nesnesi Spor/Tennis/Sonuçlar adlı bir konuyu temsil eder ve bu hedefe gönderilen tüm iletiler kalıcı değildir ve 0 önceliğine sahiptir. Konu URI ' leri hakkında daha fazla bilgi için bkz. “Birörnek kaynak tanıtıcıları (URI ' ler)” sayfa 214.

createTopic() yöntemindeki parametrenin sağlayıcıya özgü bilgiler içerdiğini unutmayın. Bu nedenle, bir Konu nesnesini JNDI ad alanından yönetilen nesne olarak almak yerine, Konu nesnesi yaratmak için createTopic() yönteminin kullanılması uygulamanızı daha az taşınabilir yapabilir.

Bir uygulama, aşağıdaki örnekte gösterildiği gibi, bir Oturum nesnesinin createTemporaryTopic () yöntemini kullanarak TemporaryTopic nesnesi yaratabilir:

```
TemporaryTopic t3 = session.createTemporaryTopic();
```

Geçici bir konu yaratmak için oturum kullanılsa da, geçici bir konunun kapsamı, oturumu yaratmak için kullanılan bağlantıdır. Bağlantının oturumlarından herhangi biri, geçici konu için ileti üreticileri ve ileti tüketicileri yaratabilir. Bağlantı sona erinceye ya da uygulama, TemporaryTopic.delete () yöntemini kullanarak geçici konuyu belirtik olarak silinceye kadar geçici konu kalır.

Bir uygulama geçici bir konu oluşturduğunda IBM MQ classes for JMS , TEMP/ *tempTopicPrefix*karakterleriyle başlayan bir konu oluşturur; burada *tempTopicPrefix* , bağlantı üreticisinin TEMPTOPICPREFIX özelliğinin değeridir.

Birörnek kaynak tanıtıcıları (URI ' ler)

Kuyruk URI 'si, bir IBM MQ kuyruğunun adını ve isteğe bağlı olarak, kuyruğun sahibi olan kuyruk yöneticisinin adını ve uygulama tarafından yaratılan Kuyruk nesnesinin bir ya da daha çok özelliğini belirten bir dizedir. Konu URI 'si, bir konunun adını ve isteğe bağlı olarak uygulama tarafından yaratılan Konu nesnesinin bir ya da daha fazla özelliğini belirten bir dizedir.

Bir kuyruk URI 'si şu biçimdedir:

```
queue://[ qMgrName ]/qName [? propertyName1 = propertyValue1
& propertyName2 = propertyValue2
&...]
```

Konu URI 'si şu biçimdedir:

```
topic://topicName [? propertyName1 = propertyValue1
& propertyName2 = propertyValue2
&...]
```

Bu biçimlerdeki değişkenler aşağıdaki anlamlara sahiptir:

qMgrAd
URI ile tanıtılan kuyruğun iyesi olan kuyruk yöneticisinin adı.

Kuyruk yöneticisi, yerel kuyruk yöneticisini ya da uzak kuyruk yöneticisini alabilir. Uzak bir kuyruk yöneticisiyse, IBM MQ yapılandırılmalıdır; böylece, bir uygulama kuyruğa ileti gönderdiğinde, WebSphere MQ iletiyi yerel kuyruk yöneticisinden uzak kuyruk yöneticisine yöneltir.

Ad belirtilmezse, yerel kuyruk yöneticisi varsayılır.

qName
IBM MQ kuyruğunun adı.

Kuyruk bir yerel kuyruk, diğer ad kuyruğu ya da uzak kuyruk tanımlaması olabilir.

Kuyruk adları yaratmaya ilişkin kurallar için IBM MQ nesnelerini adlandırma kurallarıbaşlıklı konuya bakın.

topicName
Konunun adı.

Konu adları yaratmaya ilişkin kurallar için IBM MQ nesnelerini adlandırma kurallarıbaşlıklı konuya bakın. +, #, * ve? joker karakterlerini kullanmaktan kaçının Konu adlarında. Bu karakterleri içeren konu adları, bunlara abone olduğunuzda beklenmeyen sonuçlara neden olabilir. Bkz. Konu dizgilerini birleştirme.

***propertyName1*, *propertyName2*, ...**
Uygulama tarafından yaratılan Kuyruk ya da Konu nesnesinin özelliklerinin adları. Çizelge 35 sayfa 215 içinde, bir URI ' de kullanılabilecek geçerli özellik adları listelenir.

Herhangi bir özellik belirtilmezse, Kuyruk ya da Konu nesnesi tüm özellikleri için varsayılan değerlere sahiptir.

***propertyValue1*, *propertyValue2*, ...**
Uygulama tarafından yaratılan Kuyruk ya da Konu nesnesinin özelliklerinin değerleri. Çizelge 35 sayfa 215 içinde, bir URI ' de kullanılabilecek geçerli özellik değerleri listelenir.

Parantezler ([]) isteğe bağlı bir bileşeni, üç nokta (...) ise özellik ad-değer çiftlerinin listesinin bir ya da daha fazla ad-değer çifti içerebileceği anlamına gelir.

Çizelge 35 sayfa 215 içinde, kuyrukta ve konu URI ' lerinde kullanılabilecek geçerli özellik adları ve geçerli değerler listelenir. IBM MQ JMS yönetim aracı, özellik değerleri için simgesel sabitler kullansa da, URI ' ler simgesel sabitler içeremez.

Çizelge 35. Kuyrukta ve konu URI ' lerinde kullanılmak üzere özellik adları ve geçerli değerler

Özellik adı	**Açıklama**	**Geçerli değerler**
CCSID	IBM MQ classes for JMS iletiyi hedefe ilettiğinde, iletinin gövdesindeki karakter verilerinin nasıl temsil edileceği	• IBM MQtarafından desteklenen herhangi bir kodlanmış karakter takımı tanıtıcısı.
Kodlama	IBM MQ classes for JMS iletiyi hedefe ilettiğinde, iletinin gövdesindeki sayısal verilerin nasıl temsil edileceği	• IBM MQ ileti tanımlayıcısındaki *Kodlama* alanı için geçerli herhangi bir değer.
Son kullanma tarihi	Hedefe gönderilen iletiler için yaşam süresi	• -2-send () çağrısında belirtildiği gibi ya da send () çağrısında belirtilmezse, ileti üreticisinin varsayılan yaşam süresi. • 0-Hedefe gönderilen bir iletinin süresi hiçbir zaman dolmaz. • Milisaniye cinsinden yaşam süresini belirten pozitif bir tamsayı.

Çizelge 35. Kuyrukta ve konu URI ' lerinde kullanılmak üzere özellik adları ve geçerli değerler (devamı var)		
Özellik adı	**Açıklama**	**Geçerli değerler**
çoklu yayın	Bir aracıya gerçek zamanlı bağlantı kullanılırken bir konuya ilişkin çoklu yayın ayarı	Aşağıdaki liste geçerli değerleri içerir. Her bir değerle ilişkilendirilmiş, IBM MQ JMS yönetim aracında kullanılan MULTICAST özelliğinin karşılık gelen değeridir. MULTICAST özelliğinin açıklaması ve geçerli değerleri için bkz. IBM MQ classes for JMS nesnelerinin özellikleri. • -1-ASCF • 0-DEVRE Dışı • 3-HAYıR • 5-GüVENILIR • 7-ETKIN
Kalıcılık	Hedefe gönderilen iletilerin kalıcılığı	• -2-send () çağrısında belirtildiği gibi ya da send () çağrısında belirtilmezse, ileti üreticisinin varsayılan kalıcılığını belirler. • -1- IBM MQ kuyruğunun ya da konusunun *DefPersistence* (DefPersistence) özniteliği tarafından belirtildiği şekilde. • 1-Kalıcı Değil. • 2-Kalıcı. • 3- IBM MQ JMS yönetim aracında kullanılan PERSISTENCE özelliği için HIGH değerinin eşdeğeri. Bu değerin açıklaması için bkz. “JMS kalıcı iletiler” sayfa 244.
öncelik	Hedefe gönderilen iletilerin önceliği	• -2-send () çağrısında belirtildiği gibi ya da send () çağrısında belirtilmezse, ileti üreticisinin varsayılan önceliği. • -1- IBM MQ kuyruğunun ya da konusunun *DefPriority* özniteliği tarafından belirtildiği şekilde. • Hedefe gönderilen iletilerin önceliğini belirten 0-9 aralığında bir tamsayı.
targetClient	Hedefe gönderilen iletilerin MQRFH2 üstbilgisi içerip içermediğini belirler	• 0-İletiler bir MQRFH2 üstbilgisi içeriyor. • 1-İletiler MQRFH2 üstbilgisi içermiyor.

Örneğin, aşağıdaki URI, yerel kuyruk yöneticisinin sahip olduğu Q1 adlı bir IBM MQ kuyruğunu tanıtır. Bu URI kullanılarak yaratılan bir Kuyruk nesnesi, tüm özellikleri için varsayılan değerlere sahiptir.

```
queue:///Q1
```

Aşağıdaki URI, QM2adlı bir kuyruk yöneticisinin sahip olduğu Q2 adlı bir IBM MQ kuyruğunu tanımlar. Bu hedefe gönderilen tüm iletilerin önceliği 6 'dır. Bu URI kullanılarak yaratılan Kuyruk nesnesinin geri kalan özellikleri varsayılan değerlerini içerir.

```
queue://QM2/Q2?priority=6
```

Aşağıdaki URI, Spor/Atletizm/Sonuçlar adlı bir konuyu tanımlar. Bu hedefe gönderilen tüm iletiler kalıcı değildir ve 0 önceliğine sahiptir. Bu URI kullanılarak yaratılan Konu nesnesinin geri kalan özelliklerinin varsayılan değerleri vardır.

```
topic://Sport/Athletics/Results?persistence=1&priority=0
```

JMS uygulamasında ileti gönderilmesi

Bir JMS uygulamasının bir hedefe ileti gönderebilmesi için önce hedef için bir MessageProducer nesnesi yaratması gerekir. Hedefe ileti göndermek için uygulama bir İleti nesnesi yaratır ve MessageProducer nesnesinin send () yöntemini çağırır.

Bir uygulama, ileti göndermek için MessageProducer nesnesini kullanır. Bir uygulama normalde belirli bir hedef için bir MessageProducer nesnesi yaratır; bu bir kuyruk ya da konu olabilir; böylece, ileti üreticisi kullanılarak gönderilen tüm iletiler aynı hedefe gönderilir. Bu nedenle, bir uygulamanın MessageProducer nesnesi yaratabilmesi için önce bir Kuyruk ya da Konu nesnesi yaratması gerekir. Kuyruk ya da Konu nesnesi yaratılmasıyla ilgili bilgi için aşağıdaki konulara bakın:

- “Bir JMS ya da Jakarta Messaging uygulamasında denetlenen nesneleri almak için JNDI ' yi kullanma” sayfa 198
- “IBM JMS uzantılarının kullanılması” sayfa 200
- “IBM MQ JMS uzantılarının kullanılması” sayfa 207
- “JMS uygulamasında hedef oluşturma” sayfa 213

Bir MessageProducer nesnesi yaratmak için bir uygulama, aşağıdaki örnekte gösterildiği gibi bir Oturum nesnesinin createProducer() yöntemini kullanır:

```
MessageProducer producer = session.createProducer(destination);
```

`destination` değiştirgesi, uygulamanın daha önce yarattığı bir Kuyruk ya da Konu nesnesidir.

Bir uygulamanın ileti gönderebilmesi için önce bir İleti nesnesi yaratması gerekir. Bir iletinin gövdesi uygulama verilerini içerir ve JMS beş tip ileti gövdesini tanımlar:

- Bayt
- Harita
- Nesne
- Akış
- Metin

İleti gövdesinin her tipi, İleti arabiriminin bir alt arabirimi olan kendi JMS arabirimine ve Oturum arabiriminde bu tip bir gövde ile ileti yaratmaya ilişkin bir yönteme sahiptir. Örneğin, bir metin iletisine ilişkin arabirim TextMessageolarak adlandırılır ve bir uygulama, aşağıdaki deyimde gösterildiği gibi bir metin iletisi yaratmak için bir Oturum nesnesinin createTextMessage () yöntemini kullanır:

```
TextMessage outMessage = session.createTextMessage(outString);
```

İletiler ve ileti gövdeleri hakkında daha fazla bilgi için bkz. “JMS ileti” sayfa 138.

Bir ileti göndermek için, aşağıdaki örnekte gösterildiği gibi, bir uygulama MessageProducer nesnesinin send () yöntemini kullanır:

```
producer.send(outMessage);
```

Bir uygulama, ileti alışverişi etki alanlarındaki iletileri göndermek için send () yöntemini kullanabilir. Hedefin türü, hangi ileti alışverişi etki alanının kullanıldığını belirler. Ancak, yayınlama/abone olma etki alanına özgü MessageProducer alt arabirimi olan TopicPublisher, send () yöntemi yerine kullanılabilecek bir publish () yöntemine de sahiptir. İki yöntem işlevsel olarak aynıdır.

Bir uygulama, hedef belirtilmemiş bir MessageProducer nesnesi yaratabilir. Bu durumda, uygulama send () yöntemini çağırırken hedefi belirtmelidir.

Bir uygulama bir hareket içinde ileti gönderirse, hareket kesinleştirilinceye kadar ileti hedefine teslim edilmez. Başka bir deyişle, bir uygulama aynı işlem içinde ileti gönderemez ve iletiye yanıt alamaz.

Bir hedef, bir uygulama kendisine ileti gönderdiğinde, IBM MQ classes for JMS iletiyi iletir ve kuyruk yöneticisinin iletiyi güvenli bir şekilde alıp almadığını belirlemeden denetimi uygulamaya geri döndürecek şekilde yapılandırılabilir. Buna bazen *zamanuyumsuz koyma*denir. Daha fazla bilgi için “İletileri IBM MQ classes for JMS içine zamanuyumsuz olarak koyma” sayfa 307 başlıklı konuya bakın.

JMS uygulamasında ileti alınması

Bir uygulama, ileti almak için bir ileti tüketicisi kullanır. Sürekli konu abonesi, tüketici etkin değilken gönderilenler de içinde olmak üzere bir hedefe gönderilen tüm iletileri alan bir ileti tüketicisidir. Bir uygulama, bir ileti seçici kullanarak almak istediği iletileri seçebilir ve ileti dinleyicisini kullanarak iletileri zamanuyumsuz olarak alabilir.

Bir uygulama, iletileri almak için MessageConsumer nesnesini kullanır. Bir uygulama, belirli bir hedef için bir MessageConsumer nesnesi yaratır; bu bir kuyruk ya da konu olabilir; böylece, ileti tüketicisi kullanılarak alınan tüm iletiler aynı hedeften alınır. Bu nedenle, bir uygulamanın MessageConsumer nesnesi oluşturabilmesi için önce bir Kuyruk ya da Konu nesnesi oluşturması gerekir. Kuyruk ya da Konu nesnesi yaratılmasıyla ilgili bilgi için aşağıdaki konulara bakın:

- “Bir JMS ya da Jakarta Messaging uygulamasında denetlenen nesneleri almak için JNDI ' yi kullanma” sayfa 198
- “IBM JMS uzantılarının kullanılması” sayfa 200
- “IBM MQ JMS uzantılarının kullanılması” sayfa 207
- “JMS uygulamasında hedef oluşturma” sayfa 213

Bir MessageConsumer nesnesi oluşturmak için bir uygulama, aşağıdaki örnekte gösterildiği gibi bir Oturum nesnesinin createConsumer() yöntemini kullanır:

```
MessageConsumer consumer = session.createConsumer(destination);
```

`destination` değiştirgesi, uygulamanın daha önce yarattığı bir Kuyruk ya da Konu nesnesidir.

Uygulama daha sonra, aşağıdaki örnekte gösterildiği gibi, hedeften bir ileti almak için MessageConsumer nesnesinin receive () yöntemini kullanır:

```
Message inMessage = consumer.receive(1000);
```

receive () çağrısındaki parametre, herhangi bir ileti hemen yoksa, yöntemin uygun bir iletinin ulaşmasını ne kadar süreyle bekleyeceğini milisaniye cinsinden belirtir. Bu parametreyi atlarsanız, arama uygun bir ileti gelinceye kadar süresiz olarak bloke olur. Uygulamanın bir ileti beklemesini istemiyorsanız, bunun yerine receiveNoWait () yöntemini kullanın.

Receive () yöntemi belirli bir tipte ileti döndürür. Örneğin, bir uygulama bir metin iletisi aldığında, receive () çağrısıyla döndürülen nesne bir TextMessage nesnesidir.

Ancak, bir receive () çağrısının döndürdüğü bildirilmiş nesne tipi bir İleti nesnesidir. Bu nedenle, yeni alınan bir iletinin gövdesinden verileri almak için uygulamanın İleti sınıfından daha belirli bir alt sınıfa (örneğin, TextMessage) yayınlanması gerekir. İletinin tipi bilinmezse, uygulama tipi belirlemek

için `instanceof` işlecini kullanabilir. Hataların düzgün bir şekilde işlenebilmesini sağlamak için, bir uygulamanın dönüştürme işleminden önce ileti tipini belirlemesi her zaman iyi bir uygulamadır.

Aşağıdaki kod, `instanceof` işlecini kullanır ve bir metin iletisinin gövdesinden verilerin nasıl çıkarılacağını gösterir:

```
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");
}
```

Bir uygulama bir hareket içinde ileti gönderirse, hareket kesinleştirilinceye kadar ileti hedefine teslim edilmez. Başka bir deyişle, bir uygulama aynı işlem içinde ileti gönderemez ve iletiye yanıt alamaz.

Bir ileti tüketicisi, önden okuma için yapılandırılmış bir hedeften ileti alırsa, uygulama sona erdiğinde önden okuma arabelleğindeki kalıcı olmayan iletiler atılır.

Yayınlama/abone olma etki alanında JMS , aşağıdaki iki bölümde açıklanan iki tip ileti tüketicisini, sürekli olmayan konu abonesini ve sürekli konu abonesini tanımlar.

Sürekli olmayan konu aboneleri

Sürekli olmayan bir konu abonesi, yalnızca abone etkinken yayınlanan iletileri alır. Bir uygulama kalıcı olmayan bir konu abonesi oluşturduğunda ve uygulama aboneyi kapattığında ya da abonenin kapsamı dışında kaldığında, kalıcı olmayan bir abonelik başlatılır. IBM MQ classes for JMSiçinde bir uzantı olarak, sürekli olmayan bir konu abonesi de alıkonan yayınları alır.

Kalıcı olmayan bir konu abonesi yaratmak için, bir uygulama hedef olarak bir Konu nesnesini belirterek etki alanından bağımsız createConsumer() yöntemini kullanabilir. Alternatif olarak, bir uygulama aşağıdaki örnekte gösterildiği gibi etki alanına özgü createSubscriber() yöntemini kullanabilir:

```
TopicSubscriber subscriber = session.createSubscriber(topic);
```

`topic` değiştirgesi, uygulamanın daha önce yarattığı bir Konu nesnesidir.

Sürekli konu aboneleri

Sınırlama: Bir uygulama, bir aracıya gerçek zamanlı bağlantı kullanırken sürekli konu aboneleri yaratamazdı.

Sürekli konu abonesi, sürekli abonelik süresi boyunca yayınlanan tüm iletileri alır. Bu iletiler, abone etkin değilken yayınlanan tüm iletileri içerir. IBM MQ classes for JMSiçinde bir uzantı olarak, sürekli bir konu abonesi de alıkonan yayınları alır.

Sürekli bir konu abonesi yaratmak için, bir uygulama Oturum nesnesinin createDurableSubscriber () yöntemini aşağıdaki örnekte gösterildiği gibi kullanır:

```
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001");
```

createDurableSubscriber () çağrısında, birinci parametre uygulamanın önceden yarattığı bir Konu nesnesidir ve ikinci parametre kalıcı aboneliği tanımlamak için kullanılan bir addır.

Sürekli konu abonesi yaratmak için kullanılan oturumun ilişkili bir istemci tanıtıcısı olmalıdır. Bir oturumla ilişkilendirilen istemci tanıtıcısı, oturumu yaratmak için kullanılan bağlantıya ilişkin istemci tanıtıcısıyla aynı. İstemci tanıtıcısı, ConnectionFactory nesnesinin CLIENTID özelliği ayarlanarak belirtilebilir. Diğer bir seçenek olarak, bir uygulama Connection nesnesinin setClientID () yöntemini çağırarak istemci tanıtıcısını belirtebilir.

Sürekli bir aboneliği tanımlamak için kullanılan ad, yalnızca istemci tanıtıcısı içinde benzersiz olmalıdır ve bu nedenle istemci tanıtıcısı, sürekli bir aboneliğin tam, benzersiz tanıtıcısının bir parçasını oluşturur. Daha önce yaratılmış sürekli bir aboneliği kullanmaya devam etmek için, bir uygulamanın sürekli abonelikle ilişkili istemci tanıtıcısıyla aynı oturumu kullanarak ve aynı abonelik adını kullanarak sürekli bir konu abonesi yaratması gerekir.

Sürekli abonelik, bir uygulama, sürekli aboneliği olmayan bir istemci tanıtıcısı ve abonelik adını kullanarak sürekli bir konu abonesi yarattığında başlar. Ancak, uygulama sürekli konu abonesini kapattığında sürekli abonelik sona ermez. Sürekli aboneliği sona erdirmek için, bir uygulamanın sürekli abonelikle ilişkilendirilmiş istemci tanıtıcısıyla aynı oturum nesnesine ilişkin unsubscription () yöntemini çağırması gerekir. unsubscription () çağrısındaki parametre, aşağıdaki örnekte gösterildiği gibi abonelik adıdır:

```
session.unsubscribe("D_SUB_000001");
```

Sürekli aboneliğin kapsamı bir kuyruk yöneticisidir. Bir kuyruk yöneticisinde sürekli bir abonelik varsa ve başka bir kuyruk yöneticisine bağlı bir uygulama aynı istemci tanıtıcısı ve abonelik adıyla sürekli bir abonelik oluştururursa, bu iki sürekli abonelik tamamen bağımsızdır.

İleti seçiciler

Bir uygulama, birbirini izleyen receive () çağrıları tarafından yalnızca belirli ölçütlere uyan iletilerin döndürüleceğini belirtebilir. Bir MessageConsumer nesnesi oluştururken uygulama, hangi iletilerin alındığını belirleyen bir SQL (Yapılandırılmış Sorgu Dili) ifadesi belirtebilir. Bu SQL ifadesine *ileti seçici*denir. İleti seçici, JMS ileti üstbilgisi alanlarının ve ileti özelliklerinin adlarını içerebilir. İleti seçicinin nasıl oluşturulacağına ilişkin bilgi için bkz. “JMS içindeki ileti seçiciler” sayfa 139.

Aşağıdaki örnekte, bir uygulamanın myPropadlı kullanıcı tanımlı bir özelliğe dayalı olarak iletileri nasıl seçebileceği gösterilmektedir:

```
MessageConsumer consumer;
.
consumer = session.createConsumer(destination, "myProp = 'blue'");
```

JMS belirtimi, bir uygulamanın ileti tüketicisinin ileti seçicisini değiştirmesine izin vermez. Bir uygulama ileti seçici ile bir ileti tüketicisi oluşturduktan sonra, ileti seçici o tüketicinin ömrü boyunca kalır. Bir uygulama birden çok ileti seçici gerektiriyorsa, uygulamanın her ileti seçici için bir ileti tüketicisi yaratması gerekir.

Bir uygulama Sürüm 7 kuyruk yöneticisine bağlandığında, bağlantı üreticisinin MSGSELECTION özelliğinin bir etkisi olmadığını unutmayın. Başarımı en iyi duruma getirmek için, tüm ileti seçimi kuyruk yöneticisi tarafından yapılır.

Yerel yayınları engelleme

Bir uygulama, tüketicinin kendi bağlantısında yayınlanan yayınları yoksayan bir ileti tüketicisi oluşturabilir. Uygulama bunu, aşağıdaki örnekte gösterildiği gibi, `true`için createConsumer() çağrısında üçüncü parametreyi ayarlayarak yapar:

```
MessageConsumer consumer = session.createConsumer(topic, null, true);
```

createDurableSubscriber () çağrısında, uygulama bunu aşağıdaki örnekte gösterildiği gibi dördüncü parametreyi `true`olarak ayarlayarak yapar.

```
String selector = "company = 'IBM'";
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001",
                                                              selector, true);
```

Zamanuyumsuz ileti teslimi

Bir uygulama, ileti dinleyicisini ileti tüketicisine kaydettirerek iletileri zamanuyumsuz olarak alabilir. İleti dinleyicinin onMessageadlı bir yöntemi vardır; bu yöntem, uygun bir ileti kullanılabilir olduğunda ve amacı iletiyi işlemek olduğunda zamanuyumsuz olarak adlandırılır. Aşağıdaki kod mekanizmayı gösterir:

V 9.3.0 V 9.3.0 JM 3.0

```
import jakarta.jms.*;

public class MyClass implements MessageListener
{
  // The method that is called asynchronously when a suitable message is available
  public void onMessage(Message message)
  {
    System.out.println("Message is "+message);

    // The code to process the message
    .
    .
    .
  }
}
.
.
.
// Main program (possibly in another class)
.
// Creating the message listener
MyClass listener = new MyClass();

// Registering the message listener with a message consumer
consumer.setMessageListener(listener);

// The main program now continues with other processing
```

JMS 2.0

```
import javax.jms.*;

public class MyClass implements MessageListener
{
  // The method that is called asynchronously when a suitable message is available
  public void onMessage(Message message)
  {
    System.out.println("Message is "+message);

    // The code to process the message
    .
    .
    .
  }
}
.
.
.
// Main program (possibly in another class)
.
// Creating the message listener
MyClass listener = new MyClass();

// Registering the message listener with a message consumer
consumer.setMessageListener(listener);

// The main program now continues with other processing
```

Bir uygulama, iletileri alma () çağrılarını kullanarak zamanuyumlu olarak almak ya da ileti dinleyicilerini kullanarak zamanuyumsuz olarak ileti almak için oturum kullanabilir, ancak her ikisi için kullanamaz. Bir uygulamanın iletileri zamanuyumlu ve zamanuyumsuz olarak alması gerekiyorsa, ayrı oturumlar oluşturması gerekir.

Bir oturum zamanuyumsuz olarak ileti alacak şekilde ayarlandığında, o oturumda ya da o oturumdan yaratılan nesnelerde aşağıdaki yöntemler çağrılamaz:

- MessageConsumer.receive ()

- MessageConsumer.receive (uzun)
- MessageConsumer.receiveNoWait ()
- Session.acknowledge()
- MessageProducer.send (Hedef, İleti)
- MessageProducer.send (Hedef, İleti, int, int, long)
- MessageProducer.send (İleti)
- MessageProducer.send (İleti, int, int, long)
- MessageProducer.send (Hedef, İleti, CompletionListener)
- MessageProducer.send (Destination, Message, int, int, long, CompletionListener)
- MessageProducer.send (İleti, CompletionListener)
- MessageProducer.send (Message, int, int, long, CompletionListener)
- Session.commit()
- Session.createBrowser(Kuyruk)
- Session.createBrowser(Kuyruk, Dizgi)
- Session.createBytesMessage()
- Session.createConsumer(Hedef)
- Session.createConsumer(Hedef, Dizgi, Boole)
- Session.createDurableSubscriber(Konu, Dizgi)
- Session.createDurableSubscriber(Konu, Dizgi, Dizgi, Boole)
- Session.createMapMessage()
- Session.createMessage()
- Session.createObjectMessage()
- Session.createObjectMessage(Diziselleştirilebilir)
- Session.createProducer(Hedef)
- Session.createQueue(Dizgi)
- Session.createStreamMessage()
- Session.createTemporaryQueue()
- Session.createTemporaryTopic()
- Session.createTextMessage()
- Session.createTextMessage(Dizgi)
- Session.createTopic()
- Session.getAcknowledgeMode()
- Session.getMessageListener()
- Session.getTransacted()
- Session.rollback()
- Session.unsubscribe(Dizgi)

Bu yöntemlerden herhangi biri çağrılırsa, iletiyi içeren JMSException:

```
JMSCC0033: Bir oturum zamanuyumsuz olarak kullanıldığında zamanuyumlu yöntem çağrılmasına izin
verilmez: 'yöntem adı'
```

atılır.

Zehirli mesajlar alma

Bir uygulama işlenemeyen bir ileti alabilir. İletinin işlenememesinin birkaç nedeni olabilir; örneğin, iletinin biçimi yanlış olabilir. Bu tür iletiler zehirli mesajlar olarak tanımlanır ve iletinin özyinelemeli olarak işlenmesini önlemek için özel işlem gerektirir.

Zehirli iletilerin nasıl işleneceğine ilişkin ayrıntılar için bkz. “IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 225.

Alınan iletilere uyacak şekilde arabellek boyutlarını uyarlama

JMS dışı bir uygulama tarafından IBM MQ ' den bir ileti alındığında, iletinin içine yazılması için uygulama tarafından bir ileti arabelleği sağlanmalıdır. JMS uygulamalarının el ile arabellek yaratması gerekmez. IBM MQ classes for JMS otomatik olarak ileti arabelleklerini, alınan iletilerin boyutlarına uyacak şekilde oluşturur ve boyutlandırır. Çoğu uygulama için, otomatik olarak yönetilen arabellekler, uygulama geliştiricisi için uygun bir performans dengesi ve kolaylık sağlar. Bazı durumlarda, ileti arabelleğinin ilk büyüklüğünü el ile belirtmek yararlı olabilir. IBM MQ JMS alma arabelleğinin varsayılan ilk boyutu 4 KB 'dir. Bir uygulama her zaman 256 KB ' lik iletileri alacaksa, ilk arabellek büyüklüğünü 256 KB olarak yapılandırmayı tercih edebilir. Bu, IBM MQ classes for JMS 'in iletiyi 256 KB' ye yeniden boyutlandırmadan ve başarıyla almadan önce 4 KB arabelleğe almayı denemesi ve almaması gereksinimini önleyebilir. İstemciye bağlı bir uygulama için bu, IBM MQ classes for JMS kullanılacak doğru arabellek boyutunu belirlerken potansiyel olarak boşa gitmiş bir ağ gidiş geliş gereksinmesini önleyebilir.

İlk arabellek büyüklüğü, com.ibm.mq.jmqi.defaultMaxMsgSize Java özelliği, bayt cinsinden seçtiğiniz değere ayarlanarak yapılandırılabilir. Bu özelliğin Java Virtual Machineiçinde çalışan tüm IBM MQ JMS uygulamalarını etkilediğini unutmayın; bu nedenle, farklı büyüklükteki iletileri alan diğer ileti tüketicilerini olumsuz etkilememeye dikkat edin.

IBM MQ classes for JMS , yapılandırılan büyüklükten daha az sayıda ileti alınırsa, arabelleğin büyüklüğünü otomatik olarak azaltmayı dener. Varsayılan olarak, tümü arabellek büyüklüğünden küçük olan 10 ileti alındığında bu durum oluşur. Örneğin, 128 KB 'lik bir satırda 10 ileti alınırsa, arabellek 256 KB' den 128 KB ' ye düşürülür. Daha sonra daha büyük iletiler alındığında yeniden artırılır. Arabellek büyüklüğünün azaltılmasından önce alınması gereken ileti sayısı yapılandırılabilir. Örneğin, uygulamanın beş büyük ileti ve ardından 10 küçük ileti ve daha sonra beş büyük ileti aldığı biliniyorsa bu yararlı olabilir. Varsayılan ayarlarla, arabellek 10 küçük ileti alındıktan sonra azaltılır ve daha büyük iletiler için yeniden artırılması gerekir. Java Sistem özelliği com.ibm.mq.jmqi.smallMsgBufferReductionThreshold , arabellek boyutu azaltılmadan önce alınması gereken ileti sayısına ayarlanabilir. Bu örnekte, 10 daha küçük iletinin arabellek boyutunu azaltmasını önlemek için 20 olarak ayarlanabilir.

Özellikler birbirinden bağımsız olarak ayarlanabilir. Örneğin, ilk arabellek boyutunu varsayılan 4 KB değerine bırakmayı, ancak com.ibm.mq.jmqi.smallMsgBufferReductionThreshold değerini artırmayı seçebilirsiniz; böylece, arabellek artırıldığında o büyüklük daha uzun süre kalır.

MQI istatistik kayıtlarındaki JMS uygulamalarınız için çok sayıda MQRC_TRUNCATED_MSG_FAILED (2080) dönüş kodu görülürse, bu, bu uygulamalar için daha yüksek bir başlangıç arabellek boyutu yapılandırmanızdan ya da arabellek boyutlarının azaltılmasının sıklığını azaltmanızdan yararlanacağınız bir gösterge olabilir. Ancak, uzun süre çalışan bir uygulama için büyük olasılıkla çok az sayıda MQRC_TRUNCATED_MSG_FAILED dönüş kodu görebileceksiniz. Bunun nedeni, genellikle arabelleğin ilk büyük ileti alındıktan hemen sonra doğru boyuta yükseltilmesi ve daha küçük iletiler alınmadıkça boyutunun azaltılmamasından kaynaklanır. Bu nedenle, çok sayıda MQRC_TRUNCATED_MSG_FAILED, bağlantıyı kesmeden önce yalnızca bir ya da iki ileti almak üzere IBM MQ ' e bağlanmak gibi diğer kötü uygulama uygulamalarını gösteriyor olabilir.

Abonelik kullanıcı verilerinin alınması

Bir IBM MQ classes for JMS uygulamasının bir kuyruktan tükettiği iletiler, yönetici tarafından tanımlanan sürekli bir abonelik tarafından konursa, uygulamanın abonelikle ilişkili kullanıcı verileri bilgilerine erişmesi gerekir. Bu bilgiler iletiye özellik olarak eklenir.

MQPS klasörü ile RFH2 üstbilgisi içeren bir kuyruktan bir ileti tüketildiğinde, Sud anahtarıyla ilişkili değer varsa, IBM MQ classes for JMS uygulamasına döndürülen JMS Message nesnesine String özelliği olarak eklenir. Bu özelliğin iletiden alınmasını etkinleştirmek için, JmsConstants arabirimindeki

JMS_IBM_SUBSCRIPTION_USER_DATA değişmezi, abonelik kullanıcı verilerini almak üzere aşağıdaki yöntemle kullanılabilir:

- V 9.3.0 V 9.3.0 JM 3.0 jakarta.jms.Message.getStringProperty(java.lang.String)
- JMS 2.0 javax.jms.Message.getStringProperty(java.lang.String)

Aşağıdaki örnekte, **DEFINE SUB**MQSC komutu kullanılarak sürekli denetim aboneliği tanımlanmıştır:

```
DEFINE SUB('MY.SUBCRIPTION') TOPICSTR('PUBLIC') DEST('MY.SUBSCRIPTION.Q')
USERDATA('Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q')
```

PUBLIC konu dizgisine yayınlanan iletilerin kopyaları kuyruğa yerleştirilir (MY.SUBSCRIPTION.Q). Daha sonra sürekli abonelikle ilişkili kullanıcı verileri, RFH2 üstbilgisinin MQPS klasöründe Sudanahtarıyla saklanan iletiye bir özellik olarak eklenir.

IBM MQ classes for JMS uygulaması şunları arayabilir:

V 9.3.0 V 9.3.0 JM 3.0
```
jakarta.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

JMS 2.0
```
javax.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

Bundan sonra şu dizgi döndürülür:

```
Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q
```

İlgili kavramlar
“MQRFH2 üstbilgisi ve JMS” sayfa 143
İlgili görevler
Yönetim aboneliği tanımlanması
İlgili başvurular
ALT öğEYI TAN
Arabirim JmsConstants

IBM MQ classes for JMS uygulamasını kapatma

Bir IBM MQ classes for JMS uygulamasının durmadan önce belirli JMS nesnelerini açık bir şekilde kapatması önemlidir. Kesinleştiriciler çağrılamayabilir, bu nedenle kaynakları serbest bırakmak için bunlara güvenmeyin. Sıkıştırılmış izleme etkinken bir uygulamanın sonlanmasına izin vermeyin.

Özellikle bir uygulama oturum düzeyinde ya da daha düşük düzeyde çok sayıda kısa ömürlü JMS nesne oluşturuyorsa, atık toplama tek başına tüm IBM MQ classes for JMS ve IBM MQ kaynaklarını zamanında serbest bırakamaz. Bu nedenle, bir uygulamanın artık gerekli olmadığında bir Bağlantı, Oturum, MessageConsumerya da MessageProducer nesnesini kapatması önemlidir.

Bir uygulama bir Bağlantı kapatılmadan sona ererse, bağlantının tüm hareketli oturumları için örtük bir geriye işleme gerçekleşir. Uygulama tarafından yapılan değişikliklerin kesinleştirildiğinden emin olmak için, uygulamayı kapatmadan önce Bağlantıyı belirtik olarak kapatın.

JMS nesnelerini kapatmak için uygulamada sonlandırıcıları kullanmayın. Sonlandırıcılar çağrılamayabileceğinden kaynaklar serbest bırakılmayabilir. Bir Bağlantı kapatıldığında, bu bağlantıdan yaratılan tüm Oturumlar kapatılır. Benzer şekilde, Oturum kapatıldığında bir Oturumda oluşturulan MessageConsumers ve MessageProducers da kapatılır. Ancak, kaynakların zamanında serbest bırakıldığından emin olmak için Oturumları, MessageConsumersve MessageProducers öğelerini açık bir şekilde kapatmayı düşünebilirsiniz.

İzleme sıkıştırması etkinleştirilirse, System.Halt() olanağının kapanması ve olağandışı, denetimsiz JVM sonlandırmaları büyük olasılıkla bozuk bir izleme dosyasıyla sonuçlanır. Olanaklıyken, gerek duyduğunuz izleme bilgilerini topladığınızda izleme olanağını kapatın. Bir uygulamayı olağandışı bir sona kadar izliyorsanız, sıkıştırılmamış izleme çıkışını kullanın.

Not: Bir kuyruk yöneticisiyle bağlantıyı kesmek için, JMS uygulaması bağlantı nesnesinde close () yöntemini çağırır.

IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi

Zehirli bir mesaj, alan bir uygulama tarafından işlenemeyen bir mesajdır. Bir zehirli ileti bir uygulamaya teslim edilir ve belirli bir sayıda geriye işlenirse, IBM MQ classes for JMS bunu bir geriletme kuyruğuna taşıyabilir.

Zehirli ileti, alan bir uygulama tarafından işlenemeyen bir iletidir. İleti beklenmeyen bir tipe sahip olabilir ya da uygulamanın mantığı tarafından işlenemeyen bilgiler içeriyor olabilir. Bir uygulamaya zehirli bir ileti teslim edilirse, uygulama iletiyi işleyemeyecek ve geldiği kuyruğa geri döndürecektir. Varsayılan olarak IBM MQ classes for JMS , iletiyi uygulamaya sürekli olarak yeniden teslim eder. Bu, uygulamanın sürekli olarak zehirli iletiyi işlemeye ve geri almaya çalışırken bir döngüye sıkışmasına neden olabilir.

Bunun olmasını önlemek için, IBM MQ classes for JMS zehirli iletileri saptayabilir ve bunları alternatif bir hedefe taşıyabilir. Bunu yapmak için IBM MQ classes for JMS aşağıdaki özelliklerden yararlanmanızı sağlar:

- Saptanan iletiye ilişkin MQMD içindeki BackoutCount alanının değeri.
- İletiyi içeren giriş kuyruğuna ilişkin IBM MQ kuyruk öznitelikleri **BOTHRESH** (geriletme eşiği) ve **BOQNAME** (geriletme yeniden kuyruğa alma kuyruğu).

Bir ileti bir uygulama tarafından geriye işlendiğinde, kuyruk yöneticisi iletiye ilişkin BackoutCount alanının değerini otomatik olarak artırır.

IBM MQ classes for JMS , sıfırdan büyük BackoutCount değerine sahip bir ileti algıladığında, BackoutCount değerini **BOTHRESH** özniteliğinin değeriyle karşılaştırır.

- BackoutCount , **BOTHRESH** özniteliğinin değerinden küçükse, IBM MQ classes for JMS bunu işlenmek üzere uygulamaya teslim eder.
- Ancak, BackoutCount değeri **BOTHRESH**değerinden büyük ya da bu değere eşitse, ileti zehirli bir ileti olarak kabul edilir. Bu durumda IBM MQ classes for JMS , iletiyi **BOQNAME** özniteliği tarafından belirtilen kuyruğa taşır. İleti geriletme kuyruğuna konamazsa, iletinin rapor seçeneklerine bağlı olarak kuyruk yöneticisinin gitmeyen ileti kuyruğuna taşınır ya da atılır.

Not:

- **BOTHRESH** özniteliği varsayılan değeri olan 0değerine bırakılırsa, zehirli ileti işleme devre dışı bırakılır. Bu, zehirli iletilerin giriş kuyruğuna geri konması anlamına gelir.
- Dikkat edilmesi gereken diğer bir nokta da, IBM MQ classes for JMS ' in **BOTHRESH** ve **BOQNAME** özniteliklerini, BackoutCount değeri sıfırdan büyük olan bir iletiyi ilk kez algıladığında sorgulaması. Bu özniteliklerin değerleri daha sonra önbelleğe alınır ve IBM MQ classes for JMS , sıfırdan büyük BackoutCount değerine sahip bir iletiyle karşılaştığında kullanılır.

Sisteminizin zehirli ileti işlemeyi gerçekleştirecek şekilde yapılandırılması

IBM MQ classes for JMS ' in **BOTHRESH** ve **BOQNAME** özniteliklerini sorgularken kullandığı kuyruk, gerçekleştirilmekte olan ileti alışverişi stiline bağlıdır:

- Noktadan noktaya iletişim ileti sistemi için bu, temeldeki yerel kuyruktur. Bir JMS uygulaması, diğer ad kuyruklarından ya da küme kuyruklarından gelen iletileri tüketiyorken bu önemlidir.
- Yayınlama/abone olma ileti sistemi için, bir uygulamaya ilişkin iletileri tutmak üzere yönetilen bir kuyruk yaratılır. IBM MQ classes for JMS , **BOTHRESH** ve **BOQNAME** özniteliklerine ilişkin değerleri belirlemek için yönetilen kuyruğu sorgula.

  Yönetilen kuyruk, uygulamanın abone olduğu Konu nesnesiyle ilişkili bir model kuyruğundan oluşturulur ve **BOTHRESH** ve **BOQNAME** özniteliklerinin değerlerini model kuyruğundan devralır. Kullanılan model kuyruğu, alan uygulamanın kalıcı ya da kalıcı olmayan bir aboneliği çıkarmasına bağlıdır:

  – Sürekli abonelikler için kullanılan model kuyruğu, Konunun **MDURMDL** özniteliği tarafından belirtilir. Bu özniteliğin varsayılan değeri SYSTEM.DURABLE.MODEL.QUEUE.

– Kalıcı olmayan abonelikler için, kullanılan model kuyruğu **MNDURMDL** özniteliği tarafından belirtilir. **MNDURMDL** özniteliğinin varsayılan değeri SYSTEM.NDURABLE.MODEL.QUEUE.

BOTHRESH ve **BOQNAME** özniteliklerini sorgularken IBM MQ classes for JMS:

- Yerel kuyruğu ya da diğer ad kuyruğuna ilişkin hedef kuyruğu açın.
- **BOTHRESH** ve **BOQNAME** özniteliklerini sorgulaın.
- Yerel kuyruğu ya da bir diğer ad kuyruğuna ilişkin hedef kuyruğu kapatın.

Yerel kuyruğu açarken kullanılan açma seçenekleri ya da bir diğer ad kuyruğuna ilişkin hedef kuyruk, kullanılmakta olan IBM MQ classes for JMS sürümüne bağlıdır:

- IBM MQ 9.1.0 Fix Pack 1 ve önceki sürümlerdeki IBM MQ classes for JMS için ya da IBM MQ 9.1.1, yerel kuyruk ya da bir diğer ad kuyruğuna ilişkin hedef kuyruk bir küme kuyruğuysa, IBM MQ classes for JMS kuyruğu MQOO_INPUT_AS_Q_DEF, MQOO_INQUIRE ve MQOO_FAIL_IF_QUIESCING seçenekleriyle açın. Bu, alan uygulamayı çalıştıran kullanıcının küme kuyruğunun yerel eşgörünümüne ilişkin sorma ve erişim yetkisine sahip olması gerektiği anlamına gelir.

  IBM MQ classes for JMS , MQOO_INQUIRE ve MQOO_FAIL_IF_QUIESCINGaçma seçenekleriyle diğer tüm yerel kuyruk tiplerini açar. IBM MQ classes for JMS ' in özniteliklerin değerlerini sorgulaması için, alan uygulamayı çalıştıran kullanıcının yerel kuyrukta sorgulama erişimi olmalıdır.
- IBM MQ 9.1.0 Fix Pack 2 ve sonraki sürümlerinde ya da IBM MQ 9.1.2 ve sonraki sürümlerinde IBM MQ classes for JMS kullanırken, alan uygulamayı çalıştıran kullanıcının, kuyruğun tipine bakılmaksızın, yerel kuyrukta sorgu erişimi olmalıdır.

Zehirli iletileri bir geriletme yeniden kuyruğa alma kuyruğuna ya da kuyruk yöneticisinin gitmeyen iletiler kuyruğuna taşımak için, uygulamayı çalıştıran kullanıcıya koyma ve geçiş yetkilerini vermeniz gerekir.

Zamanuyumlu uygulamalar için zehirli iletilerin işlenmesi

Bir uygulama, aşağıdaki yöntemlerden birini çağırarak iletileri zamanuyumlu olarak alırsa, IBM MQ classes for JMS , uygulama iletiyi almaya çalıştığında etkin olan iş birimi içinde bir zehirli iletiyi yeniden kuyruğa alır:

- JMSConsumer.receive()
- JMSConsumer.receive(uzun zamanaşımı)
- JMSConsumer.receiveBody(Sınıf < T> c)
- JMSConsumer.receiveBody(Sınıf < T> c, uzun zamanaşımı)
- JMSConsumer.receiveBodyNoWait Sınıfı < T> c)
- JMSConsumer.receiveNoWait()
- MessageConsumer.receive ()
- MessageConsumer.receive (uzun zamanaşımı)
- MessageConsumer.receiveNoWait ()
- QueueReceiver.receive ()
- QueueReceiver.receive (uzun zamanaşımı)
- QueueReceiver.receiveNoWait ()
- TopicSubscriber.receive ()
- TopicSubscriber.receive (uzun zamanaşımı)
- TopicSubscriber.receiveNoWait ()

Başka bir deyişle, uygulama işlemli JMS bağlamı ya da oturumu kullanıyorsa, iletinin geriletme kuyruğuna taşınması işlem kesinleştirilinceye kadar kesinleştirilmez.

BOTHRESH özniteliği sıfır dışında bir değere ayarlanırsa, **BOQNAME** özniteliği de ayarlanmalıdır. **BOTHRESH** sıfırdan büyük bir değere ayarlandıysa ve **BOQNAME** ayarlanmamışsa, davranış iletinin rapor seçenekleri tarafından belirlenir:

- İletide MQRO_DISCARD_MSG rapor seçeneği ayarlandıysa, ileti atılır.
- İletide MQRO_DEAD_LETTER_Q rapor seçeneği belirtildiyse, IBM MQ classes for JMS iletiyi kuyruk yöneticisinin ileti kuyruğuna taşımayı dener.
- İletide MQRO_DISCARD_MSG ya da MQRO_DEAD_LETTER_Q ayarlanmamışsa, IBM MQ classes for JMS iletiyi kuyruk yöneticisine ilişkin gitmeyen ileti kuyruğuna yerleştirmeyi dener.

İletiyi bir nedenden dolayı teslim edilmeyen iletiler kuyruğuna koyma girişimi başarısız olursa, iletiye ne olacağı, alan uygulamanın hareket eden bir JMS bağlamı mı, yoksa oturumu mu kullandığına göre belirlenir:

- Alan uygulama ya hareket halindeki JMS bağlamını ya da oturumunu kullanıyorsa ve işlem kesinleştirildiyse, ileti atılır.
- Alan uygulama işlemli bir JMS bağlamı ya da oturumu kullanıyorsa ve hareketi geriye işlerse, ileti giriş kuyruğuna döndürülür.
- Alan uygulama hareketsiz bir JMS bağlamı ya da oturumu yarattıysa, ileti atılır.

Zamanuyumsuz uygulamalar için zehirli iletilerin işlenmesi

Bir uygulama MessageListeneraracılığıyla zamanuyumsuz olarak ileti alıyorsa, IBM MQ classes for JMS , ileti teslimini etkilemeden zehirli iletileri yeniden kuyruğa gönderir. Yeniden kuyruğa alma işlemi, uygulamaya gerçek ileti teslimiyle ilişkili herhangi bir iş biriminin dışında gerçekleşir.

BOTHRESH sıfırdan büyük bir değere ayarlandıysa ve **BOQNAME** ayarlanmamışsa, davranış iletinin rapor seçenekleri tarafından belirlenir:

- İletide MQRO_DISCARD_MSG rapor seçeneği ayarlandıysa, ileti atılır.
- İletide MQRO_DEAD_LETTER_Q rapor seçeneği belirtildiyse, IBM MQ classes for JMS iletiyi kuyruk yöneticisinin ileti kuyruğuna taşımayı dener.
- İletide MQRO_DISCARD_MSG ya da MQRO_DEAD_LETTER_Q ayarlanmamışsa, IBM MQ classes for JMS iletiyi kuyruk yöneticisine ilişkin gitmeyen ileti kuyruğuna yerleştirmeyi dener.

İletiyi bir nedenle gelmeyen iletiler kuyruğuna koyma girişimi başarısız olursa, IBM MQ classes for JMS iletiyi giriş kuyruğuna döndürür.

Etkinleştirme belirtimlerinin ve ConnectionConsumers özelliğinin zehirli iletileri nasıl işlediğine ilişkin bilgi için ASF ' de kuyruktan ileti kaldırılmasıbaşlıklı konuya bakın.

Geriletme kuyruğuna taşındığında iletiye ne olur?

Bir zehirli ileti, geriletme yeniden kuyruğa yeniden kuyruğa eklendiğinde, IBM MQ classes for JMS buna bir RFH2 üstbilgisi ekler (önceden yoksa) ve ileti tanımlayıcısı (MQMD) içindeki bazı alanları günceller.

Zehirli ileti bir RFH2 üstbilgisi içeriyorsa (örneğin, bir JMS iletisi olduğu için), IBM MQ classes for JMS iletiyi geriletme yeniden kuyruğa taşırken MQMD içinde aşağıdaki alanları değiştirir:

- BackoutCount alanı sıfırlanır.
- İletinin Süre Bitimi alanı, JMS uygulaması tarafından zehirli iletinin alındığı sırada kalan süre bitimini yansıtacak şekilde güncellenir.

Zehirli ileti bir RFH2 üstbilgisi içermiyorsa, IBM MQ classes for JMS bir tane ekleyin ve geriletme işleminin bir parçası olarak MQMD ' de aşağıdaki alanları güncelleyin:

- BackoutCount alanı sıfırlanır.
- İletinin Süre Bitimi alanı, JMS uygulaması tarafından zehirli iletinin alındığı sırada kalan süre bitimini yansıtacak şekilde güncellenir.
- İletinin Biçim alanı MQHRF2olarak değiştirildi.
- CCSID alanı 1208 olarak değiştirilir.
- Kodlama alanı 273 olacak şekilde değiştirildi.

Buna ek olarak, zehirli iletideki CCSID ve Kodlama alanları, geriletme yeniden kuyruğa alma kuyruğundaki iletinin üstbilgi zincirinin doğru olduğundan emin olmak için RFH2 üstbilgisinin CCSID ve Kodlama alanlarına kopyalanır.

İlgili kavramlar

“ASF ' de zehirli iletilerin işlenmesi” sayfa 324

Uygulama Sunucusu Tesislerinde, zehirli ileti işleme IBM MQ classes for JMS' in başka yerlerine göre biraz daha farklı şekilde ele alınır.

IBM MQ classes for JMS içindeki kural dışı durumlar

IBM MQ classes for JMS uygulaması, JMS API çağrıları tarafından yayınlanan ya da bir kural dışı durum işleyicisine teslim edilen kural dışı durumları işlemelidir.

IBM MQ classes for JMS , kural dışı durumlar yayınlayarak çalıştırma zamanı sorunlarını bildirir. Yayınlanan kural dışı durumların tipi ve bu kural dışı durumların nasıl işlenmesi gerektiği, uygulamanız tarafından kullanılan JMS belirtiminin sürümüne bağlıdır:

- JMS 1.1 ve daha önceki yayınlarda tanımlanan arabirimlerdeki yöntemler denetlendi. Bu kural dışı durumların temel sınıfı şudur: `JMSException`. Denetlenen kural dışı durumların nasıl işleneceğine ilişkin daha fazla bilgi için bkz. “Denetlenen kural dışı durumların işlenmesi” sayfa 228.
- JMS 2.0 içinde eklenen arabirimlerdeki yöntemler, denetimsiz kural dışı durumlar yayınlıyor. Bu kural dışı durumların temel sınıfı şudur: `JMSRuntimeException`. Denetimsiz kural dışı durumların nasıl işleneceğine ilişkin daha fazla bilgi için bkz. “Denetimsiz kural dışı durumların işlenmesi” sayfa 231.

`ExceptionListener` ' i bir JMS `Connection` ya da `JMSContext`ile de kaydedebilirsiniz. JMS için MQ sınıfları daha sonra, kuyruk yöneticisiyle bağlantıda bir sorun saptanırsa ya da bir iletiyi zamanuyumsuz olarak teslim etme girişimi sırasında bir sorun ortaya çıkarsa `ExceptionListener` ' e bildirim gönderir. Daha fazla bilgi için bkz “ExceptionListeners” sayfa 234.

İlgili kavramlar

JMS için IBM MQ sınıfları

İlgili başvurular

ZAMANUYUMSUZ KURAL Dışı DURUMU

Denetlenen kural dışı durumların işlenmesi

JMS 1.1 ya da daha önceki yayın düzeylerinde tanımlanan arabirimlerdeki yöntemler denetimli kural dışı durumlar verdi. Bu kural dışı durumların temel sınıfı şudur: `JMSException`. Bu nedenle, `JMSExceptions` ' in yakalanması bu kural dışı durum tiplerini işlemek için soysal bir yol sağlar.

Her `JMSException` aşağıdaki bilgileri içerir:

- Uygulamanızın `Throwable.getMessage()` yöntemini çağırarak elde edebileceği, sağlayıcıya özgü bir kural dışı durum iletisi.
- Uygulamanızın `JMSException.getErrorCode()` yöntemini çağırarak elde edebileceği, sağlayıcıya özgü bir hata kodu.
- Bağlantılı bir kural dışı durum. JMS 1.1 API çağrısı tarafından yayınlanan bir kural dışı durum, genellikle bu kural dışı durumla bağlantılı başka bir kural dışı durum tarafından bildirilen daha düşük düzeyli bir sorunun sonucudur. Uygulamanız, `JMSException.getLinkedException()` yöntemini ya da `Throwable.getCause()` yöntemini çağırarak bağlantılı bir kural dışı durum alabilir.

JMS 1.1 API 'sini kullandığınızda, IBM MQ classes for JMS tarafından yayınlanan kural dışı durumların çoğu `JMSException`alt sınıflarının eşgörünümleridir. Bu alt sınıflar, aşağıdaki ek bilgileri sağlayan `com.ibm.msg.client.jms.JmsExceptionDetail` arabirimini gerçekleştirir:

- Kural dışı durum iletisinin açıklaması. Uygulamanız, `JmsExceptionDetail.getExplanation()` yöntemini çağırarak bu iletiyi alabilir.
- Kural dışı duruma önerilen bir kullanıcı yanıtı. Uygulamanız, `JmsExceptionDetail.getUserAction()` yöntemini çağırarak bu iletiyi alabilir.

- İletiye ilişkin anahtarlar kural dışı durum iletisine eklenir. Uygulamanız, `JmsExceptionDetail.getKeys()` yöntemini çağırarak tüm anahtarlar için bir yineleyici elde edebilir.
- İleti kural dışı durum iletisine eklenir. Örneğin, ileti ekleme işlemi kural dışı duruma neden olan kuyruğun adı olabilir ve uygulamanızın bu ada erişmesi yararlı olabilir. Uygulamanız, `JmsExceptionDetail.getValue()` yöntemini çağırarak, belirtilen bir anahtara karşılık gelen ileti eklemesini alabilir.

Ayrıntı yoksa, `JmsExceptionDetail` arabirimindeki tüm yöntemler boş değer döndürür.

Örneğin, bir uygulama var olmayan bir IBM MQ kuyruğu için ileti üreticisi yaratmayı denerse, aşağıdaki bilgilerle birlikte bir kural dışı durum yayınlanır:

```
Message : JMSWMQ2008: Failed to open MQ queue 'Q_test'.
Class : class com.ibm.msg.client.jms.DetailedInvalidDestinationException
Error Code : JMSWMQ2008
Explanation : JMS attempted to perform an MQOPEN, but IBM MQ reported an
              error.
User Action : Use the linked exception to determine the cause of this error. Check
              that the specified queue and queue manager are defined correctly.
```

Yayınlanan kural dışı durum (`com.ibm.msg.client.jms.DetailedInvalidDestinationException`), aşağıdaki sınıfın bir alt sınıfıdır ve `com.ibm.msg.client.jms.JmsExceptionDetail` arabirimini gerçekleştirir.

- V 9.3.0 V 9.3.0 JM 3.0 `jakarta.jms.InvalidDestinationException`
- JMS 2.0 `javax.jms.InvalidDestinationException`

Bağlantılı kural dışı durumlar

Bağlantılı bir kural dışı durum, bir yürütme ortamı sorunuyla ilgili daha fazla bilgi sağlar. Bu nedenle, yayınlanan her `JMSException` için bir uygulamanın bağlantılı kural dışı durumu denetlemesi gerekir.

Bağlantılı kural dışı durumun kendisi başka bir bağlantılı kural dışı duruma sahip olabilir ve bağlantılı kural dışı durumlar, özgün temel soruna yol açan bir zincir oluşturur. Bağlantılı bir kural dışı durum, `java.lang.Throwable` sınıfının zincirleme kural dışı durum düzeneği kullanılarak gerçekleştirilir ve uygulamanız `Throwable.getCause()` yöntemini çağırarak bağlantılı bir kural dışı durum alabilir. `JMSException`için, `getLinkedException()` yöntemi `Throwable.getCause()` yöntemine yetki aktarmaktadır.

Örneğin, bir uygulama bir kuyruk yöneticisine bağlanırken yanlış bir kapı numarası belirtirse, kural dışı durumlar aşağıdaki zinciri oluşturur:

```
com.ibm.msg.client.jms.DetailIllegalStateException
|
+--->
    com.ibm.mq.MQException
    |
    +--->
        com.ibm.mq.jmqi.JmqiException
        |
        +--->
            com.ibm.mq.jmqi.JmqiException
            |
            +--->
                java.net.ConnectionException
```

Genellikle, bir zincirdeki her kural dışı durum, koddaki farklı bir katmandan yayınlanır. Örneğin, önceki zincirdeki kural dışı durumlar aşağıdaki katmanlar tarafından yayınlanır:

- `JMSException`alt sınıfının bir eşgörünümü olan ilk kural dışı durum, IBM MQ classes for JMSiçindeki ortak katman tarafından yayınlanır.
- Sonraki kural dışı durum (bir `com.ibm.mq.MQException`örneği), IBM MQ ileti alışverişi sağlayıcısı tarafından yayınlanır.

- Her ikisi de `com.ibm.mq.jmqi.JmqiException`eşgörünümü olan sonraki iki kural dışı durum, Java İleti Kuyruğa Alma Arabirimi (JMQI) tarafından yayınlanır. JMQI, IBM MQ classes for JMS tarafından bir kuyruk yöneticisiyle iletişim kurmak için kullanılan bileşendir.
- Son kural dışı durum (bir `java.net.ConnectionException`eşgörünümü), Java sınıf kitaplığı tarafından yayınlanır.

IBM MQ classes for JMSkatmanlı mimarisi hakkında daha fazla bilgi için bkz. IBM MQ JMS mimarisi sınıfları.

Aşağıdaki örnekte gösterildiği gibi, uygulamanızı, uygun tüm bilgileri çıkarmak için bu zincir boyunca yinelenecek şekilde kodlayabilirsiniz:

V 9.3.0 V 9.3.0 JM 3.0

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import jakarta.jms.JMSException;
.
.
.
catch (JMSException je) {
  System.err.println("Caught JMSException");
  // Check for linked exceptions in JMSException
  Throwable t = je;
  while (t != null) {
    // Write out the message that is applicable to all exceptions
    System.err.println("Exception Msg: " + t.getMessage());
    // Write out the exception stack trace
    t.printStackTrace(System.err);

    // Add on specific information depending on the type of exception
    if (t instanceof JMSException) {
      JMSException je1 = (JMSException) t;
      System.err.println("JMS Error code: " + je1.getErrorCode());
      if (t instanceof JmsExceptionDetail){
        JmsExceptionDetail jed = (JmsExceptionDetail)je1;
        System.err.println("JMS Explanation: " + jed.getExplanation());
        System.err.println("JMS Explanation: " + jed.getUserAction());
      }
    } else if (t instanceof MQException) {
      MQException mqe = (MQException) t;
      System.err.println("WMQ Completion code: " + mqe.getCompCode());
      System.err.println("WMQ Reason code: " + mqe.getReason());
    } else if (t instanceof JmqiException){
      JmqiException jmqie = (JmqiException)t;
      System.err.println("WMQ Log Message: " + jmqie.getWmqLogMessage());
      System.err.println("WMQ Explanation: " + jmqie.getWmqMsgExplanation());
      System.err.println("WMQ Msg Summary: " + jmqie.getWmqMsgSummary());
      System.err.println("WMQ Msg User Response: " + jmqie.getWmqMsgUserResponse());
      System.err.println("WMQ Msg Severity: " + jmqie.getWmqMsgSeverity());
    }
    // Get the next cause
    t = t.getCause();
  }
}
```

JMS 2.0

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import javax.jms.JMSException;
.
.
.
catch (JMSException je) {
  System.err.println("Caught JMSException");
  // Check for linked exceptions in JMSException
  Throwable t = je;
  while (t != null) {
    // Write out the message that is applicable to all exceptions
    System.err.println("Exception Msg: " + t.getMessage());
    // Write out the exception stack trace
    t.printStackTrace(System.err);

    // Add on specific information depending on the type of exception
```

```
    if (t instanceof JMSException) {
      JMSException je1 = (JMSException) t;
      System.err.println("JMS Error code: " + je1.getErrorCode());
      if (t instanceof JmsExceptionDetail){
        JmsExceptionDetail jed = (JmsExceptionDetail)je1;
        System.err.println("JMS Explanation: " + jed.getExplanation());
        System.err.println("JMS Explanation: " + jed.getUserAction());
      }
    } else if (t instanceof MQException) {
      MQException mqe = (MQException) t;
      System.err.println("WMQ Completion code: " + mqe.getCompCode());
      System.err.println("WMQ Reason code: " + mqe.getReason());
    } else if (t instanceof JmqiException){
      JmqiException jmqie = (JmqiException)t;
      System.err.println("WMQ Log Message: " + jmqie.getWmqLogMessage());
      System.err.println("WMQ Explanation: " + jmqie.getWmqMsgExplanation());
      System.err.println("WMQ Msg Summary: " + jmqie.getWmqMsgSummary());
      System.err.println("WMQ Msg User Response: " + jmqie.getWmqMsgUserResponse());
      System.err.println("WMQ Msg Severity: " + jmqie.getWmqMsgSeverity());
    }
    // Get the next cause
    t = t.getCause();
  }
}
```

Kural dışı durum tipi değişebileceği ve farklı tiplerdeki kural dışı durumlar farklı bilgileri kapsülleyebileceği için, uygulamanızın bir zincirdeki her kural dışı durumun tipini her zaman denetlemesi gerektiğini unutmayın.

Bir sorunla ilgili belirli bilgilerin IBM MQ alınması

`com.ibm.mq.MQException` ve `com.ibm.mq.jmqi.JmqiException` örnekleri IBM MQ bir sorunla ilgili belirli bilgileri içerir.

`MQException` , aşağıdaki bilgileri içerir:

- Uygulamanızın `getCompCode()` yöntemini çağırarak elde edebileceği bir tamamlanma kodu.
- Uygulamanızın `getReason()` yöntemini çağırarak elde edebileceği bir neden kodu.

Bu yöntemlerin nasıl kullanılacağına ilişkin örnekler için, bağlantılı kural dışı durumlariçindeki örnek koda bakın.

`JmqiException` , bir tamamlanma kodunu ve neden kodunu da kapsüller. Buna ek olarak, `JmqiException` , kural dışı durumla ilişkiliyse, AMQ *nnnn* ya da CSQ *nnnn* iletisindeki bilgileri içerir. Uygulamanız, aşağıdaki yöntemleri çağırarak bu iletinin çeşitli bileşenlerini alabilir:

- `getWmqMsgExplanation()` yöntemi, AMQ *nnnn* ya da CSQ *nnnn* iletisinin açıklamasını döndürür.
- `getWmqMsgSeverity()` yöntemi, AMQ *nnnn* ya da CSQ *nnnn* iletisinin önem düzeyini döndürür.
- `getWmqMsgSummary()` yöntemi, AMQ *nnnn* ya da CSQ *nnnn* iletisinin özetini döndürür.
- `getWmqMsgUserResponse()` yöntemi, AMQ *nnnn* ya da CSQ *nnnn* iletisiyle ilişkili kullanıcı yanıtını döndürür.

Denetimsiz kural dışı durumların işlenmesi
JMS 2.0 ' da tanımlanan arabirimlerdeki yöntemler, denetimsiz kural dışı durumlar yayınlıyor. Bu kural dışı durumların temel sınıfı şudur: `JMSRuntimeException`. Bu nedenle, `JMSRuntimeExceptions` ' in yakalanması bu kural dışı durum tiplerini işlemek için soysal bir yol sağlar.

Her `JMSRuntimeException` aşağıdaki bilgileri içerir:

- Uygulamanızın `JMSRuntimeException.getMessage()` yöntemini çağırarak elde edebileceği, sağlayıcıya özgü bir kural dışı durum iletisi.
- Uygulamanızın `JMSRuntimeException.getErrorCode()` yöntemini çağırarak elde edebileceği, sağlayıcıya özgü bir hata kodu.
- Bağlantılı bir kural dışı durum. Bir JMS 2.0 API çağrısı tarafından yayınlanan bir kural dışı durum, genellikle bu kural dışı durumla bağlantılı başka bir kural dışı durum tarafından bildirilen daha

düşük düzeyli bir sorunun sonucudur. Uygulamanız, `JMSRuntimeException.getCause()` yöntemini çağırarak bağlantılı bir kural dışı durum alabilir.

JMS 2.0 API tarafından sağlanan arabirimlerde yöntemleri çağırdığınızda, IBM MQ classes for JMS tarafından yayınlanan kural dışı durumların çoğu `JMSRuntimeException`alt sınıflarının eşgörünümleridir. Bu alt sınıflar, aşağıdaki ek bilgileri sağlayan `com.ibm.msg.client.jms.JmsExceptionDetail` arabirimini gerçekleştirir:

- Kural dışı durum iletisinin açıklaması. Uygulamanız, `JmsExceptionDetail.getExplanation()` yöntemini çağırarak bu iletiyi alabilir.
- Kural dışı duruma önerilen bir kullanıcı yanıtı. Uygulamanız, `JmsExceptionDetail.getUserAction()` yöntemini çağırarak bu iletiyi alabilir.
- İletiye ilişkin anahtarlar kural dışı durum iletisine eklenir. Uygulamanız, `JmsExceptionDetail.getKeys()` yöntemini çağırarak tüm anahtarlar için bir yineleyici elde edebilir.
- İleti kural dışı durum iletisine eklenir. Örneğin, ileti ekleme işlemi kural dışı duruma neden olan kuyruğun adı olabilir ve uygulamanızın bu ada erişmesi yararlı olabilir. Uygulamanız, `JmsExceptionDetail.getValue()` yöntemini çağırarak, belirtilen bir anahtara karşılık gelen ileti eklemesini alabilir.

Ayrıntı yoksa, `JmsExceptionDetail` arabirimindeki tüm yöntemler boş değer döndürür.

Örneğin, bir uygulama var olmayan bir IBM MQ kuyruğu için `JMSProducer` yaratmayı denerse, aşağıdaki bilgilerle birlikte bir kural dışı durum yayınlanır:

```
Message : JMSWMQ2008: Failed to open MQ queue 'Q_test'.
Class : class com.ibm.msg.client.jms.DetailedInvalidDestinationException
Error Code : JMSWMQ2008
Explanation : JMS attempted to perform an MQOPEN, but IBM MQ reported an
              error.
User Action : Use the linked exception to determine the cause of this error. Check
              that the specified queue and queue manager are defined correctly.
```

Yayınlanan kural dışı durum (`com.ibm.msg.client.jms.DetailedInvalidDestinationException`), aşağıdaki sınıfın bir alt sınıfıdır ve `com.ibm.msg.client.jms.JmsExceptionDetail` arabirimini gerçekleştirir.

- V 9.3.0 V 9.3.0 JM 3.0 `jakarta.jms.InvalidDestinationException`
- JMS 2.0 `javax.jms.InvalidDestinationException`

Zincirleme kural dışı durumlar

Genellikle, kural dışı durumlar diğer kural dışı durumlardan kaynaklanır. Bu nedenle, yayınlanan her `JMSRuntimeException` için uygulamanızın bağlantılı kural dışı durumu denetlemesi gerekir.

`JMSRuntimeException` ' in nedeni başka bir kural dışı durum olabilir. Bu istisnalar, orijinal temel soruna yol açan bir zincir oluşturur. Bir kural dışı durumun nedeni, `java.lang.Throwable` sınıfının zincirleme kural dışı durum düzeneği kullanılarak gerçekleştirilir ve uygulamanız `Throwable.getCause()` yöntemini çağırarak bağlantılı bir kural dışı durum alabilir.

Örneğin, bir uygulama bir kuyruk yöneticisine bağlanırken yanlış bir kapı numarası belirtirse, kural dışı durumlar aşağıdaki zinciri oluşturur:

```
com.ibm.msg.client.jms.DetailIllegalStateException
|
+--->
    com.ibm.mq.MQException
    |
    +--->
        com.ibm.mq.jmqi.JmqiException
        |
        +--->
            com.ibm.mq.jmqi.JmqiException
            |
```

```
            +--->
                java.net.ConnectionException
```

Genellikle, bir zincirdeki her kural dışı durum, koddaki farklı bir katmandan yayınlanır. Örneğin, önceki zincirdeki kural dışı durumlar aşağıdaki katmanlar tarafından yayınlanır:

- `JMSRuntimeException`alt sınıfının bir eşgörünümü olan ilk kural dışı durum, IBM MQ classes for JMSiçindeki ortak katman tarafından yayınlanır.
- Sonraki kural dışı durum (bir `com.ibm.mq.MQException`örneği), IBM MQ ileti alışverişi sağlayıcısı tarafından yayınlanır.
- Her ikisi de `com.ibm.mq.jmqi.JmqiException`eşgörünümü olan sonraki iki kural dışı durum, Java İleti Kuyruğa Alma Arabirimi (JMQI) tarafından yayınlanır. JMQI, IBM MQ classes for JMS tarafından bir kuyruk yöneticisiyle iletişim kurmak için kullanılan bileşendir.
- Son kural dışı durum (bir `java.net.ConnectionException`eşgörünümü), Java sınıf kitaplığı tarafından yayınlanır.

IBM MQ classes for JMSkatmanlı mimarisi hakkında daha fazla bilgi için bkz. IBM MQ JMS mimarisi sınıfları.

Aşağıdaki örnekte gösterildiği gibi, uygulamanızı, uygun tüm bilgileri çıkarmak için bu zincir boyunca yinelenecek şekilde kodlayabilirsiniz:

V 9.3.0 V 9.3.0 JM 3.0

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import jakarta.jms.JMSRuntimeException;
.
.
.
catch (JMSRuntimeException je) {
  System.err.println("Caught JMSRuntimeException");
  // Check for linked exceptions in JMSRuntimeException
  Throwable t = je;
  while (t != null) {
    // Write out the message that is applicable to all exceptions
    System.err.println("Exception Msg: " + t.getMessage());
    // Write out the exception stack trace
    t.printStackTrace(System.err);

    // Add on specific information depending on the type of exception
    if (t instanceof JMSRuntimeException) {
      JMSRuntimeException je1 = (JMSRuntimeException) t;
      System.err.println("JMS Error code: " + je1.getErrorCode());
      if (t instanceof JmsExceptionDetail){
        JmsExceptionDetail jed = (JmsExceptionDetail)je1;
        System.err.println("JMS Explanation: " + jed.getExplanation());
        System.err.println("JMS Explanation: " + jed.getUserAction());
      }
    } else if (t instanceof MQException) {
      MQException mqe = (MQException) t;
      System.err.println("WMQ Completion code: " + mqe.getCompCode());
      System.err.println("WMQ Reason code: " + mqe.getReason());
    } else if (t instanceof JmqiException){
      JmqiException jmqie = (JmqiException)t;
      System.err.println("WMQ Log Message: " + jmqie.getWmqLogMessage());
      System.err.println("WMQ Explanation: " + jmqie.getWmqMsgExplanation());
      System.err.println("WMQ Msg Summary: " + jmqie.getWmqMsgSummary());
      System.err.println("WMQ Msg User Response: " + jmqie.getWmqMsgUserResponse());
      System.err.println("WMQ Msg Severity: " + jmqie.getWmqMsgSeverity());
    }
    // Get the next cause
    t = t.getCause();
  }
}
```

JMS 2.0

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import javax.jms.JMSRuntimeException;
.
```

```
.
.
catch (JMSRuntimeException je) {
  System.err.println("Caught JMSRuntimeException");
  // Check for linked exceptions in JMSRuntimeException
  Throwable t = je;
  while (t != null) {
    // Write out the message that is applicable to all exceptions
    System.err.println("Exception Msg: " + t.getMessage());
    // Write out the exception stack trace
    t.printStackTrace(System.err);

    // Add on specific information depending on the type of exception
    if (t instanceof JMSRuntimeException) {
      JMSRuntimeException je1 = (JMSRuntimeException) t;
      System.err.println("JMS Error code: " + je1.getErrorCode());
      if (t instanceof JmsExceptionDetail){
        JmsExceptionDetail jed = (JmsExceptionDetail)je1;
        System.err.println("JMS Explanation: " + jed.getExplanation());
        System.err.println("JMS Explanation: " + jed.getUserAction());
      }
    } else if (t instanceof MQException) {
      MQException mqe = (MQException) t;
      System.err.println("WMQ Completion code: " + mqe.getCompCode());
      System.err.println("WMQ Reason code: " + mqe.getReason());
    } else if (t instanceof JmqiException){
      JmqiException jmqie = (JmqiException)t;
      System.err.println("WMQ Log Message: " + jmqie.getWmqLogMessage());
      System.err.println("WMQ Explanation: " + jmqie.getWmqMsgExplanation());
      System.err.println("WMQ Msg Summary: " + jmqie.getWmqMsgSummary());
      System.err.println("WMQ Msg User Response: " + jmqie.getWmqMsgUserResponse());
      System.err.println("WMQ Msg Severity: " + jmqie.getWmqMsgSeverity());
    }
    // Get the next cause
    t = t.getCause();
  }
}
```

Kural dışı durum tipi değişebileceği ve farklı tiplerdeki kural dışı durumlar farklı bilgileri kapsülleyebileceği için, uygulamanızın bir zincirdeki her kural dışı durumun tipini her zaman denetlemesi gerektiğini unutmayın.

Bir sorunla ilgili belirli bilgilerin IBM MQ alınması

`com.ibm.mq.MQException` ve `com.ibm.mq.jmqi.JmqiException` örnekleri IBM MQ bir sorunla ilgili belirli bilgileri içerir.

`MQException` , aşağıdaki bilgileri içerir:

- Uygulamanızın `getCompCode()` yöntemini çağırarak elde edebileceği bir tamamlanma kodu.
- Uygulamanızın `getReason()` yöntemini çağırarak elde edebileceği bir neden kodu.

Bu yöntemlerin nasıl kullanılacağına ilişkin örnekler için, zincirleme kural dışı durumlariçindeki örnek koda bakın.

`JmqiException` , bir tamamlanma kodunu ve neden kodunu da kapsüller. Buna ek olarak, `JmqiException` , kural dışı durumla ilişkiliyse, AMQ *nnnn* ya da CSQ *nnnn* iletisindeki bilgileri içerir. Uygulamanız, aşağıdaki yöntemleri çağırarak bu iletinin çeşitli bileşenlerini alabilir:

- `getWmqMsgExplanation()` yöntemi, AMQ *nnnn* ya da CSQ *nnnn* iletisinin açıklamasını döndürür.
- `getWmqMsgSeverity()` yöntemi, AMQ *nnnn* ya da CSQ *nnnn* iletisinin önem düzeyini döndürür.
- `getWmqMsgSummary()` yöntemi, AMQ *nnnn* ya da CSQ *nnnn* iletisinin özetini döndürür.
- `getWmqMsgUserResponse()` yöntemi, AMQ *nnnn* ya da CSQ *nnnn* iletisiyle ilişkili kullanıcı yanıtını döndürür.

ExceptionListeners
JMS `Connection` ve `JMSContext` nesnelerinin bir kuyruk yöneticisiyle ilişkili bağlantısı vardır. Uygulamanız bir `ExceptionListener` 'i bir JMS `Connection` ya da `JMSContext`ile kaydedebilir. `Connection` ya da `JMSContext` ile ilişkili bağlantıyı kullanılamaz kılan bir sorun ortaya çıkarsa, IBM

MQ classes for JMS , onException() yöntemini çağırarak ExceptionListener ' e bir kural dışı durum gönderir. Böylece uygulamanız bağlantıyı yeniden kurma fırsatına sahip olur.

IBM MQ classes for JMS , bir iletiyi zamanuyumsuz olarak teslim etme girişimi sırasında bir sorun oluşursa kural dışı durum dinleyicisine bir kural dışı durum da sağlayabilir.

Kural dışı durum dinleyicileri

IBM MQ 8.0.0 Fix Pack 2'dan, bir JMS MessageListener ve bir JMS ExceptionListeneryapılandıran geçerli JMS uygulamalarının davranışını korumak ve IBM MQ classes for JMS ' in JMS belirtimiyle tutarlı olduğundan emin olmak için ConnectionFactory özelliği ASYNCEXCEPTION için varsayılan değer ASYNC_EXCEPTIONS_CONNECTIONBROKEN olarak değiştirilir. Sonuç olarak, bir uygulamanın ExceptionListener ' e yalnızca bozuk bağlantı hata kodlarına karşılık gelen kural dışı durumlar teslim edilir.

APAR IT14820, IBM MQ 9.0.0 Fix Pack 1'in içerdiği, IBM MQ classes for JMS ' yi aşağıdaki şekilde günceller:

- Bir uygulama tarafından kaydedilen bir ExceptionListener , uygulamanın zamanuyumlu ya da zamanuyumsuz ileti tüketicileri kullanıp kullanmadığından bağımsız olarak, bağlantı kesilmiş kural dışı durumlar için çağrılır.
- Uygulama zamanuyumsuz ileti tüketicileri kullandığında ve uygulama tarafından kullanılan JMS ConnectionFactory ' de ASYNC_EXCEPTION özelliği ASYNC_EXCEPTIONS_ALL değerine ayarlandığında, ileti teslimi sırasında oluşan bağlantı dışı kural dışı durumlar (örneğin, MQRC_GET_INENGELLENDI) bir uygulamanın ExceptionListener öğesine teslim edilir.

Not: İki TCP/IP bağlantısı (biri JMS Connection tarafından kullanılan, diğeri JMS Oturumu tarafından kullanılan) bozuk olsa bile, bağlantı kesilmiş bir kural dışı durum için ExceptionListener yalnızca bir kez çağrılır.

Başka bir sorun tipi için, geçerli JMS API çağrısı tarafından bir kural dışı durum yayınlanır. Yayınlanan kural dışı durum tipi, uygulamanın kullandığı JMS API sürümüne bağlıdır:

- Uygulama, JMS 1.1 belirtimi tarafından sağlanan arabirimleri kullanıyorsa, kural dışı durum bir JMSExceptionolur. Bu kural dışı durumların nasıl işleneceğine ilişkin daha fazla bilgi için bkz. “Denetlenen kural dışı durumların işlenmesi” sayfa 228.
- Uygulama JMS 2.0 arabirimlerini kullanıyorsa, kural dışı durum JMSRuntimeExceptionolur. Bu kural dışı durumların nasıl işleneceğine ilişkin daha fazla bilgi için bkz. “Denetimsiz kural dışı durumların işlenmesi” sayfa 231.

Bir uygulama bir kural dışı durum dinleyicisini Connection ya da JMSContextile kaydettirmezse, kural dışı durum dinleyicisine teslim edilecek kural dışı durumlar IBM MQ classes for JMS günlüğüne yazılır.

Bir IBM MQ classes for JMS uygulamasından IBM MQ özelliklerine erişme

IBM MQ classes for JMS , çeşitli IBM MQözelliklerinden yararlanmak için olanaklar sağlar.

Uyarı: Bu özellikler JMS belirtiminin dışındadır ya da bazı durumlarda JMS belirtimini ihlal etmektedir. Bunları kullanırsanız, uygulamanızın diğer JMS sağlayıcılarıyla uyumlu olma olasılığı düşüktür. JMS belirtimiyle uyumlu olmayan aksamlar bir Uyarı notuyla etiketlenir.

IBM MQ classes for JMS uygulamasından ileti tanımlayıcısının okunması ve yazılması

Hedef ve İleti üzerinde özellikler ayarlayarak ileti tanımlayıcısına (MQMD) erişme yeteneğini denetleyebilirsiniz.

Bazı IBM MQ uygulamaları, kendilerine gönderilen iletilerin MQMD ' de belirli değerlerin ayarlanmasını gerektirir. IBM MQ classes for JMS , JMS uygulamalarının MQMD alanlarını ayarlamasına ve bu nedenle JMS uygulamalarının "sürücü" IBM MQ uygulamaları için etkinleştirilmesine olanak sağlayan ileti özniteliklerini sağlar.

MQMD özelliklerinin ayarının etkili olması için WMQ_MQMD_WRITE_ENABLED Hedef nesne özelliğini true değerine ayarlamalısınız. Daha sonra, MQMD alanlarına değer atamak için iletinin özellik ayarı

yöntemlerini (örneğin, setStringÖzelliği) kullanabilirsiniz. StrucId ve Version dışında tüm MQMD alanları gösterilir; BackoutCount okunabilir, ancak yazılamaz.

Bu örnek, bir iletinin MQMD.UserIdentifier , "JoeBloggs" olarak ayarlandı.

```
// 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
// ...
```

JMS_IBM_MQMD_UserIdentifierayarlanmadan önce WMQ_MQMD_MESSAGE_CONTEXT ayarlanmalıdır. WMQ_MQMD_MESSAGE_CONTEXT kullanımı hakkında daha fazla bilgi için bkz. “JMS ileti nesnesi özellikleri” sayfa 238.

Benzer şekilde, bir ileti almadan önce WMQ_MQMD_READ_ENABLED öğesini true değerine ayarlayarak ve daha sonra, getStringözelliği gibi iletinin alma yöntemlerini kullanarak MQMD alanlarının içeriğini çıkarabilirsiniz. Alınan özellikler salt okunur özelliklerdir.

Bu örnek, bir kuyruktan ya da konudan alınan bir iletinin MQMD.ApplIdentityData alanının değerini tutan *değer* alanıyla sonuçlanır.

```
// 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");
```

JMS hedef nesne özellikleri
Hedef nesnenin iki özelliği JMSiçinden MQMD ' ye erişimi denetler ve üçüncü bir özellik ileti bağlamını denetler.

Çizelge 36. Özellik adları ve açıklamaları

Özellik	**Kısa Biçim**	**Açıklama**
WMQ_MQMD_WRITE_ENABLED	MDW	Bir JMS uygulamasının MQMD alanlarının değerlerini ayarlayıp ayarlayamayacağını belirler
WMQ_MQMD_READ_ENABLED	MDR	Bir JMS uygulamasının MQMD alanlarının değerlerini alıp alamayacağını belirler
WMQ_MQMD_MESSAGE_ BAğLAMı	MDCTX	JMS uygulaması tarafından ayarlanacak ileti bağlamı düzeyi. Bu özelliğin yürürlüğe girmesi için uygulamanın uygun bağlam yetkisiyle çalışması gerekir

Çizelge 37. Özellik adları, değerler ve ayar yöntemleri

Özellik	**Yönetim aracında geçerli değerler (varsayılan değerler koyu)**	**Programlardaki geçerli değerler**	**Yöntemi ayarla**
WMQ_MQMD_WRITE_ENABLED	• **HAYIR** Tüm JMS_IBM_MQMD* özellikleri yoksayılır ve değerleri temel MQMD yapısına kopyalanmaz. • EVET JMS_IBM_MQMD* özellikleri işlendi. Değerleri temeldeki MQMD yapısına kopyalanır.	• **Yanlış** • Doğru	setMQMDWriteEtkin
WMQ_MQMD_READ_ENABLED	• **HAYIR** İleti gönderilirken, gönderilen bir iletideki JMS_IBM_MQMD* özellikleri MQMD ' deki güncellenen alan değerlerini yansıtacak şekilde güncellenmez. İleti alınırken, JMS_IBM_MQMD* özelliklerinin hiçbiri, gönderen bunların bazılarını ya da tümünü ayarlamış olsa da, alınan bir iletide kullanılamaz. • EVET İleti gönderilirken, gönderilen bir iletideki tüm JMS_IBM_MQMD* özellikleri, gönderenin belirtik olarak ayarlı olmayanlar da içinde olmak üzere, MQMD ' deki güncellenen alan değerlerini yansıtacak şekilde güncellenir. İleti alınırken, gönderenin belirtik olarak ayarlamadığı iletiler de içinde olmak üzere, alınan bir iletide tüm JMS_IBM_MQMD* özellikleri kullanılabilir.	• **Yanlış** • Doğru	setMQMDReadEtkin

Çizelge 37. Özellik adları, değerler ve ayar yöntemleri (devamı var)			
Özellik	**Yönetim aracında geçerli değerler (varsayılan değerler koyu)**	**Programlardaki geçerli değerler**	**Yöntemi ayarla**
WMQ_MQMD _MESSAGE_CONTEXT	• **VARSAYILAN** MQOPEN API çağrısı ve MQPMO yapısı belirtik ileti bağlamı seçenekleri belirtmiyor • SET_IDENTITY_CONTEXT MQOPEN API çağrısı MQOO_SET_IDENTITY_CONTEXT ileti bağlamı seçeneğini belirtiyor ve MQPMO yapısı MQPMO_SET_IDENTITY_CONTEXT öğesini belirtiyor • SET_ALL_CONTEXT MQOPEN API çağrısı MQOO_SET_ALL_CONTEXT ileti bağlamı seçeneğini belirtiyor ve MQPMO yapısı MQPMO_SET_ALL_CONTEXT öğesini belirtiyor	• **WMQ_MDCTX_DEFAULT** • WMQ_MDCTX_SET_IDENTITY_CONTEXT • WMQ_MDCTX_SET_ALL_CONTEXT	setMQMDMessageİçeriği

JMS ileti nesnesi özellikleri
İleti nesnesi özellikleri önekli JMS_IBM_MQMD, ilgili MQMD alanını ayarlamanıza ya da okumanızı sağlar.

İleti gönderilmesi

StrucId ve Version dışındaki tüm MQMD alanları gösterilir. Bu özellikler yalnızca MQMD alanlarını belirtir; bir özellik hem MQMD 'de hem de MQRFH2 üstbilgisinde gerçekleştiğinde, MQRFH2 ' deki sürüm ayarlanmaz ya da alınmaz.

JMS_IBM_MQMD_BackoutCountdışında, bu özelliklerden herhangi biri ayarlanabilir. JMS_IBM_MQMD_BackoutCount için ayarlanan herhangi bir değer yoksayılır.

Bir özelliğin uzunluk üst sınırı varsa ve çok uzun bir değer sağlarsanız, değer kesilir.

Belirli özellikler için, Hedef nesnede WMQ_MQMD_MESSAGE_CONTEXT özelliğini de ayarlamanız gerekir. Bu özelliğin yürürlüğe girmesi için uygulamanın uygun bağlam yetkisiyle çalışması gerekir. WMQ_MQMD_MESSAGE_CONTEXT değerini uygun bir değere ayarlamazsanız, özellik değeri yoksayılır. WMQ_MQMD_MESSAGE_CONTEXT değerini uygun bir değere ayarlarsanız, ancak kuyruk yöneticisi için yeterli bağlam yetkiniz yoksa, bir JMSException yayınlanır. WMQ_MQMD_MESSAGE_CONTEXT 'in belirli değerlerini gerektiren özellikler şunlardır.

Aşağıdaki özellikler WMQ_MQMD_MESSAGE_CONTEXT 'in WMQ_MDCTX_SET_IDENTITY_CONTEXT ya da WMQ_MDCTX_SET_ALL_CONTEXT olarak ayarlanmasını gerektirir:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- JMS_IBM_MQMD_ApplIdentityVerileri

Aşağıdaki özellikler, WMQ_MQMD_MESSAGE_CONTEXT 'in WMQ_MDCTX_SET_ALL_CONTEXT olarak ayarlanmasını gerektirir:

- JMS_IBM_MQMD_PutApplTipi
- JMS_IBM_MQMD_PutApplAdı

- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- JMS_IBM_MQMD_ApplOriginVerileri

İleti alınması

WMQ_MQMD_READ_ENABLED özelliği, üreten uygulamanın ayarladığı gerçek özelliklerden bağımsız olarak true değerine ayarlanırsa, alınan bir iletide tüm bu özellikler kullanılabilir. JMS belirtimine göre, önce tüm özellikler temizlenmedikçe, bir uygulama alınan bir iletinin özelliklerini değiştiremez. Alınan ileti, özellikler değiştirilmeden iletilebilir.

Uyarı: Uygulamanız WMQ_MQMD_READ_ENABLED özelliği true olarak ayarlanmış bir hedeften bir ileti alırsa ve bunu WMQ_MQMD_WRITE_ENABLED değeri true olarak ayarlanmış bir hedefe iletiyorsa, bu, alınan iletinin tüm MQMD alan değerlerinin iletilen iletiye kopyalanmasıyla sonuçlanır.

Özellikler tablosu

Bu çizelge, MQMD alanlarını gösteren İleti nesnesinin özelliklerini listeler. Alanların ve bunların izin verilen değerlerinin tam açıklamaları için bağlantılara bakın.

Çizelge 38. Özellik adları, tanımlar ve tipler

Özellik	Açıklama	Java Tip	Tam açıklamaya bağla
JMS_IBM_MQMD_Raporu	Rapor iletileri için seçenekler	Tamsayı	Rapor
JMS_IBM_MQMD_MsgType	İleti tipi	Tamsayı	MsgType
JMS_IBM_MQMD_Expiry	İleti geçerlik süresi	Tamsayı	Son kullanma tarihi
JMS_IBM_MQMD_Feedback	Geribildirim ya da neden kodu	Tamsayı	Geri bildirim
JMS_IBM_MQMD_Kodlaması	İleti verilerinin sayısal kodlaması	Tamsayı	Kodlama
JMS_IBM_MQMD_CodedCharSetId	İleti verilerinin karakter kümesi tanıtıcısı	Tamsayı	CodedCharSetId
JMS_IBM_MQMD_Format	İleti verilerinin biçim adı	Dizgi	Biçim
JMS_IBM_MQMD_Priority [1]	İleti önceliği	Tamsayı	Öncelik
JMS_IBM_MQMD_Persistence	İleti kalıcılığı	Tamsayı	Kalıcılık
JMS_IBM_MQMD_MsgId [2]	İleti Tanıtıcısı	Nesne (bayt []) [4]	MsgId
JMS_IBM_MQMD_CorrelId [3]	İlinti tanıtıcısı	Nesne (bayt []) [4]	CorrelId
JMS_IBM_MQMD_BackoutCount	Geriletme sayacı	Tamsayı	BackoutCount
JMS_IBM_MQMD_ReplyToQ	Yanıt kuyruğunun adı	Dizgi	ReplyToQ
JMS_IBM_MQMD_ReplyToQMgr	Yanıt kuyruğu yöneticisinin adı	Dizgi	ReplyToQMgr
JMS_IBM_MQMD_UserIdentifier	Kullanıcı kimliği	Dizgi	UserIdentifier
JMS_IBM_MQMD_AccountingToken	Muhasebe simgesi	Nesne (bayt []) [4]	AccountingToken

Çizelge 38. Özellik adları, tanımlar ve tipler (devamı var)			
Özellik	**Açıklama**	**Java Tip**	**Tam açıklamaya bağla**
JMS_IBM_MQMD_ApplIdentityVeri leri	Kimlikle ilgili uygulama verileri	Dizgi	ApplIdentityVerileri
JMS_IBM_MQMD_PutApplTipi	İletiyi koyan uygulamanın tipi	Tamsayı	PutApplTip
JMS_IBM_MQMD_PutApplAdı	İletiyi koyan uygulamanın adı	Dizgi	PutApplAdı
JMS_IBM_MQMD_PutDate	İletinin konma tarihi	Dizgi	PutDate
JMS_IBM_MQMD_PutTime	İletinin konma zamanı	Dizgi	PutTime
JMS_IBM_MQMD_ApplOriginVerile ri	Kaynak ile ilgili uygulama verileri	Dizgi	ApplOriginVerileri
JMS_IBM_MQMD_GroupId	Grup tanıtıcısı	Nesne (bayt []) [4]	GroupId
JMS_IBM_MQMD_MsgSeqNumara sı	Grup içindeki mantıksal iletinin sıra numarası	Tamsayı	MsgSeqSayı
JMS_IBM_MQMD_Göreli Konumu	Mantıksal iletinin başlangıcından fiziksel iletideki verilerin göreli konumu	Tamsayı	Göreli Konum
JMS_IBM_MQMD_MsgFlags	İleti İşaretleri	Tamsayı	MsgFlags
JMS_IBM_MQMD_OriginalLength	Özgün iletinin uzunluğu	Tamsayı	OriginalLength

1. **Uyarı:** JMS_IBM_MQMD_Priority için 0-9 aralığında olmayan bir değer atarsanız, bu JMS belirtimini ihlal eder.
2. **Uyarı:** JMS belirtimi, ileti tanıtıcısının JMS sağlayıcısı tarafından ayarlanması gerektiğini ve benzersiz ya da boş olması gerektiğini belirtiyor. JMS_IBM_MQMD_MsgId'ye bir değer atarsanız, bu değer JMSMessageID' ye kopyalanır. Bu nedenle, JMS sağlayıcısı tarafından ayarlanmaz ve benzersiz olmayabilir: Bu, JMS belirtimini ihlal eder.
3. **Uyarı:** JMS_IBM_MQMD_CorrelId öğesine 'ID:' dizgisiyle başlayan bir değer atarsanız, bu JMS belirtimini ihlal eder.
4. **Uyarı:** Bir iletide bayt dizisi özelliklerinin kullanılması JMS belirtimine aykırılık oluşturur.

IBM MQ İleti verilerine bir uygulamadan IBM MQ classes for JMS kullanarak erişme
IBM MQ classes for JMSkullanarak bir uygulama içindeki IBM MQ ileti verilerinin tamamına erişebilirsiniz. Tüm verilere erişmek için iletinin bir `JMSBytesMessage`olması gerekir. `JMSBytesMessage` gövdesi herhangi bir `MQRFH2` üstbilgisini, diğer IBM MQ üstbilgilerini ve aşağıdaki ileti verilerini içerir.

`JMSBytesMessage`içindeki tüm ileti gövdesi verilerini almak için, hedefin `WMQ_MESSAGE_BODY` özelliğini `WMQ_MESSAGE_BODY_MQ`olarak ayarlayın.

`WMQ_MESSAGE_BODY` `WMQ_MESSAGE_BODY_JMS` ya da `WMQ_MESSAGE_BODY_UNSPECIFIED`olarak ayarlanırsa, ileti gövdesi JMS `MQRFH2` üstbilgisi olmadan döndürülür ve `JMSBytesMessage` özellikleri `RFH2`içinde ayarlanan özellikleri yansıtır.

Bazı uygulamalar bu konuda açıklanan işlevleri kullanamaz. Bir uygulama IBM MQ V6 kuyruk yöneticisine bağlıysa ya da `PROVIDERVERSION` değerini `6`olarak ayarladıysa, işlevler kullanılamaz.

İleti gönderilmesi

İletiler hedef özelliğe gönderilirken WMQ_MESSAGE_BODY, WMQ_TARGET_CLIENT' a göre önceliklidir.

WMQ_MESSAGE_BODY WMQ_MESSAGE_BODY_JMSolarak ayarlanırsa, IBM MQ classes for JMS otomatik olarak JMSMessage özelliklerinin ve üstbilgi alanlarının ayarlarına dayalı olarak bir MQRFH2 üstbilgisi oluşturur.

WMQ_MESSAGE_BODY WMQ_MESSAGE_BODY_MQolarak ayarlanırsa, ileti gövdesine ek üstbilgi eklenmez

WMQ_MESSAGE_BODY WMQ_MESSAGE_BODY_UNSPECIFIEDolarak ayarlanırsa, WMQ_TARGET_CLIENT WMQ_TARGET_DEST_MQolarak ayarlanmadıkça IBM MQ classes for JMS bir MQRFH2 üstbilgisi gönderir. Alma sırasında, WMQ_TARGET_CLIENT WMQ_TARGET_DEST_MQ olarak ayarlandığında, MQRFH2 ileti gövdesinden kaldırılır.

Not: JMSBytesMessage ve JMSTextMessage , bir MQRFH2gerektirmezken, JMSStreamMessage, JMSMapMessageve JMSObjectMessage gerektirir.

WMQ_MESSAGE_BODY_UNSPECIFIED , WMQ_MESSAGE_BODYiçin varsayılan ayardır ve WMQ_TARGET_DEST_JMS , WMQ_TARGET_CLIENTiçin varsayılan ayardır.

JMSBytesMessagegönderirseniz, IBM MQ iletisi oluşturulduğunda JMS ileti gövdesine ilişkin varsayılan ayarları geçersiz kılabilirsiniz. Aşağıdaki özellikleri kullanın:

- JMS_IBM_Format ya da JMS_IBM_MQMD_Format: Bu özellik, daha önce WebSphere MQ üstbilgisi yoksa, JMS ileti gövdesini başlatan IBM MQ üstbilgisinin ya da uygulama bilgi yükünün biçimini belirtir.
- JMS_IBM_Character_Set ya da JMS_IBM_MQMD_CodedCharSetId: Bu özellik, daha önce WebSphere MQ üstbilgisi yoksa, JMS ileti gövdesini başlatan IBM MQ üstbilgisine ya da uygulama bilgi yüküne ilişkin CCSID değerini belirtir.
- JMS_IBM_Encoding ya da JMS_IBM_MQMD_Encoding: Bu özellik, daha önce WebSphere MQ üstbilgisi yoksa, JMS ileti gövdesini başlatan IBM MQ üstbilgisinin ya da uygulama bilgi yükünün kodlamasını belirtir.

Her iki özellik tipi de belirtilirse, WMQ_MQMD_WRITE_ENABLED hedef özelliği trueolarak ayarlandığı sürece JMS_IBM_MQMD_* özellikleri ilgili JMS_IBM_* özelliklerini geçersiz kılar.

JMS_IBM_MQMD_* ve JMS_IBM_* kullanılarak ileti özelliklerinin ayarlanması arasındaki farklar önemlidir:

1. JMS_IBM_MQMD_* özellikleri, IBM MQ JMS sağlayıcısına özgüdür.
2. JMS_IBM_MQMD_* özellikleri yalnızca MQMDiçinde ayarlanır. JMS_IBM_* özellikleri MQMD içinde yalnızca iletinin bir MQRFH2 JMS üstbilgisi yoksa ayarlanır. Tersi durumda, bunlar JMS RFH2 üstbilgisinde ayarlanır.
3. JMS_IBM_MQMD_* özellikleri, bir JMSMessageiçine yazılan metin ve sayıların kodlamasını etkilemez.

   Alan bir uygulama, MQMD.Encoding ve MQMD.CodedCharSetId değerlerinin, ileti gövdesindeki sayı ve metin kodlamasının ve karakter kümesinin karşılığı olduğunu varsayabilir. JMS_IBM_MQMD_* özellikleri kullanılırsa, bunu yapmak gönderen uygulamanın sorumluluğundadır. İleti gövdesindeki sayı ve metnin kodlaması ve karakter kümesi, JMS_IBM_* özellikleri tarafından ayarlanır.

   Şekil 39 sayfa 242 içindeki kötü kodlanmış kod parçası, MQMD.CodedCharSetId 37 olarak ayarlanmış olarak 1208 karakter takımındaki bir iletiyi gönderir.

a. Yanlış kodlanmış ileti gönder

```
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. MQMD.CodedCharSetIddeğeriyle ayarlanan JMS_IBM_CHARACTER_SET değerine dayanarak ileti alınıyor:

```
TextMessage tmi = (TextMessage) cons.receive();
System.out.println("Message is \"" + tmi.getText() + "\"");
```

c. Sonuçta elde edilen çıktı:

```
Message is "éÈÊ'>...??>?"
```

Şekil 39. Tutarsız olarak kodlanmış MQMD ve ileti verileri

Şekil 40 sayfa 242 içindeki kod parçacıklarının herhangi biri, bir iletinin otomatik olarak oluşturulan MQRFH2 üstbilgisi eklenmeden uygulama bilgi yükünü içeren bir kuyruğa ya da konuya konmasıyla sonuçlanır.

1. WMQ_MESSAGE_BODY_MQayarı:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

2. WMQ_TARGET_DEST_MQayarı:

```
((MQDestination) destination).setMessageBodyStyle
                (WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);
((MQDestination) destination).
                setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
```

Şekil 40. MQ ileti gövdesiyle bir ileti gönderin.

İleti alınması

WMQ_MESSAGE_BODY WMQ_MESSAGE_BODY_JMSolarak ayarlanırsa, gelen JMS ileti tipi ve gövdesi, alınan WebSphere MQ iletisinin içeriğine göre belirlenir. İleti tipi ve gövdesi, MQRFH2 üstbilgisindeki ya da MQRFH2yoksa MQMDiçindeki alanlara göre belirlenir.

WMQ_MESSAGE_BODY WMQ_MESSAGE_BODY_MQolarak ayarlanırsa, gelen JMS ileti tipi JMSBytesMessageolur. JMS ileti gövdesi, temel MQGET API çağrısıyla döndürülen ileti verileridir. İleti gövdesinin uzunluğu, MQGET çağrısının döndürdüğü uzunluktur. İleti gövdesindeki verilerin karakter kümesi ve kodlaması, MQMD' in CodedCharSetId ve Encoding alanları tarafından belirlenir. İleti gövdesindeki verilerin biçimi, MQMD içindeki Biçim alanıyla belirlenir.

WMQ_MESSAGE_BODY WMQ_MESSAGE_BODY_UNSPECIFIEDolarak ayarlanırsa, varsayılan değer olan IBM MQ classes for JMS bunu WMQ_MESSAGE_BODY_JMSolarak ayarlar.

Bir JMSBytesMessagealdığınızda, aşağıdaki özelliklere başvurarak kodunu çözebilirsiniz:

- `JMS_IBM_Format` ya da `JMS_IBM_MQMD_Format`: Bu özellik, daha önce WebSphere MQ üstbilgisi yoksa, JMS ileti gövdesini başlatan IBM MQ üstbilgisinin ya da uygulama bilgi yükünün biçimini belirtir.
- `JMS_IBM_Character_Set` ya da `JMS_IBM_MQMD_CodedCharSetId`: Bu özellik, daha önce WebSphere MQ üstbilgisi yoksa, JMS ileti gövdesini başlatan IBM MQ üstbilgisine ya da uygulama bilgi yüküne ilişkin `CCSID` değerini belirtir.
- `JMS_IBM_Encoding` ya da `JMS_IBM_MQMD_Encoding`: Bu özellik, daha önce WebSphere MQ üstbilgisi yoksa, JMS ileti gövdesini başlatan IBM MQ üstbilgisinin ya da uygulama bilgi yükünün kodlamasını belirtir.

Aşağıdaki kod parçacığı, `JMSBytesMessage`olan bir alınan iletiyle sonuçlanır. Alınan iletinin içeriğinden ve alınan `MQMD`biçim alanından bağımsız olarak, ileti bir `JMSBytesMessage`iletisidir.

```
((MQDestination)destination).setMessageBodyStyle
                            (WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

WMQ_MESSAGE_BODY hedef özelliği
WMQ_MESSAGE_BODY, bir JMS uygulamasının ileti bilgi yükünün (JMS ileti gövdesinin bir parçası olarak) bir IBM MQ iletisinin MQRFH2 öğesini işleyip işlemediğini belirler.

Çizelge 39. Özellik adları ve açıklamaları

Özellik	**Kısa Biçim**	**Açıklama**
WMQ_MESSAGE_BODY	MBODY	Bir JMS uygulamasının, ileti bilgi yükünün (JMS ileti gövdesinin bir parçası olarak) bir IBM MQ iletisinin MQRFH2 öğesini işleyip işlemediğini belirler.

Çizelge 40. Özellik adları, değerler ve ayar yöntemleri			
Özellik	**Yönetim aracında geçerli değerler (varsayılan değerler koyu)**	**Programlardaki geçerli değerler**	**Yöntemi ayarla**
WMQ_MESSAGE_BODY	• **Belirtilmedi** Gönderirken IBM MQ classes for JMS , WMQ_TARGET_CLIENT değerine bağlı olarak bir MQRFH2 üstbilgisi oluşturur ya da içermez. Alma sırasında, JMSdeğeri olarak hareket eder. • JMS Gönderirken IBM MQ classes for JMS otomatik olarak bir MQRFH2 üstbilgisi oluşturur ve bunu IBM MQ iletisine ekler. IBM MQ classes for JMS alırken, JMS ileti özelliklerini MQRFH2 'deki değerlere göre ayarlayın (varsa); MQRFH2 ' yi JMS ileti gövdesinin bir parçası olarak sunmaz. • MQ Gönderirken IBM MQ classes for JMS , bir MQRFH2oluşturmaz. IBM MQ classes for JMS , alırken JMS ileti gövdesinin bir parçası olarak MQRFH2 ' yi sunar.	• **WMQ_MESSAGE_ BODY_BELIRTILMEDI** • WMQ_MESSAGE_BODY_JMS • WMQ_MESSAGE_BODY_MQ	setMessageBodySty le

JMS kalıcı iletiler

IBM MQ classes for JMS uygulamaları, bazı güvenilirlik pahasına JMS kalıcı iletilere daha iyi performans sağlamak için **NonPersistentMessageClass** kuyruk özniteliğini kullanabilir.

Bir IBM MQ kuyruğunun **NonPersistentMessageClass**adlı bir özniteliği vardır. Bu özniteliğin değeri, kuyruk yöneticisi yeniden başlatıldığında kuyruktaki kalıcı olmayan iletilerin atılıp atılmayacağını belirler.

Aşağıdaki değiştirgelerden biriyle IBM MQ Script (MQSC) DEFINE QLOCAL komutunu kullanarak bir yerel kuyruk için öznitelik ayarlayabilirsiniz:

NPMSıNıF (OLAğAN)
Kuyruk yöneticisi yeniden başlatıldığında, kuyruktaki kalıcı olmayan iletiler atılır. Bu varsayılan değerdir.

NPMSıNıF (YüKSEK)
Kuyruk yöneticisi susturulmuş ya da anında kapandıktan sonra yeniden başlatıldığında, kuyruktaki kalıcı olmayan iletiler atılmaz. Ancak, önleyici bir sona erdirme ya da bir hata sonrasında kalıcı olmayan iletiler atılabilir.

Bu konuda, IBM MQ classes for JMS uygulamalarının JMS kalıcı iletileri için daha iyi performans sağlamak üzere bu kuyruk özniteliğini nasıl kullanabileceği açıklanmaktadır.

Bir Kuyruk ya da Konu nesnesinin PERSISTENCE özelliğinin değeri HIGH olabilir. Bu değeri ayarlamak için IBM MQ JMS yönetim aracını kullanabilir ya da bir uygulama, parametre olarak WMQConstants.WMQ_PER_NPHIGH değerini aktararak Destination.setPersistence() yöntemini çağırabilir.

Bir uygulama PERSISTENCE özelliğinin HIGH değerine sahip olduğu bir hedefe JMS kalıcı ileti ya da JMS kalıcı olmayan bir ileti gönderirse ve temeldeki IBM MQ kuyruğu NPMCLASS (HIGH) olarak ayarlanırsa, ileti IBM MQ kalıcı olmayan bir ileti olarak kuyruğa konur. Hedefin PERSISTENCE özelliğinin değeri HIGH değilse ya da temeldeki kuyruk NPMCLASS (NORMAL) olarak ayarlandıysa, kuyruğa bir JMS kalıcı iletisi IBM MQ kalıcı iletisi olarak konur ve JMS kalıcı olmayan bir ileti IBM MQ kalıcı olmayan bir ileti olarak kuyruğa konur.

Bir JMS kalıcı iletisi, IBM MQ kalıcı olmayan bir ileti olarak kuyruğa konursa ve bir kuyruk yöneticisinin susturulmuş ya da anında kapatılmasının ardından iletinin atılmamasını istiyorsanız, iletinin yöneltilebileceği tüm kuyruklar NPMCLASS (HIGH) olarak ayarlanmalıdır. Yayınlama/abone olma etki alanında, bu kuyruklar abone kuyruklarını içerir. Bu yapılandırmanın uygulanmasına yardımcı olmak için IBM MQ classes for JMS , bir uygulama PERSISTENCE özelliğinin HIGH değerine sahip olduğu ve temeldeki IBM MQ kuyruğunun NPMCLASS (NORMAL) olarak ayarlandığı bir hedef için bir ileti tüketicisi yaratmaya çalışırsa InvalidDestinationkural dışı durumu verir.

Bir hedefin PERSISTENCE özelliğinin HIGH olarak ayarlanması, o hedeften bir iletinin nasıl alınacağını etkilemez. JMS kalıcı iletisi olarak gönderilen bir ileti, JMS kalıcı iletisi olarak alınır ve JMS kalıcı olmayan ileti olarak gönderilen bir ileti, JMS kalıcı olmayan iletisi olarak alınır.

Bir uygulama ilk iletiyi PERSISTENCE özelliğinin HIGH değerine sahip olduğu bir hedefe gönderdiğinde ya da bir uygulama PERSISTENCE özelliğinin HIGH değerine sahip olduğu bir hedef için ilk ileti tüketicisi oluşturduğunda, IBM MQ classes for JMS NPMCLASS (HIGH) ' ın temeldeki IBM MQ kuyruğunda ayarlanıp ayarlanmadığını saptamak için bir MQINQ çağrısı yayınlar. Bu nedenle, uygulamanın kuyrukta sorma yetkisi olmalıdır. Buna ek olarak IBM MQ classes for JMS , hedef silininceye kadar MQINQ çağrısının sonucunu korur ve daha fazla MQINQ çağrısı vermez. Bu nedenle, uygulama hedefi kullanmaya devam ederken temel kuyruktaki NPMCLASS ayarını değiştirirseniz, IBM MQ classes for JMS yeni ayarı fark etmez.

JMS Kalıcı iletilerin IBM MQ kuyruklarına IBM MQ kalıcı olmayan iletiler olarak konmasına izin vererek, bazı güvenilirlik pahasına performans kazanıyorsunuz. JMS kalıcı iletileri için en yüksek güvenilirliğe gereksinim duyarsanız, iletileri PERSISTENCE özelliğinin HIGH değerine sahip olduğu bir hedefe göndermeyin.

JMS Katmanı, SYSTEM.DEFAULT.MODEL.QUEUEyerine SYSTEM.JMS.TEMPQ.MODELkullanabilir. SYSTEM.JMS.TEMPQ.MODEL , SYSTEM.DEFAULT.MODEL.QUEUE , kalıcı iletileri kabul edemiyor. Kalıcı iletileri kabul etmek üzere geçici kuyrukları kullanmak için SYSTEM.JMS.TEMPQ.MODELya da model kuyruğunu seçtiğiniz diğer bir kuyruğa çevirin.

TLS ' yi IBM MQ classes for JMS ile kullanma

IBM MQ classes for JMS uygulamaları, Transport Layer Security (TLS) şifrelemesini kullanabilir. Bunu yapmak için bir JSSE sağlayıcısı gerekir.

TRANSPORT (CLIENT) kullanan IBM MQ classes for JMS bağlantıları TLS şifrelemesini destekler. TLS, iletişim şifrelemesi, kimlik doğrulaması ve ileti bütünlüğü sağlar. Genellikle İnternet üzerindeki ya da bir intranet içindeki herhangi iki eş arasındaki iletişimin güvenliğini sağlamak için kullanılır.

IBM MQ classes for JMS , TLS şifrelemesini işlemek için Java Secure Socket Extension (JSSE) kullanır ve bu nedenle bir JSSE sağlayıcısı gerektirir. JSE v1.4 JVM ' lerin yerleşik bir JSSE sağlayıcısı vardır. Sertifikaların nasıl yönetileceği ve saklanacağına ilişkin ayrıntılar, sağlayıcıdan sağlayıcıya farklılık gösterebilir. Bu konuda bilgi için JSSE sağlayıcınızın belgelerine bakın.

Bu bölümde, JSSE sağlayıcınızın doğru olarak kurulduğu ve yapılandırıldığı ve uygun sertifikaların kurulduğu ve JSSE sağlayıcınızın kullanımına sunulduğu varsayılır. Artık birkaç yönetim özelliği ayarlamak için JMSAdmin 'i kullanabilirsiniz.

IBM MQ classes for JMS uygulamanız bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, bkz. “IBM MQ classes for JMS ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 273.

SSLCIPHERSUITE nesne özelliği
ConnectionFactory nesnesinde TLS şifrelemesini etkinleştirmek için SSLCIPHERSUITE öğesini ayarlayın.

ConnectionFactory nesnesinde TLS şifrelemesini etkinleştirmek için JMSAdmin 'i kullanarak SSLCIPHERSUITE özelliğini JSSE sağlayıcınız tarafından desteklenen bir CipherSuite değerine ayarlayın. Bu, hedef kanalda ayarlanan CipherSpec ile eşleşmelidir. Ancak CipherSuites , CipherSpecs ' ten farklıdır ve bu nedenle farklı adlara sahiptir. “IBM MQ classes for JMS içinde TLS CipherSpecs ve CipherSuites” sayfa 249 , IBM MQ tarafından desteklenen CipherSpecs ' i JSSE tarafından bilinen eşdeğer CipherSuites ile eşlemek için bir tablo içerir. IBM MQile CipherSpecs ve CipherSuites hakkında daha fazla bilgi için bkz. IBM MQ güvenliğinin sağlanması.

Örneğin, TLS_RSA_WITH_AES_128_CBC_SHATLS_RSA_WITH_AES_128_CBC_SHA CipherSpec özelliğine sahip TLS etkinleştirilmiş bir MQI kanalı üzerinden bağlantı yaratmak için kullanılabilecek bir ConnectionFactory nesnesi ayarlamak için JMSAdmin 'e şu komutu verin:

```
ALTER CF(my.cf) SSLCIPHERSUITE(SSL_RSA_WITH_AES_128_CBC_SHA)
```

Bu, bir MQConnectionFactory nesnesinde setSSLCipherSuite () yöntemi kullanılarak bir uygulamadan da ayarlanabilir.

Kolaylık olması için, SSLCIPHERSUITE özelliğinde bir CipherSpec belirtilirse, JMSAdmin CipherSpec öğesini uygun bir CipherSuite ile eşlemeyi dener ve bir uyarı verir. Özellik bir uygulama tarafından belirtilirse bu eşleme girişiminde bulunulmaz.

Diğer bir seçenek olarak, İstemci Kanal Tanımlama Çizelgesi 'ni (CCDT) kullanın. Daha fazla bilgi için bkz “IBM MQ classes for JMS ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 273.

SSLFIPSREQUIRED nesne özelliği
IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) tarafından desteklenen bir CipherSuite kullanmak için bağlantıya gereksinim duyarsanız, bağlantı üreticisinin SSLFIPSREQUIRED özelliğini YES olarak ayarlayın.

Not: AIX, Linux, and Windowsişletim sistemlerinde IBM MQ , IBM Crypto for C (ICC) şifreleme modülü aracılığıyla FIPS 140-2 uyumluluğu sağlar. Bu modüle ilişkin sertifika Geçmiş durumuna taşındı. Müşteriler, IBM Crypto for C (ICC) sertifikasını görüntüleyip NIST tarafından sağlanan tüm önerilere dikkat etmelidir. Yeni bir FIPS 140-3 modülü şu anda devam ediyor ve durumu İşlem listesindeki NIST CMVP modüllerindearanarak görüntülenebilir.

Bu özelliğin varsayılan değeri NO 'dur; bu, bir bağlantının IBM MQtarafından desteklenen herhangi bir CipherSuite ' i kullanabileceği anlamına gelir.

Bir uygulama birden çok bağlantı kullanıyorsa, uygulama ilk bağlantıyı yarattığında kullanılan SSLFIPSREQUIRED değeri, uygulama sonraki bir bağlantı yarattığında kullanılacak değeri belirler. Bu, sonraki bir bağlantı yaratmak için kullanılan bağlantı üreticisinin SSLFIPSREQUIRED özelliğinin değerinin yoksayıldığı anlamına gelir. Farklı bir SSLFIPSREQUIRED değerini kullanmak istiyorsanız uygulamayı yeniden başlatmanız gerekir.

Bir uygulama, ConnectionFactory nesnesinin setSSLFipsRequired () yöntemini çağırarak bu özelliği ayarlayabilir. CipherSuite ayarlanmazsa özellik yoksayılır.

İlgili görevler
MQI istemcisinde çalıştırma zamanında yalnızca FIPS onaylı CipherSpecs kullanılmasının belirtilmesi
İlgili başvurular
AIX, Linux, and Windows için Federal Bilgi İşleme Standartları (FIPS)

SSLPEERNAME nesne özelliği
JMS uygulamanızın doğru kuyruk yöneticisine bağlandığından emin olmak için bir ayırt edici ad örüntüsü belirtmek üzere SSLPEERNAME öğesini kullanın.

Bir JMS uygulaması, ayırt edici ad (DN) örüntüsü belirterek doğru kuyruk yöneticisine bağlanmasını güvenceye alabilir. Bağlantı, kuyruk yöneticisi kalıpla eşleşen bir DN sunduğunda başarılı olur. Bu örüntünün biçimiyle ilgili daha fazla ayrıntı için ilgili konulara bakın.

DN, bir ConnectionFactory nesnesinin SSLPEERNAME özelliği kullanılarak ayarlanır. Örneğin, aşağıdaki JMSAdmin komutu bir ConnectionFactory nesnesinin, kuyruk yöneticisinin kendisini QMGR. karakterleriyle başlayan bir Ortak Ad ile ve en az iki Kuruluş Birimi adıyla (ilkinin IBM ve ikinci WEBSPHERE olması gereken) tanımasını beklemesini sağlar:

```
ALTER CF(my.cf) SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Denetim büyük ve küçük harfe duyarlı değildir ve virgül yerine noktalı virgül kullanılabilir. SSLPEERNAME, bir MQConnectionFactory nesnesinde setSSLPeerName () yöntemi kullanılarak bir uygulamadan da ayarlanabilir. Bu özellik ayarlanmazsa, kuyruk yöneticisi tarafından sağlanan Ayırt Edici Ad üzerinde denetim gerçekleştirilmez. CipherSuite ayarlanmazsa bu özellik yoksayılır.

SSLCERTSTORES nesne özelliği
Sertifika iptal listesi (CRL) denetimi için kullanılacak LDAP sunucularının bir listesini belirtmek için SSLCERTSTORES kullanın.

Artık güvenilmeyen sertifikaları tanımlamak için bir sertifika iptal listesi (CRL) kullanılması yaygındır. CRL ' ler genellikle LDAP sunucularında barındırılır. JMS , Java 2 v1.4 ya da üstü altında CRL denetimi için bir LDAP sunucusunun belirtilmesine izin verir. Aşağıdaki JMSAdmin örneği, JMS 'i crl1.ibm.com:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com)
```

Not: LDAP sunucusunda barındırılan bir CRL ile CertStore ' u başarıyla kullanmak için Java Software Development Kit (SDK) ürününüzü CRL ile uyumlu olduğundan emin olun. Bazı SDK 'lar CRL' nin LDAP v2için bir şema tanımlayan RFC 2587 'ye uymasını gerektirir. LDAP v3 sunucularının çoğu RFC 2256 'yı kullanır.

LDAP sunucunuz varsayılan 389 kapısında çalışmıyorsa, anasistem adının sonuna iki nokta (:) ve kapı numarasını ekleyerek kapıyı belirtebilirsiniz. Kuyruk yöneticisi tarafından sunulan sertifika, crl1.ibm.comüzerinde barındırılan CRL ' de varsa, bağlantı tamamlanmaz. Tek bir hata noktasını önlemek için JMS , boşluk karakteriyle sınırlanmış LDAP sunucularının bir listesini sağlayarak birden çok LDAP sunucusunun sağlanmasına izin verir. Örnek:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com ldap://crl2.ibm.com)
```

Birden çok LDAP sunucusu belirtildiğinde JMS , kuyruk yöneticisinin sertifikasını başarıyla doğrulayabileceği bir sunucu buluncaya kadar sırayla her birini dener. Her sunucu aynı bilgileri içermelidir.

Bu biçimdeki bir dizgi, MQConnectionFactory.setSSLCertStores () yöntemindeki bir uygulama tarafından sağlanabilir. Diğer bir seçenek olarak, uygulama bir ya da daha çok java.security.cert.CertStore nesnesi yaratabilir, bunları uygun bir Derlem nesnesine yerleştirebilir ve bu Derlem nesnesini setSSLCertStores () yöntemine sağlayabilir. Bu şekilde, uygulama CRL denetimini özelleştirebilir. CertStore nesnelerinin oluşturulmasına ve kullanılmasına ilişkin ayrıntılar için JSSE belgelerinize bakın.

Bir bağlantı kurulurken kuyruk yöneticisi tarafından sunulan sertifikanın geçerliliği aşağıdaki gibi denetlenir:

1. sslCertmağazaları tarafından tanıtılan derlemdeki ilk CertStore nesnesi, bir CRL sunucusunu tanımlamak için kullanılır.
2. CRL sunucusuyla iletişim kurma girişiminde bulunuldu.
3. Girişim başarılı olursa, sunucuda sertifika için bir eşleşme aranır.
   a. Sertifikanın iptal edilmiş olduğu tespit edilirse, arama işlemi sona ermiştir ve bağlantı isteği başarısız olur; neden kodu MQRC_SSL_CERTIFICATE_REIPTAL edildi.
   b. Sertifika bulunamazsa, arama işlemi sona ermiştir ve bağlantının devam etmesine izin verilir.
4. Sunucuyla iletişim kurma girişimi başarısız olursa, bir CRL sunucusunu tanımlamak için sonraki CertStore nesnesi kullanılır ve işlem 2. adımdan yinelenir.

Bu, derlemdeki son CertStore ise ya da kaynak grubunda CertStore nesnesi yoksa, arama işlemi başarısız oldu ve bağlantı isteği MQRC_SSL_CERT_STORE_ERROR neden koduyla başarısız oldu.

Derlem nesnesi, CertStores ' un kullanıldığı sırayı belirler.

Uygulamanız bir CertStore nesneleri derlemi ayarlamak için setSSLCertStores () öğesini kullanıyorsa, MQConnectionFactory artık bir JNDI ad alanına bağlanamaz. Bunu yapma girişimi bir kural dışı duruma neden olur. sslCertStores özelliği ayarlanmazsa, kuyruk yöneticisi tarafından sağlanan sertifika üzerinde iptal denetimi gerçekleştirilmez. CipherSuite ayarlanmazsa bu özellik yoksayılır.

SSLRESETCOUNT nesne özelliği
Bu özellik, şifreleme için kullanılan gizli anahtar yeniden anlaşılmadan önce bir bağlantı tarafından gönderilen ve alınan toplam bayt sayısını gösterir.

Gönderilen bayt sayısı, şifrelemeden önceki sayıdır ve alınan bayt sayısı, şifre çözmeden sonraki sayıdır. Bayt sayısı, IBM MQ classes for JMStarafından gönderilen ve alınan denetim bilgilerini de içerir.

Örneğin, 4 MB veri aktıktan sonra yeniden görüşülen bir gizli anahtarla TLS etkin bir MQI kanalı üzerinden bağlantı oluşturmak için kullanılabilecek bir ConnectionFactory nesnesi yapılandırmak için JMSAdmin 'e şu komutu verin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Bir uygulama, ConnectionFactory nesnesinin setSSLResetCount () yöntemini çağırarak bu özelliği ayarlayabilir.

Bu özelliğin değeri sıfır (varsayılan değer) ise, gizli anahtar hiçbir zaman yeniden anlaşılmaz. CipherSuite ayarlanmazsa özellik yoksayılır.

SSLSocketFactory nesne özelliği
Bir uygulamaya ilişkin TLS bağlantısının diğer yönlerini özelleştirmek için bir SSLSocketFactory oluşturun ve JMS ' yi bunu kullanacak şekilde yapılandırın.

Bir uygulama için TLS bağlantısının diğer yönlerini özelleştirmek isteyebilirsiniz. Örneğin, şifreleme donanımını başlatmak ya da kullanılmakta olan anahtar deposunu ve güvenilir depoyu değiştirmek isteyebilirsiniz. Bunu yapmak için, uygulamanın önce uygun şekilde uyarlanmış bir javax.net.ssl.SSLSocketFactory nesnesi yaratması gerekir. Özelleştirilebilir özellikler sağlayıcıdan sağlayıcıya değiştiğinden, bunun nasıl yapılacağını öğrenmek için JSSE belgelerinize bakın. Uygun bir SSLSocketFactory nesnesi alındıktan sonra, JMS ' yi özelleştirilmiş SSLSocketFactory nesnesini kullanacak şekilde yapılandırmak için MQConnectionFactory.setSSLSocketFactory () yöntemini kullanın.

Uygulamanız özelleştirilmiş bir SSLSocketFactory nesnesi ayarlamak için setSSLSocketFactory () yöntemini kullanıyorsa, MQConnectionFactory nesnesi artık bir JNDI ad alanına bağlanamaz. Bunu yapma girişimi bir kural dışı duruma neden olur. Bu özellik ayarlanmazsa, varsayılan SSLSocketFactory nesnesi kullanılır. Varsayılan SSLSocketFactory nesnesinin davranışına ilişkin ayrıntılar için JSSE belgelerinize bakın. CipherSuite ayarlanmazsa bu özellik yoksayılır.

Önemli: Kendi güvenli olmayan bir JNDI ad alanından bir ConnectionFactory nesnesi alındığında, SSL özelliklerinin kullanılmasının güvenliği sağladığını varsaymayın. Özellikle, JNDI ' ın standart LDAP uygulaması güvenli değildir. Bir saldırgan, JMS uygulamasını fark etmeden yanlış sunucuya bağlanmaya yönlendirerek LDAP sunucusunu taklit edebilir. Uygun güvenlik düzenlemeleriyle, JNDI ' ın diğer uygulamaları (fscontext somutlaması gibi) güvenlidir.

JSSE anahtar deposunda ya da güvenli depoda değişiklik yapılması
Anahtar deposunda ya da güvenli depoda değişiklik yaparsanız, değişikliklerin çekilebilmesi için bazı işlemleri gerçekleştirmeniz gerekir.

JSSE anahtar deposu ya da güvenilir deposunun içeriğini değiştirir ya da anahtar deposu ya da güvenli depo dosyasının konumunu değiştirirseniz, o sırada çalışan IBM MQ classes for JMS uygulamaları değişiklikleri otomatik olarak algılamaz. Değişikliklerin yürürlüğe girmesi için aşağıdaki işlemler gerçekleştirilmelidir:

- Uygulamaların tüm bağlantılarını kapatması ve bağlantı havuzlarındaki kullanılmayan bağlantıları yok etmesi gerekir.
- JSSE sağlayıcınız anahtar deposundaki ve güvenilir depodaki bilgileri önbelleğe alırsa, bu bilgilerin yenilenmesi gerekir.

Bu işlemler gerçekleştirildikten sonra uygulamalar bağlantılarını yeniden yaratabilir.

Uygulamalarınızı nasıl tasarladığınıza ve JSSE sağlayıcınız tarafından sağlanan işleve bağlı olarak, uygulamalarınızı durdurmadan ve yeniden başlatmadan bu işlemleri gerçekleştirmek mümkün olabilir. Ancak, uygulamaların durdurulması ve yeniden başlatılması en basit çözüm olabilir.

IBM MQ classes for JMS içinde TLS CipherSpecs ve CipherSuites
IBM MQ classes for JMS uygulamalarının bir kuyruk yöneticisiyle bağlantı kurabilme yeteneği, MQI kanalının sunucu ucunda belirtilen CipherSpec 'e ve istemci ucunda belirtilen CipherSuite ' e bağlıdır.

Aşağıdaki tabloda, IBM MQ tarafından desteklenen CipherSpecs ve bunların eşdeğer CipherSuiteslistelenmektedir.

Deprecated Aşağıdaki çizelgede listelenen CipherSpecsöğesinin IBM MQ tarafından kullanımdan kaldırılıp kaldırılmadığını ve kaldırılıp kaldırılmadığını görmek için kullanımdan kaldırılan CipherSpecs konusunu gözden geçirmeniz gerekir.

Önemli: Listelenen CipherSuites , IBM MQile birlikte verilen IBM Java Runtime Environment (JRE) tarafından desteklenmektedir. Listelenen CipherSuites , Oracle Java JRE tarafından desteklenenleri içerir. Uygulamanızın Oracle Java JRE kullanacak şekilde yapılandırılmasıyla ilgili ek bilgi için Uygulamanızın IBM Java ya da Oracle Java CipherSuite eşlemelerini kullanacak şekilde yapılandırılmasıbaşlıklı konuya bakın.

Çizelge, iletişim için kullanılan protokolü ve CipherSuite ' in FIPS 140-2 standardına uyup uymadığını da gösterir.

Not: AIX, Linux, and Windowsişletim sistemlerinde IBM MQ , IBM Crypto for C (ICC) şifreleme modülü aracılığıyla FIPS 140-2 uyumluluğu sağlar. Bu modüle ilişkin sertifika Geçmiş durumuna taşındı. Müşteriler, IBM Crypto for C (ICC) sertifikasını görüntüleyip NIST tarafından sağlanan tüm önerilere dikkat etmelidir. Yeni bir FIPS 140-3 modülü şu anda devam ediyor ve durumu İşlem listesindeki NIST CMVP modüllerindearanarak görüntülenebilir.

Uygulama, FIPS 140-2 uyumluluğunu zorunlu kılacak şekilde yapılandırılmadıysa, ancak uygulama için FIPS 140-2 uyumluluğu yapılandırıldıysa (yapılandırmaya ilişkin aşağıdaki notlara bakın) yalnızca FIPS 140-2 uyumlu olarak işaretlenmiş CipherSuites yapılandırılabilir; diğer CipherSuites kullanılmaya çalışılması bir hatayla sonuçlanır.

Not: Her JRE birden çok şifreleme güvenliği sağlayıcıya sahip olabilir; bunların her biri aynı CipherSuiteuygulamasına katkıda bulunabilir. Ancak, tüm güvenlik sağlayıcıları FIPS 140-2 sertifikalı değildir. Bir uygulama için FIPS 140-2 uyumluluğu uygulanmazsa, CipherSuite için onaylanmamış bir uygulama kullanılabilir. CipherSuite teorik olarak standardın gerektirdiği minimum güvenlik düzeyini karşılasa bile, onaylanmamış uygulamalar FIPS 140-2 ile uyumlu çalışmayabilir. IBM MQ JMS uygulamalarında FIPS 140-2 uygulamasını yapılandırma hakkında daha fazla bilgi için aşağıdaki notlara bakın.

CipherSpecs ve CipherSuitesiçin FIPS 140-2 ve Suite-B uyumluluğu hakkında daha fazla bilgi için bkz. CipherSpecs. ABD Federal Information Processing Standards(Federal Bilgi İşleme Standartları) ile ilgili bilgileri de bilmeniz gerekebilir.

CipherSuites ' in tam kümesini kullanmak ve sertifikalı FIPS 140-2 ve/veya Suite-B uyumluluğuyla çalışmak için uygun bir JRE gereklidir. IBM Java 7 Service Refresh 4 Fix Pack 2 ya da daha yüksek bir IBM JRE düzeyi, Çizelge 41 sayfa 250içinde listelenen TLS 1.2 CipherSuites için uygun desteği sağlar.

TLS 1.3 şifrelerini kullanabilmek için uygulamanızı çalıştıran JRE 'nin TLS 1.3' ü desteklemesi gerekir.

Not: Bazı CipherSuiteskullanmak için JRE 'de' unrestricted ' ilke dosyalarının yapılandırılması gerekir. İlke dosyalarının bir SDK ya da JRE ' de nasıl ayarlandığına ilişkin daha fazla ayrıntı için, kullandığınız sürüme ilişkin *Security Reference for IBM SDK, Java Technology Edition* adlı yayındaki *IBM SDK Policy files* başlıklı konuya bakın.

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	evet

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_AES_128_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS 1.2	evet

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_AES_128_GCM_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS 1.2	evet

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_AES_256_CBC_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS 1.2	evet

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_AES_256_GCM_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS 1.2	evet
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLS 1.2	hayır

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS 1.2	hayır
ECDHE_RSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	evet

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_AES_128_CBC_SHA256	SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	evet

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_AES_128_GCM_SHA256	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	evet

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_AES_256_CBC_SHA384	SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS 1.2	evet

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_AES_256_GCM_SHA384	SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	evet
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLS 1.2	hayır

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLS 1.2	hayır
TLS_RSA_WITH_3DES_EDE_CBC_SHA "2" sayfa 268	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.0	hayır "4" sayfa 268

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	hayır "4" sayfa 268
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	hayır "4" sayfa 268

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_RSA_WITH_AES_128_GCM_SHA256	SSL_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	hayır "4" sayfa 268
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0	hayır "4" sayfa 268

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	hayır "4" sayfa 268
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	hayır "4" sayfa 268

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLS 1.0	hayır
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	TLS_RSA_WITH_NULL_SHA256	TLS 1.2	hayır
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA	TLS 1.2	hayır

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ANY_TLS12	*TLS12	*TLS12	TLS 1.2	evet
TLS_AES_128_GCM_SHA256 "3" sayfa 268	TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLS V1.3	hayır
TLS_AES_256_GCM_SHA384 "3" sayfa 268	TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLS V1.3	hayır

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_CHACHA20_POLY1305_SHA256 "3" sayfa 268	TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLS V1.3	hayır
TLS_AES_128_CCM_SHA256 "3" sayfa 268	TLS_AES_128_CCM_SHA256	TLS_AES_128_CCM_SHA256	TLS V1.3	hayır

Çizelge 41. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 268**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_AES_128_CCM_8_SHA256 "3" sayfa 268	TLS_AES_128_CCM_8_SHA256	TLS_AES_128_CCM_8_SHA256	TLS V1.3	hayır
Herhangi"3" sayfa 268	*ANY	*ANY	Çoklu	hayır
ANY_TLS13 "3" sayfa 268	*TLS13	*TLS13	TLS V13	hayır
ANY_TLS12_OR_HIGHER "3" sayfa 268	*TLS12ORHIGHER	*TLS12ORHIGHER	TLS 1.2 ve üstü	hayır
ANY_TLS13_OR_HIGHER "3" sayfa 268	*TLS13ORHIGHER	*TLS13ORHIGHER	TLS 1.3 ve üstü	hayır

Notlar:

1. Bu, bir CCDT (ikili ya da JSON) de dahil olmak üzere IBM MQiçindeki bir kanalda yapılandırılan değerdir.
2. Deprecated CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA kullanımdan kaldırılmıştır. Ancak, bağlantı AMQ9288hatasıyla sonlandırılmadan önce 32 GB ' ye kadar veri aktarmak için kullanılabilir. Bu hatayı önlemek için üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckullanırken gizli anahtar sıfırlamasını etkinleştirmeniz gerekir.
3. TLS v1.3 şifrelerini kullanabilmek için uygulamanızı çalıştıran Java runtime environment (JRE) TLS v1.3' ü desteklemelidir.
4. V 9.3.0.17 V 9.3.5.1 IBM MQ 9.3.5 CSU 1 ve IBM MQ 9.3.0 CSU 17işletim sistemlerinde IBM Java 8 JRE, FIPS kipinde çalışırken RSA anahtar değiştokuşu desteğini kaldırır.

IBM MQ classes for JMS uygulamasında şifreleme takımlarını ve FIPS uyumluluğunu yapılandırma

- IBM MQ classes for JMS kullanan bir uygulama, bir bağlantı için CipherSuite ' i ayarlamak üzere iki yöntemden birini kullanabilir:
  - Bir ConnectionFactory nesnesinin setSSLCipherSuite yöntemini çağırın.
  - Bir ConnectionFactory nesnesinin SSLCIPHERSUITE özelliğini ayarlamak için IBM MQ JMS yönetim aracını kullanın.
- IBM MQ classes for JMS kullanan bir uygulama, FIPS 140-2 uyumluluğunu zorlamak için iki yöntemden birini kullanabilir:
  - Bir ConnectionFactory nesnesinin setSSLFipsgerekli yöntemini çağırın.
  - Bir ConnectionFactory nesnesinin SSLFIPSREQUIRED özelliğini ayarlamak için IBM MQ JMS yönetim aracını kullanın.

Uygulamanızın IBM Java ya da Oracle Java CipherSuite eşlemelerini kullanacak şekilde yapılandırılması

Not: V 9.3.3 Continuous Delivery from IBM MQ 9.3.3için, hangi eşlemelerin kullanılacağını denetleyen Java Sistem Özelliği `com.ibm.mq.cfg.useIBMCipherMappings`süründen kaldırılır. IBM MQ 9.3.3' den bir Şifre, CipherSpec ya da CipherSuite adı olarak tanımlanabilir ve IBM MQtarafından doğru şekilde işlenir. IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging uygulamanız bir kuyruk yöneticisine güvenli TLS bağlantıları oluşturursa, aşağıdaki üç Jackson JAR dosyası gereklidir:

- `jackson-annotations.jar`
- `jackson-core.jar`
- `jackson-databind.jar`

Önemli: `com.ibm.mq.cfg.useIBMCipherMappings` ile ilgili aşağıdaki bilgiler yalnızca IBM MQ 9.3.3 öncesinde Long Term Support ve Continuous Delivery için geçerlidir.

Uygulamanızın varsayılan IBM Java CipherSuite ile IBM MQ CipherSpec eşlemelerini mi, yoksa Oracle CipherSuite ile IBM MQ CipherSpec eşlemelerini mi kullanacağını yapılandırabilirsiniz. Bu nedenle, uygulamanızın IBM JRE ya da Oracle JRE kullandığından bağımsız olarak TLS CipherSuites kullanabilirsiniz. Java Sistem Özelliği `com.ibm.mq.cfg.useIBMCipherMappings` hangi eşlemelerin kullanılacağını denetler. Özellik aşağıdaki değerlerden biri olabilir:

doğru
IBM Java CipherSuite - IBM MQ CipherSpec eşlemelerini kullanın.

Bu değer varsayılan değerdir.

yanlış
Oracle CipherSuite - IBM MQ CipherSpec eşlemelerini kullanın.

IBM MQ Java ve TLS şifrelerinin kullanılmasıyla ilgili daha fazla bilgi için MQ Java, TLS Şifreleri, Non-IBM JRE 'leri ve APAR' ları IT06775, IV66840, IT09423, IT10837başlıklı MQdev web günlüğü gönderisine bakın.

Birlikte çalışabilirlik sınırlamaları

Bazı CipherSuites , kullanılmakta olan protokole bağlı olarak birden çok IBM MQ CipherSpecile uyumlu olabilir. Ancak yalnızca Tablo 1 'de belirtilen TLS sürümünü kullanan CipherSuite/CipherSpec birleşimi desteklenir. Desteklenmeyen CipherSuites ve CipherSpecs birleşimlerini kullanma girişimi uygun bir kural dışı durumla başarısız olur. Bu CipherSuite/CipherSpec birleşimlerinden herhangi birini kullanan kuruluşlar, desteklenen bir birleşime taşınmalıdır.

Aşağıdaki tabloda, bu sınırlamanın geçerli olduğu CipherSuites gösterilmektedir.

Çizelge 42. CipherSuites ve bunların desteklenen ve desteklenmeyen CipherSpecs

CipherSuite	**Desteklenen TLS CipherSpec**	**Desteklenmeyen SSL CipherSpec**
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA "1" sayfa 269	TRIPLE_DES_SHA_US
SSL_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA	DES_SHA_EXPORT (DışAKTARMA)
SSL_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_RC4_128_SHA256	RC4_SHA_US

Not:

1. Deprecated Bu CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA kullanımdan kaldırılmıştır. Ancak, bağlantı AMQ9288hatasıyla sonlandırılmadan önce 32 GB ' ye kadar veri aktarmak için kullanılabilir. Bu hatayı önlemek için üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckullanırken gizli anahtar sıfırlamasını etkinleştirmeniz gerekir.

IBM MQ classes for JMS için Java içinde kanal çıkışları yazılıyor
Kanal çıkışlarını, belirtilen arabirimleri gerçekleştiren Java sınıflarını tanımlayarak yaratırsınız.

Güvenlik çıkışlarına giriş için Kanal güvenliği çıkış programları başlıklı konudan başlayın.

com.ibm.mq.exits paketinde üç arabirim tanımlanır:

- WMQSendExit, gönderme çıkışı için
- Alma çıkışı için WMQReceiveExit
- WMQSecurityExit, güvenlik çıkışı için

Aşağıdaki örnek kod, üç arabirimi de gerçekleştiren bir sınıfı tanımlar:

```
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
  }
}
```

Her çıkış, bir MQCXP nesnesi ve bir MQCD nesnesi olarak alır. Bu nesneler, yordamsal arabirimde tanımlı MQCXP ve MQCD yapılarını gösterir.

Bir gönderme çıkışı çağrıldığında, agentBuffer parametresi sunucu kuyruk yöneticisine gönderilmek üzere olan verileri içerir. agentBuffer.limit () ifadesi verilerin uzunluğunu sağladığından, uzunluk parametresi gerekmez. Gönderme çıkışı, sunucu kuyruk yöneticisine gönderilecek verileri değeri olarak döndürür. Ancak, gönderme çıkışı bir gönderme çıkışı sırasındaki son gönderme çıkışı değilse, döndürülen veriler, sıradaki sonraki gönderme çıkışına geçirilir. Gönderme çıkışı, agentBuffer değiştirgesinde aldığı verilerin değiştirilmiş bir sürümünü döndürebilir ya da verileri değişmeden döndürebilir. Bu nedenle, olası en basit çıkış gövdesi:

```
{ return agentBuffer; }
```

Bir alma çıkışı çağrıldığında, agentBuffer parametresi sunucu kuyruk yöneticisinden alınan verileri içerir. Alma çıkışı, IBM MQ classes for JMStarafından uygulamaya geçirilecek verilerin değeri olarak döndürülür. Ancak, alma çıkışı bir alma çıkışları sırasındaki son alma çıkışı değilse, döndürülen veriler sıradaki sonraki alma çıkışına geçirilir.

Bir güvenlik çıkışı çağrıldığında, agentBuffer parametresi, bağlantının sunucu ucundaki güvenlik çıkışından alınan verileri içerir. Güvenlik çıkışı, sunucu güvenlik çıkışına bir güvenlik akışında gönderilecek verilerin değeri olarak döndürülür.

Kanal çıkışları, arka dizisi olan bir arabellekle çağrılır. En iyi başarımı elde etmek için, çıkışın bir arka diziye sahip bir arabellek döndürmesi gerekir.

Çağrıldığında kanal çıkışına en çok 32 karakterlik kullanıcı verileri geçirilebilir. Çıkış, MQCXP nesnesinin getExitData () yöntemini çağırarak kullanıcı verilerine erişir. Çıkış, setExitData () yöntemini çağırarak kullanıcı verilerini değiştirebilse de, çıkış her çağrıldığında kullanıcı verileri yenilenir. Bu nedenle, kullanıcı verilerinde yapılan değişiklikler kaybolur. Ancak çıkış, MQCXP nesnesinin çıkış kullanıcı alanını kullanarak bir çağrıdan diğerine veri geçirebilir. Çıkış, getExitUserArea() yöntemini çağırarak başvuru yoluyla çıkış kullanıcı alanına erişir.

Her çıkış sınıfının bir oluşturucusu olmalıdır. Oluşturucu, önceki örnekte gösterildiği gibi varsayılan oluşturucu ya da dizgi değiştirgesi olan bir oluşturucu olabilir. Oluşturucu, sınıfta tanımlı her çıkışa ilişkin çıkış sınıfının somut örneğini yaratmak için çağrılır. Bu nedenle, önceki örnekte, gönderme çıkışı için MyMQExits sınıfının bir eşgörünümü yaratılır, alma çıkışı için başka bir yönetim ortamı yaratılır ve güvenlik çıkışı için üçüncü bir yönetim ortamı yaratılır. Dizgi değiştirgesi olan bir oluşturucu çağrıldığında, değiştirge, yönetim ortamının yaratıldığı kanal çıkışına geçirilen kullanıcı verileriyle aynı verileri içerir. Bir çıkış sınıfının hem varsayılan oluşturucusu, hem de tek değiştirge oluşturucusu varsa, tek değiştirge oluşturucu önceliklidir.

Bağlantıyı kanal çıkışından kapatmayın.

Veriler bir bağlantının sunucu sonuna gönderildiğinde, TLS şifrelemesi, herhangi bir kanal çıkışının çağrılmasından *sonra* gerçekleştirilir. Benzer şekilde, bir bağlantının sunucu ucundan veri alındığında, herhangi bir kanal çıkışının çağrılmasından *önce* TLS şifre çözme işlemi gerçekleştirilir.

IBM WebSphere MQ 7.0öncesi IBM MQ classes for JMS sürümlerinde, kanal çıkışları MQSendExit, MQReceiveExitve MQSecurityExitarabirimleri kullanılarak gerçekleştirilmiştir. Bu arabirimleri kullanmaya devam edebilirsiniz, ancak yeni arabirimler geliştirilmiş işlev ve başarım için tercih edilir.

IBM MQ classes for JMS ' nin kanal çıkışlarını kullanacak şekilde yapılandırılması
IBM MQ classes for JMS uygulaması, uygulama bir kuyruk yöneticisine bağlandığında başlayan MQI kanalında kanal güvenliğini, gönderme ve alma çıkışlarını kullanabilir. Uygulama, Java, C ya da C + + ile yazılmış çıkışları kullanabilir. Uygulama, art arda çalıştırılan bir gönderme ya da alma çıkışları sırasını da kullanabilir.

Aşağıdaki özellikler, bir gönderme çıkışını ya da JMS bağlantısı tarafından kullanılan bir gönderme çıkışları sırasını belirtir:

- Bir MQConnectionFactory nesnesinin **SENDEXIT** özelliği.
- Gelen iletişim için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir etkinleştirme belirtimindeki **sendexit** özelliği,
- Çıkış iletişimi için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir ConnectionFactory nesnesindeki **sendexit** özelliği.

Özelliğin değeri, virgülle ayrılmış bir ya da daha çok öğeden oluşan bir dizedir. Her öğe, gönderme çıkışını aşağıdaki yollardan biriyle tanımlar:

- Javaiçine yazılan bir gönderme çıkışı için `WMQSendExit` arabirimini uygulayan bir sınıfın adı.
- C ya da C + + ile yazılmış bir gönderme çıkışı için *libraryName* (*entryPointName*) biçiminde bir dizgi.

Benzer bir şekilde, aşağıdaki özellikler bir bağlantı tarafından kullanılan alma çıkışını ya da alma çıkışlarının sırasını belirtir:

- Bir MQConnectionFactory nesnesinin **RECEXIT** özelliği.
- Gelen iletişim için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir etkinleştirme belirtimindeki **receiveexit** özelliği,
- Çıkış iletişimi için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir ConnectionFactory nesnesindeki **receiveexit** özelliği.

Aşağıdaki özellikler, bir bağlantı tarafından kullanılan güvenlik çıkışını belirtir:

- Bir MQConnectionFactory nesnesinin **SECEXIT** özelliği.
- Gelen iletişim için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir etkinleştirme belirtimindeki **securityexit** özelliği,
- Çıkış iletişimi için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir ConnectionFactory nesnesindeki **securityexit** özelliği.

MQConnectionFactoriesiçin, IBM MQ JMS yönetim aracını ya da IBM MQ Explorerkullanarak **SENDEXIT**, **RECEXIT** ve **SECEXIT** özelliklerini ayarlayabilirsiniz. Alternatif olarak, bir uygulama `setSendExit()`, `setReceiveExit()`ve `setSecurityExit()` yöntemlerini çağırarak özellikleri ayarlayabilir.

Kanal çıkışları kendi sınıf yükleyicisi tarafından yüklenir. Bir kanal çıkışını bulmak için, sınıf yükleyici aşağıdaki yerleri belirtilen sırayla arar.

1. IBM MQ istemci yapılandırma dosyasının Kanalları kısmına ilişkin **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** özelliği ya da **JavaExitsClassPath** özniteliği tarafından belirtilen sınıf yolu.
2. Deprecated Java Sistem özelliği **com.ibm.mq.exitClasspath**tarafından belirtilen sınıf yolu. Bu özelliğin artık kullanımdan kaldırılmış olduğunu unutmayın.
3. IBM MQ , Çizelge 43 sayfa 271içinde gösterildiği gibi dizinden çıkar. Sınıf yükleyici öncelikle Java arşiv (JAR) dosyalarında paketlenmeyen sınıf dosyalarını arar. Kanal çıkışı bulunamazsa, sınıf yükleyici dizindeki JAR dosyalarını arar.

Çizelge 43. IBM MQ çıkış dizini

Hizmet olarak sunulan	Dizin
Linux AIX AIX and Linux	`/var/mqm/exits` (32 bit kanal çıkışları) `/var/mqm/exits64` (64 bit kanal çıkışları)

Çizelge 43. IBM MQ çıkış dizini (devamı var)	
Hizmet olarak sunulan	**Dizin**
Windows Windows	*kuruluş_veri_dizini*\çıkışları Burada *kuruluş_veri_dizini* , kuruluş sırasında IBM MQ veri dosyaları için seçtiğiniz dizindir. Varsayılan dizin C:\ProgramData\IBM\MQdizinidir.

Not: Bir kanal çıkışı birden çok yerde varsa, IBM MQ classes for JMS bulduğu ilk örneği yükler.

Sınıf yükleyicinin üst öğesi, IBM MQ classes for JMSöğesini yüklemek için kullanılan sınıf yükleyicidir. Bu nedenle, üst sınıf yükleyicinin, önceki konumların hiçbirinde bulunamazsa, bir kanal çıkışını yüklemesi mümkündür. Ancak, IBM MQ classes for JMS ürününü JEE uygulama sunucusu gibi bir ortamda kullanırken, üst sınıf yükleyicisinin seçimini etkileyemeyeceksiniz ve bu nedenle sınıf yükleyici, uygulama sunucusunda Java sistem özelliği **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** ayarlanarak yapılandırılmalıdır.

Uygulamanız Java security manager etkinleştirilmiş olarak çalıştırılıyorsa, uygulamanın çalıştığı Java yürütme ortamı tarafından kullanılan ilke yapılandırma dosyası, bir kanal çıkış sınıfını yükleme izinlerine sahip olmalıdır. Bunun nasıl yapılacağını öğrenmek için bkz. Java Security Manager altında JMS uygulamaları için IBM MQ sınıflarının çalıştırılması.

IBM WebSphere MQ 7.0 sürümünden önceki sürümlerle birlikte sağlanan `MQSendExit`, `MQReceiveExit`ve `MQSecurityExit` arabirimleri hala desteklenmektedir. Bu arabirimleri gerçekleştiren kanal çıkışlarını kullanıyorsanız, `com.ibm.mq.jar` sınıf yolunda bulunmalıdır.

C dilinde kanal çıkışlarının nasıl yazılacağı hakkında bilgi için bkz. “İleti sistemi kanalları için kanal çıkış programları” sayfa 920. C ya da C++ dilinde yazılmış kanal çıkış programlarını Çizelge 43 sayfa 271içinde gösterilen dizinde saklamanız gerekir.

Uygulamanız bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, bkz. “IBM MQ classes for JMS ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 273.

IBM MQ classes for JMS kullanılırken kanal çıkışlarına geçirilecek kullanıcı verilerinin belirtilmesi
Çağrıldığında kanal çıkışına en çok 32 karakterlik kullanıcı verileri geçirilebilir.

Bir MQConnectionFactory nesnesinin SENDEXITINIT özelliği, çağrıldığında her gönderme çıkışına geçirilen kullanıcı verilerini belirtir. Özelliğin değeri, virgülle ayrılmış olarak kullanıcı verilerinin bir ya da daha fazla öğesini içeren bir dizedir. Kullanıcı verilerinin dizgi içindeki her bir öğesinin konumu, hangi gönderme çıkışının gönderileceğini, bir gönderme çıkışları sırasında kullanıcı verilerinin geçirileceğini belirler. Örneğin, dizedeki kullanıcı verilerinin ilk öğesi, bir gönderme çıkışları sırasındaki ilk gönderme çıkışına geçirilir.

IBM MQ JMS yönetim aracını ya da IBM MQ Explorerkullanarak SENDEXITINIT özelliğini ayarlayabilirsiniz. Alternatif olarak, bir uygulama setSendExitInit() yöntemini çağırarak özelliği ayarlayabilir.

Benzer bir şekilde, bir ConnectionFactory nesnesinin RECEXITINIT özelliği, her alma çıkışına geçirilen kullanıcı verilerini belirtir ve SECEXITINIT özelliği, bir güvenlik çıkışına geçirilen kullanıcı verilerini belirtir. Bu özellikleri IBM MQ JMS yönetim aracını ya da IBM MQ Explorerkullanarak ayarlayabilirsiniz. Alternatif olarak, bir uygulama setReceiveExitInit() ve setSecurityExitInit() yöntemlerini çağırarak özellikleri ayarlayabilir.

Kanal çıkışlarına geçirilen kullanıcı verilerini belirtirken aşağıdaki kurallara dikkat edin:

- Bir dizedeki kullanıcı verilerinin öğe sayısı bir dizideki çıkış sayısından fazlaysa, kullanıcı verilerinin fazla öğeleri yoksayılır.
- Bir dizedeki kullanıcı verilerinin öğe sayısı bir dizideki çıkış sayısından azsa, kullanıcı verilerinin belirlenmemiş her öğesi boş bir dizgiye ayarlanır. Bir dizgi içinde art arda iki virgül ya da bir dizginin başındaki bir virgül, kullanıcı verilerinin belirlenmemiş bir öğesini de gösterir.

Bir uygulama bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, istemci bağlantı kanalı tanımında belirtilen tüm kullanıcı verileri çağrıldığında kanal çıkışlarına geçirilir. İstemci kanal tanımlama çizelgesinin kullanılmasıyla ilgili daha fazla bilgi için bkz. “IBM MQ classes for JMS ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 273.

IBM MQ classes for JMS ile istemci kanal tanımlama çizelgesinin kullanılması
Bir IBM MQ classes for JMS uygulaması, bir istemci kanal tanımlama çizelgesinde (CCDT) saklanan istemci bağlantısı kanal tanımlarını kullanabilir. CCDT ' yi kullanmak için bir ConnectionFactory nesnesi yapılandırırsınız. Kullanımına ilişkin bazı kısıtlamalar vardır.

Bir ConnectionFactory nesnesinin belirli özelliklerini ayarlayarak istemci bağlantı kanalı tanımlaması yaratmanın bir alternatifi olarak, bir IBM MQ classes for JMS uygulaması istemci kanal tanımlama çizelgesinde saklanan istemci bağlantı kanalı tanımlamalarını kullanabilir. Bu tanımlar IBM MQ Script (MQSC) komutları ya da IBM MQ Programlanabilir Komut Biçimi (PCF) komutlarıyla yaratılır. Uygulama bir Bağlantı nesnesi yarattığında, IBM MQ classes for JMS uygun bir istemci bağlantı kanalı tanımlaması için istemci kanal tanımlama çizelgesini arar ve bir MQI kanalını başlatmak için kanal tanımlamasını kullanır. İstemci kanal tanımlama çizelgelerine ve bunların nasıl oluşturulacağına ilişkin ek bilgi için İstemci kanal tanımlama çizelgesibaşlıklı konuya bakın.

Bir istemci kanal tanımlama çizelgesini kullanmak için, ConnectionFactory nesnesinin CCDTURL özelliği bir URL nesnesine ayarlanmalıdır. IBM MQ classes for JMS , CCDT ile ilgili bilgileri IBM MQ MQI client yapılanış kütüğünden okumayın; ancak, bu dosyada başka bazı değerler de vardır (hangi değerin geçerli olduğunu görmek için bkz. “IBM MQ classes for JMS/Jakarta Messaging yapılandırma dosyası” sayfa 94). URL nesnesi, istemci kanal tanımlama çizelgesini içeren dosyanın adını ve yerini tanımlayan ve dosyaya nasıl erişilebileceğini belirten bir tek tip kaynak konum belirleyici (URL) içerir. IBM MQ JMS yönetim aracını kullanarak CCDTURL özelliğini ayarlayabilir ya da bir uygulama, bir URL nesnesi yaratıp ConnectionFactory nesnesinin setCCDTURL() yöntemini çağırarak özelliği ayarlayabilir.

Örneğin, ccdt1.tab dosyası bir istemci kanal tanımlama çizelgesi içeriyorsa ve uygulamanın çalıştığı sistemde saklanıyorsa, uygulama CCDTURL özelliğini aşağıdaki şekilde ayarlayabilir:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
factory.setCCDTURL(chanTab1);
```

Başka bir örnek olarak, ccdt2.tab dosyasının bir istemci kanal tanımlama çizelgesi içerdiğini ve uygulamanın çalıştığından farklı bir sistemde saklandığını varsayın. Dosyaya FTP protokolü kullanılarak erişilebiliyorsa, uygulama CCDTURL özelliğini aşağıdaki şekilde ayarlayabilir:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
factory.setCCDTURL(chanTab2);
```

ConnectionFactory nesnesinin CCDTURL özelliğini ayarlamanın yanı sıra, aynı nesnenin QMANAGER özelliği aşağıdaki değerlerden birine ayarlanmalıdır:

- Kuyruk yöneticisinin adı
- Bir yıldız imi (*) ve ardından bir kuyruk yöneticisi grubu adı

Bunlar, Message Queue Interface (MQI) kullanan bir istemci uygulaması tarafından verilen bir MQCONN çağrısındaki **QMgrName** değiştirgesi için kullanılabilecek değerlerle aynıdır. Bu değerlerin anlamı hakkında daha fazla bilgi için bkz. MQCONN. IBM MQ JMS yönetim aracını ya da IBM MQ Gezgini 'ni kullanarak QMANAGER özelliğini ayarlayabilirsiniz. Alternatif olarak, bir uygulama ConnectionFactory nesnesinin setQueueManager () yöntemini çağırarak özelliği ayarlayabilir.

Bir uygulama daha sonra ConnectionFactory nesnesinden bir Connection nesnesi yaratırsa, IBM MQ classes for JMS CCDTURL özelliği tarafından tanımlanan istemci kanal tanımlama çizelgesine erişir, uygun bir istemci bağlantı kanalı tanımlaması için çizelgeyi aramak üzere QMANAGER özelliğini kullanır ve daha sonra, kuyruk yöneticisine bir MQI kanalı başlatmak için kanal tanımlamasını kullanır.

Uygulama createConnection() yöntemini çağırdığında, ConnectionFactory nesnesinin CCDTURL ve CHANNEL özelliklerinin her ikisi de ayarlanamaz. Her iki özellik de ayarlanırsa, yöntem bir kural dışı durum

verir. Değeri boş değer, boş dizgi ya da tüm boş karakterleri içeren bir dizgi dışında bir değerse, CCDTURL ya da CHANNEL özelliği ayarlanmış olarak kabul edilir.

IBM MQ classes for JMS , istemci kanal tanımlama çizelgesinde uygun bir istemci bağlantı kanalı tanımlaması bulduğunda, MQI kanalını başlatmak için yalnızca çizelgeden çıkarılan bilgileri kullanır. ConnectionFactory nesnesinin kanalla ilgili özellikleri yoksayılır.

Özellikle TLS kullanıyorsanız aşağıdaki noktalara dikkat edin:

- Bir MQI kanalı TLS 'yi yalnızca istemci kanal tanımlaması çizelgesinden çıkarılan kanal tanımlaması IBM MQ classes for JMStarafından desteklenen bir CipherSpec ' in adını belirtiyorsa kullanır.
- Bir istemci kanal tanımlama çizelgesi, sertifika iptal listelerini (CRL ' ler) bulunduran LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) sunucularının yeriyle ilgili bilgiler de içerir. IBM MQ classes for JMS , CRL ' leri tutan LDAP sunucularına erişmek için yalnızca bu bilgileri kullanır.
- Bir istemci kanal tanımlama çizelgesi, OCSP yanıtlayıcısının yerini de içerebilir. IBM MQ classes for JMS , bir istemci kanal tanımlama çizelgesi kütüğünde OCSP bilgilerini kullanamıyor. Ancak, OCSP ' yi Java ve JMS istemci uygulamalarında OCSP (Online Certificate Status Protocol; Çevrimiçi Sertifika Durumu Protokolü)bölümünde açıklandığı gibi yapılandırabilirsiniz.

TLS ' yi bir istemci kanal tanımlama tablosuyla kullanma hakkında daha fazla bilgi için TLS kanallarıyla genişletilmiş işlem istemcisini kullanmabaşlıklı konuya bakın.

Kanal çıkışlarını kullanıyorsanız aşağıdaki noktalara da dikkat edin:

- Bir MQI kanalı yalnızca, istemci kanal tanımlama çizelgesinden ayıklanan kanal tanımlaması tarafından belirtilen kanal çıkışlarını ve ilişkili kullanıcı verilerini kullanır.
- Bir istemci kanal tanımlama çizelgesinden çıkarılan bir kanal tanımı, Javaiçinde yazılan kanal çıkışlarını belirtebilir. Bu, örneğin, bir istemci bağlantı kanalı tanımlaması yaratmak için DEFINE CHANNEL komutundaki SCYEXIT değiştirgesinin WMQSecurityExit arabirimini gerçekleştiren bir sınıfın adını belirtebileceği anlamına gelir. Benzer şekilde, SENDEXIT değiştirgesi WMQSendExit arabirimini gerçekleştiren bir sınıfın adını belirtebilir ve RCVEXIT değiştirgesi WMQReceiveExit arabirimini gerçekleştiren bir sınıfın adını belirtebilir. Javaiçinde bir kanal çıkışının nasıl yazılacağı hakkında daha fazla bilgi için bkz.“IBM MQ classes for JMS için Java içinde kanal çıkışları yazılıyor” sayfa 269.

  Java dışında bir dilde yazılmış kanal çıkışlarının kullanımı da desteklenir. Başka bir dilde yazılan kanal çıkışları için DEFINE CHANNEL komutunda SCYEXIT, SENDEXIT ve RCVEXIT parametrelerinin nasıl belirtileceğini öğrenmek için DEFINE CHANNELbaşlıklı konuya bakın.

Otomatik JMS istemcisi yeniden bağlantısı
JMS istemcinizi bir ağ, kuyruk yöneticisi ya da sunucu hatasının ardından otomatik olarak yeniden bağlanacak şekilde yapılandırın.

Olağan durumda, istemci iletimi kullanılarak bağımsız bir IBM MQ classes for JMS uygulaması bir kuyruk yöneticisine bağlandıysa ve kuyruk yöneticisi herhangi bir nedenle kullanılamaz duruma gelirse (örneğin, bir ağ kesintisi, kuyruk yöneticisi hatası ya da kuyruk yöneticisi durdurulduğunda), uygulama kuyruk yöneticisiyle iletişim kurmaya çalıştığında IBM MQ classes for JMS bir JMSException oluşturur. Uygulamanın JMSException 'ı yakalaması ve kuyruk yöneticisine yeniden bağlanmayı denemesi gerekir. Otomatik istemci yeniden bağlantısını etkinleştirerek uygulamanın tasarımını basitleştirebilirsiniz. Kuyruk yöneticisi kullanılamaz duruma geldiğinde, IBM MQ classes for JMS uygulama adına kuyruk yöneticisine otomatik olarak yeniden bağlanmayı dener. Bu, uygulamanın yeniden bağlanmak için mantık içermesi gerekmediği anlamına gelir.

Bu otomatik istemci yeniden bağlantısı uygulamasının kullanımı, Java Platform, Enterprise Edition uygulama sunucularında desteklenmez. Alternatif bir uygulama için bkz. “Java EE ortamlarında otomatik istemci yeniden bağlantısının kullanılması” sayfa 280 .

Otomatik JMS istemcisi yeniden bağlantısının kullanılması
Bağımsız bir IBM MQ classes for JMS uygulaması CONNECTIONNAMELIST ya da CCDTURL özelliği ayarlanmış bir bağlantı üreticisini kullanıyorsa, uygulama otomatik istemci yeniden bağlantısını kullanabilir.

Otomatik istemci yeniden bağlantısı, yüksek kullanılabilirlik (HA) yapılandırmasının bir parçası olanlar da içinde olmak üzere kuyruk yöneticilerine yeniden bağlanmak için kullanılabilir. HA yapılandırmaları, IBM MQ aygıtında çok eşgörünümlü kuyruk yöneticilerini, RDQM kuyruk yöneticilerini ya da HA kuyruk yöneticilerini içerir.

IBM MQ classes for JMS tarafından sağlanan otomatik istemci yeniden bağlanma işlevinin davranışı, aşağıdaki özelliklere bağlıdır:

JMS Connection Factory özelliği TRANSPORT (Kısa ad TRAN)

TRANSPORT, Bağlantı Üreticisi kullanan uygulamaların bir kuyruk yöneticisine nasıl bağlanacağını belirtir. Otomatik istemci yeniden bağlantısının kullanılabilmesi için bu özellik CLIENT değerine ayarlanmalıdır. Otomatik istemci yeniden bağlantısı, TRANSPORT özelliği BIND, DIRECT ya da DIRECTHTTP olarak ayarlanmış bir bağlantı üreticisini kullanan bir kuyruk yöneticisine bağlanan uygulamalar için kullanılamaz.

JMS Connection Factory özelliği QMANAGER (Kısa ad QMGR)

QMANAGER özelliği, Bağlantı Üreticisi 'nin bağlandığı kuyruk yöneticisinin adını belirtir.

JMS Connection Factory özelliği CONNECTIONNAMELIST (Kısa ad CRHOSTS)

CONNECTIONNAMELIST özelliği, virgülle ayrılmış bir listedir; burada her giriş, CLIENT iletimini kullanırken QMANAGER özelliği tarafından belirlenen kuyruk yöneticisine bağlanmak için kullanılacak anasistem adı ve kapısıyla ilgili bilgileri içerir. Liste şu biçimdedir: anasistem adı (kapı), anasistem adı (kapı).

JMS Connection Factory özelliği CCDTURL (Kısa ad CCDT)

CCDTURL özelliği, IBM MQ classes for JMS ' un CCDT kullanarak bir kuyruk yöneticisine bağlanırken kullandığı istemci kanal tanımlama çizelgesini gösterir.

JMS Connection Factory özelliği CLIENTRECONNECTOPTIONS (Kısa ad CROPT)

CLIENTRECONNECTOPTIONS, bir kuyruk yöneticisi kullanılabilir duruma gelirse, IBM MQ classes for JMS ' un bir uygulama adına bir kuyruk yöneticisine otomatik olarak bağlanma girişiminde bulunup bulunmayacağını denetler.

İstemci yapılanış kütüğünün kanal (Channel) kısmına ilişkin DefRecon özniteliği

DefRecon özniteliği, tüm uygulamaların otomatik olarak yeniden bağlanmasını sağlamak ya da otomatik olarak yeniden bağlanmak üzere yazılan uygulamalar için otomatik yeniden bağlantıyı geçersiz kılmak üzere bir denetim seçeneği sağlar.

Otomatik istemci yeniden bağlantısı yalnızca, bir uygulama bir kuyruk yöneticisine başarıyla bağlandığında kullanılabilir.

Bir uygulama CLIENT iletimini kullanan bir kuyruk yöneticisine bağlandığında, uygulamanın bağlı olduğu kuyruk yöneticisi kullanılamaz duruma gelirse, otomatik istemci yeniden bağlantısının kullanılıp kullanılmayacağını saptamak için IBM MQ classes for JMS Connection Factory özelliği CLIENTRECONNECTOPTIONS değerini kullanır. Tablo 1, CLIENTRECONNECTOPTIONS özelliği için olası değerleri ve bu değerlerin her biri için IBM MQ classes for JMS davranışını gösterir:

Çizelge 44. Olası CLIENTRECCECTOPTIONS özellik değerleri.

CLIENTRECONNECTOPTIONS	**IBM MQ classes for JMS davranışı**
Fark Etmez	CONNECTIONNAMELIST ayarlandıysa, anasistem adı ve kapı birleşimine bağlantı açmak için CONNECTIONNAMELIST özelliğinin değerini kullanın ve herhangi bir kuyruk yöneticisine bağlanın. Bu otomatik istemci yeniden bağlanma seçeneğini kullanmak için, QMANAGER özelliğinin varsayılan değere ya da "*" değerine ayarlanması gerekir. CCDTURL ayarlanırsa, CCDTURL özelliği tarafından belirtilen istemci kanal tanımlama çizelgesini açın, çizelgeden bir giriş seçin ve kuyruk yöneticisine istemci bağlantı kanalı başlatmak için bu girişi kullanın. Bu otomatik istemci yeniden bağlanma seçeneğini kullanmak için QMANAGER özelliği aşağıdakilerden birine ayarlanmalıdır: • Yıldız işareti (*) • Bir yıldız imi (*) ve ardından bir kuyruk yöneticisi grubu adı • Boş bir dizgi ya da tüm boş karakterleri içeren bir dizgi
VARLıK	Otomatik istemci yeniden bağlantısının olup olmadığını saptamak için DefRecon değerini kullanın.
DEVRE DIŞI	Otomatik istemci yeniden bağlantısı gerçekleştirmeyin ve uygulamaya JMSException döndürmeyin.
QMGR	İstemcinin aynı kuyruk yöneticisine yeniden bağlanması gerektiğini belirtir. Bu seçenek, aynı kuyruk yöneticisinin başka bir örneğine yeniden bağlanmanın gerekli olduğu yüksek kullanılabilirlikli çözümler için kullanılmalıdır. CONNECTIONNAMELIST ayarlandıysa, bir anasistem adı ve kapı birleşimine bağlantı açmak için CONNECTIONNAMELIST özelliğinin değerini kullanın ve QMANAGER özelliği tarafından belirtilen kuyruk yöneticisine bağlanın. CCDTURL ayarlanırsa, CCDTURL özelliği tarafından belirlenen istemci kanal tanımlama çizelgesini açın, QMANAGER özelliği tarafından belirlenen kuyruk yöneticisi adıyla eşleşen çizelgedeki girişleri bulun ve bu girişleri kullanarak o kuyruk yöneticisine istemci bağlantısı kanalı başlatın.

CONNECTIONNAMELIST ayarlanırsa, otomatik istemci yeniden bağlantısı gerçekleştirdiğinizde IBM MQ classes for JMS , hangi sisteme yeniden bağlanılacağını saptamak için Connection Factory özelliği CONNNECTIONNAMELIST içindeki bilgileri kullanır.

IBM MQ classes for JMS başlangıçta, CONNECTIONNAMELIST içindeki ilk girişte belirtilen anasistem adını ve kapıyı kullanarak yeniden bağlanmayı dener. Bir bağlantı kurulursa, IBM MQ classes for

JMS , QMANAGER özelliğinde belirtilen ada sahip kuyruk yöneticisine bağlanmayı dener. Kuyruk yöneticisiyle bağlantı kurulabilirse, IBM MQ classes for JMS otomatik istemci yeniden bağlanmasından önce uygulamanın açmış olduğu tüm IBM MQ nesnelerini yeniden açar ve önceki gibi çalışmaya devam eder.

CONNECTIONNAMELIST içindeki ilk giriş kullanılarak gerekli kuyruk yöneticisiyle bağlantı kurulamazsa, IBM MQ classes for JMS CONNECTIONNAMELIST içindeki ikinci girişi dener ve bu şekilde devam eder.

IBM MQ classes for JMS CONNECTIONNAMELIST içindeki tüm girişleri denediğinde, yeniden bağlanmayı denemeden önce belirli bir süre bekler. Yeni yeniden bağlanma girişimini gerçekleştirmek için, IBM MQ classes for JMS CONNECTIONNAMELIST içindeki ilk girişle başlar. Daha sonra, yeniden bağlantı oluşuncaya ya da CONNECTIONNAMELIST ' in sonuna ulaşılıncaya kadar CONNECTIONNAMELIST içindeki her girişi dener; bu noktada IBM MQ classes for JMS , yeniden denemeden önce belirli bir süre bekler.

CCDTURL ayarlanırsa, otomatik istemci yeniden bağlantısı gerçekleştirilirken IBM MQ classes for JMS , hangi sisteme yeniden bağlanılacağını saptamak için CCDTURL özelliğinde belirtilen istemci kanal tanımlama çizelgesini kullanır.

IBM MQ classes for JMS başlangıçta istemci kanal tanımlama çizelgesini ayrıştırır ve QMANAGER özelliğinin değeriyle eşleşen uygun bir giriş bulur. Bir giriş bulunduğunda, IBM MQ classes for JMS bu girişi kullanarak gerekli kuyruk yöneticisine yeniden bağlanmayı dener. Kuyruk yöneticisiyle bağlantı kurulabilirse, IBM MQ classes for JMS otomatik istemci yeniden bağlanmasından önce uygulamanın açmış olduğu tüm IBM MQ nesnelerini yeniden açar ve önceki gibi çalışmaya devam eder.

Gerekli kuyruk yöneticisiyle bağlantı kurulamazsa, IBM MQ classes for JMS istemci kanal tanımlama çizelgesinde başka bir giriş arar ve bunu kullanmayı dener.

IBM MQ classes for JMS istemci kanal tanımlama çizelgesindeki uygun girişlerin tümünü denediğinde, yeniden bağlanmayı denemeden önce belirli bir süre bekler. Yeni yeniden bağlanma girişimini gerçekleştirmek için, IBM MQ classes for JMS istemci kanal tanımlama çizelgesini yeniden ayrıştırır ve ilk uygun girişi dener. Daha sonra, bir yeniden bağlantı oluşuncaya ya da istemci kanal tanımlama çizelgesindeki son uygun giriş deneninceye kadar istemci kanal tanımlama çizelgesindeki her girişi dener; bu durumda, IBM MQ classes for JMS yeniden denemeden önce belirli bir süre bekler.

CONNECTIONNAMELIST ya da CCDTURL kullanılıyorsa, IBM MQ classes for JMS QMANAGER özelliği tarafından belirtilen kuyruk yöneticisine başarıyla yeniden bağlanıncaya kadar otomatik istemci yeniden bağlanma işlemi devam eder.

Varsayılan olarak, yeniden bağlanma girişimleri aşağıdaki aralıklarla gerçekleşir:

- İlk deneme, 1 saniyelik başlangıç gecikmesinden ve 250 milisaniyeye kadar rasgele bir öğeden sonra yapılır.
- İkinci deneme 2 saniye artı ilk girişim başarısız olduktan sonra 500 milisaniyeye kadar rasgele bir aralık yapılır.
- Üçüncü deneme 4 saniye, ikinci deneme başarısız olduktan sonra 1 saniyeye kadar rasgele bir aralık yapılır.
- Dördüncü deneme 8 saniye artı üçüncü deneme başarısız olduktan sonra 2 saniyeye kadar rastgele bir aralık yapılır.
- Beşinci deneme 16 saniye ve dördüncü girişim başarısız olduktan sonra 4 saniyeye kadar rastgele bir aralık yapılır.
- Altıncı deneme ve sonraki tüm denemeler 25 saniye, ayrıca önceki denemeden sonra 6 saniye ve 250 milisaniyeye kadar rasgele bir aralık yapılır.

Yeniden bağlanma girişimleri, kısmen sabit ve kısmen rasgele aralıklarla geciktirilir. Bu, artık kullanılamayan bir kuyruk yöneticisine bağlı tüm IBM MQ classes for JMS uygulamalarının eşzamanlı olarak yeniden bağlanmasını önlemek için kullanılır.

Varsayılan değerleri artırmanız gerekiyorsa, bir kuyruk yöneticisinin ya da yedek kuyruk yöneticisinin etkin duruma gelmesi için gereken süreyi daha doğru bir şekilde yansıtmak için, istemci yapılandırma dosyasının Kanal bölümündeki ReconDelay özniteliğini değiştirin (ek bilgi için istemci yapılandırma dosyasının CHANNEL kısmıkonusuna bakın).

Bir IBM MQ classes for JMS uygulamasının otomatik olarak yeniden bağlandıktan sonra tasarımına bağlı olarak çalışmaya devam edip etmeyeceğini belirler. Uygulamaların otomatik yeniden bağlanma işlevini nasıl tasarlayacağını anlamak için ilgili konuları okuyun.

Bir kuyruk yöneticisinin artık kullanılamadığını gösteren neden kodları
Hangi neden kodları, otomatik IBM MQ classes for JMS yeniden bağlanma girişiminde bulunulurken bir kuyruk yöneticisinin artık kullanılamadığını ya da bu yöneticiye ulaşılamadığını gösterir.

"Otomatik JMS istemcisi yeniden bağlantısı" sayfa 274 içinde, JMSExceptions ile ilgili genel bir bakış ve uygulamalarınızın otomatik olarak nasıl yeniden başlatılacağı ve "Otomatik JMS istemcisi yeniden bağlantısının kullanılması" sayfa 274 içindeki bilgiler, otomatik istemci yeniden bağlantısına ilişkin gereksinimleri ayrıntılarıyla verir.

Aşağıdaki bilgiler, uygulamanızın denetlemesi gereken IBM MQ neden kodlarını listeler:

RC2009
MQRC_CONNECTION_BROKEN

RC2059
MQRC_Q_MGR_YOK

RC2161
MQRC_Q_MGR_QUIESCING

RC2162
MQRC_Q_MGR_DURDURULUYOR

RC2202
MQRC_CONNECTION_QUIESCING

RC2203
MQRC_CONNECTION_DURDURULUYOR

RC2223
MQRC_Q_MGR_NOT_ETKIN

RC2279
MQRC_CHANNEL_STOPD_BY_USER

RC2537
MQRC_CHANNEL_NOT_AVAILABLE

RC2538
MQRC_ANASISTEM_YOK

Kurum uygulamalarına geri yayınlanan çoğu JMSExceptions, neden kodunu tutan bağlantılı bir MQException içeriyor. Önceki listedeki neden kodlarına ilişkin yeniden deneme mantığını uygulamak için, kurum uygulamalarınız aşağıdaki örneğe benzer bir kod kullanarak bu bağlantılı kural dışı durumu kontrol etmelidir:

```
} catch (JMSException ex) {
        Exception linkedEx = ex.getLinkedException();
        if (ex.getLinkedException() != null) {
            if (linkedEx instanceof MQException) {
                MQException mqException = (MQException) linkedEx;
                int reasonCode = mqException.reasonCode;
                // Handle the reason code accordingly
            }
        }
    }
```

İlgili kavramlar
JMS için IBM MQ sınıfları

Java SE ve Java EE ortamlarında otomatik istemci yeniden bağlantısının kullanılması
Bir Java SE ve Java EE ortamında çeşitli yüksek düzeyde kullanılabilirlik (HA) ve olağanüstü durum kurtarma (DR) çözümlerini kolaylaştırmak için IBM MQ otomatik istemci yeniden bağlantısını kullanabilirsiniz.

Çeşitli yüksek düzeyde kullanılabilirlik ve olağanüstü durum kurtarma çözümleri farklı platformlarda mevcuttur:

- Multi Çok eşgörünümlü kuyruk yöneticileri, farklı sunucularda yapılandırılmış aynı kuyruk yöneticisinin eşgörünümleridir (bkz. Çok eşgörünümlü kuyruk yöneticileri). Kuyruk yöneticisinin bir yönetim ortamı etkin yönetim ortamı olarak tanımlandı ve başka bir yönetim ortamı yedek yönetim ortamı olarak tanımlandı. Etkin yönetim ortamı başarısız olursa, çok eşgörünümlü kuyruk yöneticisi yedek sunucuda otomatik olarak yeniden başlatılır.

  Hem etkin hem de yedek kuyruk yöneticilerinin kuyruk yöneticisi tanıtıcısı (QMID) aynı. Çok eşgörünümlü bir kuyruk yöneticisine bağlanan IBM MQ istemci uygulamaları, otomatik istemci yeniden bağlantısı kullanılarak bir kuyruk yöneticisinin yedek yönetim ortamına otomatik olarak yeniden bağlanacak şekilde yapılandırılabilir.
- Linux RDQM (eşlenmiş veri kuyruğu yöneticisi), Linux platformlarında kullanılabilen yüksek kullanılabilirlikli bir çözümdür (bkz. RDQM yüksek kullanılabilirlik). RDQM yapılandırması, her biri kuyruk yöneticisinin eşgörünümüne sahip bir yüksek kullanılabilirlik (HA) grubunda yapılandırılan üç sunucudan oluşur. Bir yönetim ortamı, verilerini diğer iki yönetim ortamına zamanuyumlu olarak eşleyen, çalışmakta olan kuyruk yöneticisidir. Bu kuyruk yöneticisini çalıştıran sunucu başarısız olursa, kuyruk yöneticisinin başka bir eşgörünümü başlatılır ve üzerinde çalışılmakta olan veriler vardır. Kuyruk yöneticisinin üç eşgörünümü bir kayan IP adresini paylaşır, bu nedenle istemcilerin yalnızca tek bir IP adresiyle yapılandırılması gerekir. Bir RDQM kuyruk yöneticisine bağlanan istemci uygulamaları, otomatik istemci yeniden bağlantısı kullanılarak bir kuyruk yöneticisinin yedek eşgörünümüne otomatik olarak yeniden bağlanacak şekilde yapılandırılabilir.
- MQ Appliance HA çözümü, bir çift IBM MQ Aracı tarafından da sağlanabilir (IBM MQ Appliance belgelerinde High Availability ve Disaster Recovery başlıklı konuya bakın). Bir HA kuyruk yöneticisi, verileri diğer aygıttaki kuyruk yöneticisinin yedek eşgörünümüne zamanuyumlu olarak eşlerken aygıtlardan birinde çalışır. Birincil aygıt başarısız olursa, kuyruk yöneticisi otomatik olarak başlatılır ve diğer araçta çalışır. Kuyruk yöneticisinin iki örneği, kayan bir IP adresini paylaşacak şekilde yapılandırılabilir; bu nedenle, istemcilerin yalnızca tek bir IP adresiyle yapılandırılması gerekir. Bir IBM MQ Appliance ürünündeki HA kuyruk yöneticisine bağlanan istemci uygulamaları, otomatik istemci yeniden bağlantısı kullanılarak bir kuyruk yöneticisinin yedek eşgörünümüne otomatik olarak yeniden bağlanacak şekilde yapılandırılabilir.

Not: WebSphere Application Servergibi Java EE ortamlarında, IBM MQ classes for JMS tarafından sağlanan işlevselliği kullanarak etkinleştirme belirtimleriyle otomatik istemci yeniden bağlantısı desteklenmez. IBM MQ kaynak bağdaştırıcısı, etkinleştirme belirtiminin bağlandığı kuyruk yöneticisi kullanılamaz duruma gelirse, etkinleştirme belirtimlerini yeniden bağlamak için kendi mekanizmasını sağlar. Daha fazla bilgi için bkz “Java EE ortamlarında otomatik istemci yeniden bağlantısı desteği” sayfa 281.

İlgili kavramlar

Çok eşgörünümlü kuyruk yöneticileri

Otomatik istemci yeniden bağlantısı

İlgili başvurular

rdqm yüksek kullanılabilirlik

Java SE ortamlarında otomatik istemci yeniden bağlantısının kullanılması

Java SE ortamlarında çalışan IBM MQ classes for JMS kullanan uygulamalar, **CLIENTRECONNECTOPTIONS**bağlantı üreticisi özelliği aracılığıyla otomatik istemci yeniden bağlantısı işlevini kullanabilir.

CLIENTRECONNECTOPTIONS bağlantı üreticisi özelliği, kuyruk yöneticisinin çalıştığı sunucuya nasıl bağlanacağınızı belirlemek için iki ek bağlantı üreticisi özelliğini (**CONNECTIONNAMELIST** ve **CCDTURL**) kullanır.

CONNECTIONNAMELIST özellik

CONNECTIONNAMELIST özelliği, istemci kipinde bir kuyruk yöneticisine bağlanmak için kullanılacak anasistem adını ve kapı bilgilerini içeren virgülle ayrılmış bir listedir. Bu özellik, **QMANAGER** ve **CHANNEL** değerleriyle kullanılır. Bir uygulama istemci bağlantısı yaratmak için **CONNECTIONNAMELIST** özelliğini kullandığında, IBM MQ classes for JMS her anasisteme liste sırasında bağlanmayı dener. İlk kuyruk yöneticisi anasistemi kullanılamıyorsa, IBM MQ classes for JMS listede sonraki anasisteme bağlanmayı dener. Bağlantı yaratmadan bağlantı adı listesinin sonuna ulaşılırsa, IBM MQ classes for JMS `MQRC_QMGR_NOT_AVAILABLE` IBM MQ neden kodunu atar.

Uygulamanın bağlı olduğu kuyruk yöneticisi başarısız olursa, o kuyruk yöneticisine bağlanmak için **CONNECTIONNAMELIST** kullanan uygulamalar, kuyruk yöneticisinin kullanılamadığını belirten bir kural dışı durum alır. Uygulamanın kural dışı durumu yakalaması ve kullandığı kaynakları temizlemesi gerekir. Bağlantı yaratmak için uygulamanın bağlantı üreticisini kullanması gerekir. Bağlantı üreticisi her anasisteme liste sırasıyla bağlanmayı yeniden denerse, başarısız olan kuyruk yöneticisi artık kullanılamaz. Bağlantı üreticisi listedeki başka bir anasisteme bağlanmayı dener.

CCDTURL özellik

CCDTURL özelliği, bir CCDT ' yi (Client Channel Definition Table; İstemci Kanal Tanımlama Çizelgesi) gösteren bir Uniform Resource Locator (URL) içerir; bu özellik **QMANAGER** özelliğiyle kullanılır. CCDT, IBM MQ sisteminde tanımlı bir kuyruk yöneticisine bağlanmak için kullanılan istemci kanallarının bir listesini içerir. CCDT ' lerin IBM MQ classes for JMStarafından nasıl kullanıldığına ilişkin bilgi için bkz. “IBM MQ classes for JMS ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 273.

IBM MQ classes for JMS içinde otomatik istemci yeniden bağlantısını etkinleştirmek için CLIENTRECONNECTOPTIONS özelliğinin kullanılması

CLIENTRECONNECTOPTIONS özelliği, IBM MQ classes for JMSiçinde otomatik istemci yeniden bağlantısını etkinleştirmek için kullanılır. Bu özellik için olası değerler şunlardır:

ASDEF
Otomatik istemci yeniden bağlanma davranışı, IBM MQ istemci yapılanış kütüğünün (`mqclient.ini`) kanal kısmına ilişkin olarak belirtilen varsayılan değerle tanımlanır.

DEVRE DIŞI
Otomatik istemci yeniden bağlantısı devre dışı bırakıldı.

QMGR
IBM MQ classes for JMS , aşağıdaki seçeneklerden birini kullanarak, bağlı olduğu kuyruk yöneticisiyle aynı kuyruk yöneticisi tanıtıcısıyla bir kuyruk yöneticisine bağlanmayı dener:

- **CONNECTIONNAMELIST** özelliği ve **CHANNEL** özelliğinde tanımlanan kanal.
- **CCDTURL** özelliğinde tanımlanan CCDT.

Fark Etmez
IBM MQ classes for JMS , **CONNECTIONNAMELIST** özelliğini ya da **CCDTURL**özelliğini kullanarak aynı adı taşıyan bir kuyruk yöneticisine yeniden bağlanmayı dener.

İlgili bilgiler

İstemci yapılandırma dosyasının KANAL kısmı

Java EE ortamlarında otomatik istemci yeniden bağlantısının kullanılması
Java EE (Java Platform, Enterprise Edition) ortamlarına konuşlandırılabilecek IBM MQ kaynak bağdaştırıcısı ve WebSphere Application Server IBM MQ ileti alışverişi sağlayıcısı, IBM MQ kuyruk yöneticileriyle iletişim kurmak için IBM MQ classes for JMS olanağını kullanır. IBM MQ kaynak bağdaştırıcısı ve WebSphere Application Server IBM MQ ileti alışverişi sağlayıcısı, bir kuyruk yöneticisine otomatik olarak yeniden bağlanmak için istemci kapları içinde çalışan WebSphere Application Server dinleyici kapıları ve uygulamaları etkinleştirme belirtimlerine izin verecek çeşitli mekanizmalar sağlar. Kurumsal JavaBeans (EJB) ve web tabanlı uygulamaların kendi yeniden bağlanma mantığını gerçekleştirmeleri gerekir.

Not: IBM MQ classes for JMS tarafından sağlanan işlevsellik kullanılarak etkinleştirme belirtimleriyle otomatik istemci yeniden bağlantısı desteklenmez (bkz. “Otomatik JMS istemcisi yeniden bağlantısı” sayfa 274). IBM MQ kaynak bağdaştırıcısı, etkinleştirme belirtiminin bağlandığı kuyruk yöneticisi kullanılamaz duruma gelirse, etkinleştirme belirtimlerini yeniden bağlamak için kendi mekanizmasını sağlar.

Kaynak bağdaştırıcısının sağladığı mekanizma aşağıdakiler tarafından denetlenir:

- IBM MQ Kaynak bağdaştırıcısı özelliği **reconnectionRetryCount**.
- IBM MQ Kaynak bağdaştırıcısı özelliği **reconnectionRetryInterval**.
- Etkinleştirme belirtimi özelliği **connectionNameList**.

Bu özelliklerle ilgili daha fazla bilgi için bkz. “ResourceAdapter nesne özellikleri için yapılandırma” sayfa 433.

İletiyle yönlendirilen bir Bean uygulamasının onMessage() yönteminde ya da Java Platform, Enterprise Edition ortamında çalışan başka bir uygulamada otomatik istemci yeniden bağlantısının kullanılması desteklenmez. Uygulamanın bağlandığı kuyruk yöneticisi kullanılamaz duruma gelirse, uygulamanın kendi yeniden bağlanma mantığını gerçekleştirmesi gerekir. Daha fazla bilgi için bkz “Java EE uygulamasında yeniden bağlantı mantığını uygulama” sayfa 288.

Java EE ortamlarında otomatik istemci yeniden bağlantısı desteği
Java EE ortamlarında (WebSphere Application Server, IBM MQ kaynak bağdaştırıcısı ve WebSphere Application Server IBM MQ ileti alışverişi sağlayıcısı gibi), uygulamaların bir kuyruk yöneticisine otomatik olarak yeniden bağlanmasına izin veren çeşitli mekanizmalar sağlar. Ancak, bazı durumlarda, bu destek için kısıtlamalar geçerlidir.

Java EE ortamlarına ve WebSphere Application Server IBM MQ ileti alışverişi sağlayıcısına konuşlandırılabilecek IBM MQ kaynak bağdaştırıcısı, IBM MQ kuyruk yöneticileriyle iletişim kurmak için IBM MQ classes for JMS kullanın.

Aşağıdaki çizelge, IBM MQ kaynak bağdaştırıcısının ve WebSphere Application Server IBM MQ ileti alışverişi sağlayıcısının otomatik istemci yeniden bağlantısı için destek sağladığı desteği özetler.

Çizelge 45. Java EE ortamlarındaki otomatik istemci yeniden bağlanma seçenekleri için destek özeti

Otomatik yeniden bağlanma seçenekleri	**CONNECTIONNAMELIST öZELLIğI**	**CCDTURL özelliği**	**CLIENTRECONNECTOPTIONS özelliği**	**Otomatik istemci yeniden bağlantısına alternatif yaklaşım**
Etkinleştirme belirtimleri	Kısıtlamalarla desteklenir	Kısıtlamalarla desteklenir	Desteklenmiyor	Java EE ortam ve etkinleştirme belirtimleri kendi yeniden bağlantı mekanizmalarını sağlar
WebSphere Application Server dinleyici kapıları	Kısıtlamalarla desteklenir	Kısıtlamalarla desteklenir	Desteklenmiyor	WebSphere Application Server , kendi yeniden bağlantı mekanizmasını sağlar
Kurumsal JavaBeans ve web tabanlı uygulamalar	Kısıtlamalarla desteklenir	Kısıtlamalarla desteklenir	Desteklenmiyor	Uygulama kendi yeniden bağlantı mantığını uygulamalıdır

Çizelge 45. Java EE ortamlarındaki otomatik istemci yeniden bağlanma seçenekleri için destek özeti (devamı var)				
Otomatik yeniden bağlanma seçenekleri	**CONNECTIONNAMELIST öZELLIğI**	**CCDTURL özelliği**	**CLIENTRECONNECTOPTIONS özelliği**	**Otomatik istemci yeniden bağlantısına alternatif yaklaşım**
İstemci taşıyıcıları içinde çalışan uygulamalar	Desteklenenler:	Desteklenenler:	Desteklenenler:	Geçerli değildir

IBM MQ classes for JMSgibi bir Java EE ortamında kurulu ileti odaklı Bean uygulamaları, IBM MQ sistemindeki iletileri işlemek için etkinleştirme belirtimlerini kullanabilir. Etkinleştirme belirtimleri, bir IBM MQ sistemine gelen iletileri saptamak ve işlenmek üzere ileti odaklı Bean 'lere teslim etmek için kullanılır. İleti odaklı Bean 'ler, **onMessage()** yöntemlerinin içinden IBM MQ sistemleriyle daha fazla bağlantı kurabilir. Bu bağlantıların otomatik istemci yeniden bağlantısını nasıl kullanabileceklerine ilişkin ek bilgi için Enterprise JavaBeans ve Web tabanlı uygulamalarbaşlıklı konuya bakın.

Etkinleştirme belirtimleri
Etkinleştirme belirtimleri için, **CONNECTIONNAMELIST** ve **CCDTURL** özellikleri kısıtlamalarla desteklenir ve **CLIENTRECONNECTOPTIONS** özelliği desteklenmez.

WebSphere Application Servergibi bir Java EE ortamında kurulu MDB uygulamaları, IBM MQ sistemindeki iletileri işlemek için etkinleştirme belirtimlerini kullanabilir.

Etkinleştirme belirtimleri, bir IBM MQ sistemine gelen iletileri algılamak ve işlenmek üzere MDB ' lere teslim etmek için kullanılır. Bu bölümde, etkinleştirme belirtiminin IBM MQ sistemini nasıl izlediği ele alınmıştır.

MDB ' ler ayrıca onMessage() yönteminin içinden IBM MQ sistemlerine ek bağlantılar da yapabilir.

Bu bağlantıların otomatik istemci yeniden bağlantısını nasıl kullanabileceğine ilişkin ayrıntıları “Kurumsal JavaBeans ve web tabanlı uygulamalar” sayfa 286adresinde bulabilirsiniz.

CONNECTIONNAMELIST özellik

Başlatma sırasında, etkinleştirme belirtimi aşağıdaki işlemleri kullanarak kuyruk yöneticisine bağlanmayı dener:

- **QMANAGER** özelliğinde belirtilen bir özellik
- **CHANNEL** özelliğinde belirtilen kanal
- **CONNECTIONNAMELIST** içindeki ilk girişten anasistem adı ve kapı bilgileri

Etkinleştirme belirtimi listedeki ilk girişi kullanarak kuyruk yöneticisine bağlanamazsa, etkinleştirme belirtimi ikinci girişe geçer ve kuyruk yöneticisiyle bağlantı kuruluncaya ya da listenin sonuna ulaşılıncaya kadar bu şekilde devam eder.

Etkinleştirme belirtimi belirtilen kuyruk yöneticisine **CONNECTIONNAMELIST**içindeki girişlerden herhangi birini kullanarak bağlanamıyorsa, etkinleştirme belirtimi durur ve yeniden başlatılması gerekir.

Etkinleştirme belirtimi çalıştıktan sonra, etkinleştirme belirtimi IBM MQ sisteminden iletileri alır ve iletileri işlenmek üzere bir MDB ' ye teslim eder.

Bir ileti işlenirken kuyruk yöneticisi başarısız olursa, Java EE ortamı hatayı saptar ve etkinleştirme belirtimini yeniden bağlamayı dener.

Etkinleştirme belirtimi, etkinleştirme belirtimi yeniden bağlanma girişimlerini gerçekleştirdiğinde, **CONNECTIONNAMELIST** özelliğindeki bilgileri daha önce olduğu gibi kullanır.

Etkinleştirme belirtimi **CONNECTIONNAMELIST** içindeki tüm girişleri denerse ve kuyruk yöneticisine hala bağlanamıyorsa, etkinleştirme belirtimi yeniden denemeden önce IBM MQ kaynak bağdaştırıcısı özelliği **reconnectionRetryInterval** tarafından belirtilen süre boyunca bekler.

IBM MQ Kaynak bağdaştırıcısı özelliği **reconnectionRetryCount** , bir etkinleştirme belirtimi durdurulmadan önce yapılacak ardışık yeniden bağlanma girişimlerinin sayısını tanımlar ve el ile yeniden başlatma gerektirir.

Etkinleştirme belirtimi bir IBM MQ sistemine yeniden bağlandıktan sonra, Java EE ortamı gerekli olan işlemsel temizlemeyi gerçekleştirir ve işlenmek üzere MDB ' lere ileti göndermeye devam eder.

Hareket temizleme işleminin doğru çalışması için, Java EE ortamının başarısız olan kuyruk yöneticisine ilişkin günlüklere erişebilmesi gerekir.

Etkinleştirme belirtimleri XA hareketlerine katılan işlemsel veritabanı yöneticileriyle kullanılıyorsa ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, **CONNECTIONNAMELIST** hem etkin hem de yedek kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

Bu, ortamın hareket kurtarma işlemi gerçekleştirmesi gerekiyorsa, ortamın bir hata sonrasında hangi kuyruk yöneticisine yeniden bağlandığına bakılmaksızın, Java EE ortamının kuyruk yöneticisi günlüklerine erişebileceği anlamına gelir.

Hareket işleme veritabanı yöneticileri bağımsız kuyruk yöneticileriyle kullanılıyorsa, etkinleştirme belirtiminin bir hata sonrasında aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlanmasını sağlamak için **CONNECTIONNAMELIST** özelliğinin tek bir giriş içermesi gerekir.

CCDTURL özellik

Başlatma sırasında, etkinleştirme belirtimi istemci kanal tanımlama çizelgesindeki (CCDT) ilk girişi kullanarak **QMANAGER** özelliğinde belirtilen kuyruk yöneticisine bağlanmayı dener.

Etkinleştirme belirtimi çizelgedeki ilk girişi kullanarak kuyruk yöneticisine bağlanamazsa, etkinleştirme belirtimi ikinci girişe geçer ve kuyruk yöneticisiyle bağlantı kuruluncaya ya da çizelgenin sonuna ulaşılıncaya kadar devam eder.

Etkinleştirme belirtimi belirtilen kuyruk yöneticisine CCDT ' deki girişlerden herhangi birini kullanarak bağlanamıyorsa, etkinleştirme belirtimi durur ve yeniden başlatılması gerekir.

Etkinleştirme belirtimi çalıştıktan sonra, etkinleştirme belirtimi IBM MQ sisteminden iletileri alır ve iletileri işlenmek üzere bir MDB ' ye teslim eder.

Bir ileti işlenirken kuyruk yöneticisi başarısız olursa, Java EE ortamı hatayı saptar ve etkinleştirme belirtimini yeniden bağlamayı dener.

Etkinleştirme belirtimi, etkinleştirme belirtimi yeniden bağlanma girişimlerini gerçekleştirdiğinde, CCDT özelliğindeki bilgileri daha önce olduğu gibi kullanır.

Etkinleştirme belirtimi CCDT ' deki tüm girişleri denerse ve kuyruk yöneticisine hala bağlanamıyorsa, etkinleştirme belirtimi yeniden denemeden önce IBM MQ kaynak bağdaştırıcısı özelliği **reconnectionRetryInterval** tarafından belirtilen süre boyunca bekler.

IBM MQ Kaynak bağdaştırıcısı özelliği **reconnectionRetryCount** , bir etkinleştirme belirtimi durdurulmadan önce yapılacak ardışık yeniden bağlanma girişimlerinin sayısını tanımlar ve el ile yeniden başlatma gerektirir.

Etkinleştirme belirtimi bir IBM MQ sistemine yeniden bağlandıktan sonra, Java EE ortamı gerekli olan işlemsel temizlemeyi gerçekleştirir ve işlenmek üzere MDB ' lere ileti göndermeye devam eder.

Hareket temizleme işleminin doğru çalışması için, Java EE ortamının başarısız olan kuyruk yöneticisine ilişkin günlüklere erişebilmesi gerekir.

Etkinleştirme belirtimleri XA hareketlerine katılan işlemsel MDB 'lerle kullanılıyorsa ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, CCDT' nin hem etkin hem de yedek kuyruk yöneticisi yönetim ortamı için bir giriş içermesi gerekir.

Bu, ortamın hareket kurtarma işlemi gerçekleştirmesi gerekiyorsa, ortamın bir hata sonrasında hangi kuyruk yöneticisine yeniden bağlandığına bakılmaksızın, Java EE ortamının kuyruk yöneticisi günlüklerine erişebileceği anlamına gelir.

Hareket işlemsel MDB ' ler bağımsız kuyruk yöneticileriyle kullanılıyorsa, etkinleştirme belirtiminin bir başarısızlığın ardından aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlanmasını sağlamak için CCDT tek bir giriş içermelidir.

Etkinleştirme belirtimleriyle birlikte kullanılan CCDT ' lerdeki **AFFINITY** özelliği için varsayılan *PREFERRED* değerini ayarladığınızdan emin olun; böylece bağlantılar aynı etkin kuyruk yöneticisine yapılır.

CLIENTRECONNECTOPTIONS özellik

Etkinleştirme belirtimleri, kendi yeniden bağlantı işlevlerini sağlar. Sağlanan işlevsellik, bağlı oldukları kuyruk yöneticisi başarısız olursa belirtimlerin bir IBM MQ sistemine otomatik olarak yeniden bağlanmasını sağlar.

Bu nedenle, IBM MQ classes for JMS tarafından sağlanan otomatik istemci yeniden bağlantısı işlevselliği desteklenmez.

Java EEiçinde kullanılan tüm etkinleştirme belirtimleri için **CLIENTRECONNECTOPTIONS** özelliğini *DEVRE Dışı* olarak ayarlamanız gerekir.

WebSphere Application Server dinleyici kapıları
WebSphere Application Server ' e kurulan ileti odaklı Bean (MDB) uygulamaları, IBM MQ sistemindeki iletileri işlemek için dinleyici kapılarını da kullanabilir.

Dinleyici kapıları, bir IBM MQ sistemine gelen iletileri algılamak ve işlenmek üzere MDB ' lere teslim etmek için kullanılır. Bu konuda, dinleyici kapısının IBM MQ sistemini nasıl izlediği açıklanmaktadır.

MDB ' ler ayrıca onMessage() yönteminin içinden IBM MQ sistemlerine ek bağlantılar da yapabilir.

Bu bağlantıların otomatik istemci yeniden bağlantısını nasıl kullanabileceğine ilişkin ek bilgi için bkz. “Kurumsal JavaBeans ve web tabanlı uygulamalar” sayfa 286

WebSphere Application Server dinleyici kapıları için:

- **CONNECTIONNAMELIST** ve **CCDTURL** kısıtlamalar ile desteklenir
- **CLIENTRECONNECTOPTIONS** desteklenmiyor

CONNECTIONNAMELIST özellik

Dinleyici kapıları, IBM MQ' e bağlanırken JMS bağlantı havuzlarından yararlanır; bu nedenle, bağlantı havuzlarının kullanılmasının etkileri söz konusu olabilir. Daha fazla bilgi için bkz. “Etkinleştirme belirtimleri” sayfa 282 .

Serbest bağlantı yoksa ve bu bağlantı üreticisinden henüz bağlantı sayısı üst sınırı oluşturulmamışsa, IBM MQile yeni bir bağlantı oluşturmak için **CONNECTIONNAMELIST** özelliği kullanılır.

CONNECTIONNAMELIST içindeki tüm IBM MQ sistemlerine erişilemiyorsa, dinleyici kapısı durur.

Daha sonra dinleyici kapısı, **RECOVERY.RETRY.INTERVAL** ileti dinleyici hizmeti özel özelliği tarafından belirtilen süreyi bekler ve yeniden bağlanmayı dener.

Bu yeniden bağlanma girişimi, bağlantı havuzunda boş bağlantı olup olmadığını denetler; bağlantı girişimleri arasında bir bağlantı döndürülürse. Dinleyici kapısı kullanılamıyorsa, daha önce olduğu gibi **CONNECTIONNAMELIST** ' i kullanır.

Dinleyici kapısı bir IBM MQ sistemine yeniden bağlandıktan sonra, Java EE ortamı gerekli olan işlemsel temizlemeyi gerçekleştirir ve ardından işlenmek üzere MDB ' lere ileti göndermeye devam eder.

Hareket temizleme işleminin doğru çalışması için, Java EE ortamının başarısız olan kuyruk yöneticisine ilişkin günlüklere erişebilmesi gerekir.

Dinleyici kapıları, XA hareketlerine katılan işlemsel MDB ' lerle kullanılıyorsa ve bir **çok eşgörünümlü kuyruk yöneticisine**bağlanıyorsa, **CONNECTIONNAMELIST** hem etkin hem de yedek kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

Bu, ortamın hareket kurtarma işlemi gerçekleştirmesi gerekiyorsa, ortamın bir hata sonrasında hangi kuyruk yöneticisine yeniden bağlandığına bakılmaksızın, Java EE ortamının kuyruk yöneticisi günlüklerine erişebileceği anlamına gelir.

Hareket işleme veritabanı yöneticileri bağımsız kuyruk yöneticileriyle kullanılıyorsa, etkinleştirme belirtiminin bir hata sonrasında aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlanmasını sağlamak için **CONNECTIONNAMELIST** özelliğinin tek bir giriş içermesi gerekir.

CCDTURL özellik

Başlatılırken, dinleyici kapısı CCDT ' deki ilk girişi kullanarak **QMANAGER** özelliğinde belirtilen kuyruk yöneticisine bağlanmayı dener.

Dinleyici kapısı, çizelgedeki ilk girişi kullanarak kuyruk yöneticisine bağlanamıyorsa, dinleyici kapısı ikinci girişe geçer ve böylece, kuyruk yöneticisiyle bağlantı kuruluncaya ya da çizelgenin sonuna ulaşılıncaya kadar devam eder.

Dinleyici kapısı CCDT ' deki girişlerden herhangi birini kullanarak belirtilen kuyruk yöneticisine bağlanamazsa, dinleyici kapısı durur.

Daha sonra dinleyici kapısı, **RECOVERY.RETRY.INTERVAL** ileti dinleyici hizmeti özel özelliği tarafından belirtilen süreyi bekler ve yeniden bağlanmayı dener.

Bu yeniden bağlanma girişimi, CCDT ' deki tüm girişlerde önceki gibi çalışır.

Dinleyici Kapısı çalıştıktan sonra, iletileri IBM MQ sisteminden alır ve işlenmek üzere bir MDB ' ye teslim eder.

Bir ileti işlenirken kuyruk yöneticisi başarısız olursa, Java EE ortamı hatayı saptar ve dinleyici kapısını yeniden bağlamayı dener. Dinleyici kapısı, yeniden bağlanma girişimlerini gerçekleştirirken CCDT ' deki bilgileri kullanır.

Dinleyici kapısı CCDT ' deki tüm girişleri denerse ve kuyruk yöneticisine hala bağlanamıyorsa, kapı yeniden denemeden önce **RECOVERY.RETRY.INTERVAL** özelliğinin belirlediği süreyi bekler.

MAX.RECOVERY.RETRIES ileti dinleyici hizmeti özelliği, bir dinleyici kapısı durmadan ve el ile yeniden başlatma gerektirmeden önce yapılan ardışık yeniden bağlanma girişimlerinin sayısını tanımlar.

Dinleyici kapısı bir IBM MQ sistemine yeniden bağlandıktan sonra, Java EE ortamı gerekli olan işlemsel temizlemeyi gerçekleştirir ve ardından işlenmek üzere MDB ' lere ileti göndermeye devam eder.

Hareket temizleme işleminin doğru çalışması için, Java EE ortamının başarısız olan kuyruk yöneticisine ilişkin günlüklere erişebilmesi gerekir.

Dinleyici kapıları, XA hareketlerine katılan işlemsel MDB 'lerle kullanılıyorsa ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, CCDT' nin hem etkin hem de yedek kuyruk yöneticisi yönetim ortamı için bir giriş içermesi gerekir.

Bu, ortamın hareket kurtarma işlemi gerçekleştirmesi gerekiyorsa, ortamın bir hata sonrasında hangi kuyruk yöneticisine yeniden bağlandığına bakılmaksızın, Java EE ortamının kuyruk yöneticisi günlüklerine erişebileceği anlamına gelir.

Hareket işleme MDB ' leri bağımsız kuyruk yöneticileriyle kullanılıyorsa, dinleyici kapısının bir başarısızlıktan sonra aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlanmasını sağlamak için CCDT tek bir giriş içermelidir.

Aynı etkin kuyruk yöneticisiyle bağlantı kurulabilmesi için, dinleyici kapılarıyla kullanılan CCDT ' lerdeki **AFFINITY** özelliği için varsayılan *PREFERRED* değerini ayarladığınızdan emin olun.

CLIENTRECONNECTOPTIONS özellik

Dinleyici kapıları kendi yeniden bağlanma işlevlerini sağlar. Sağlanan işlev, bağlı oldukları kuyruk yöneticisi başarısız olursa, dinleyici kapılarının bir IBM MQ sistemine otomatik olarak yeniden bağlanmasını sağlar.

Bu nedenle, IBM MQ classes for JMS tarafından sağlanan otomatik istemci yeniden bağlantısı işlevselliği desteklenmez.

CLIENTRECONNECTOPTIONS özelliğini, Java EEiçinde kullanılan tüm dinleyici kapıları için *DEVRE Dışı* olarak ayarlamanız gerekir.

Kurumsal JavaBeans ve web tabanlı uygulamalar
Kurumsal JavaBean (EJB) uygulamaları ve Web taşıyıcısı içinde çalışan uygulamalar (örneğin, Sunucu Uygulamaları), bir IBM MQ kuyruk yöneticisine bağlantı yaratmak için JMS bağlantı üreticisini kullanır.

EJB ' ler ve Web tabanlı uygulamalar için aşağıdaki kısıtlamalar geçerlidir:

- **CONNECTIONNAMELIST** ve **CCDTURL** kısıtlamalar ile desteklenir
- **CLIENTRECONNECTOPTIONS** desteklenmiyor

CONNECTIONNAMELIST özellik

Java EE ortamı JMS bağlantıları için bir bağlantı havuzu sağlarsa, bunun **CONNECTIONNAMELIST** özelliğinin davranışını nasıl etkilediğine ilişkin bilgi için bkz. “Bağlantı havuzunda CONNECTIONNAMELIST ya da CCDT kullanılması” sayfa 287 .

Java EE ortamı bir JMS bağlantı havuzu sağlamazsa. Uygulama, **CONNECTIONNAMELIST** özelliğini Java SE uygulamalarıyla aynı şekilde kullanır.

Uygulamalar XA hareketlerine katılan işlemsel veritabanı yöneticileriyle kullanılıyorsa ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, **CONNECTIONNAMELIST** hem etkin hem de yedek kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

Bu, ortamın hareket kurtarma işlemi gerçekleştirmesi gerekiyorsa, ortamın bir hata sonrasında hangi kuyruk yöneticisine yeniden bağlandığına bakılmaksızın, Java EE ortamının kuyruk yöneticisi günlüklerine erişebileceği anlamına gelir.

Uygulamalar bağımsız kuyruk yöneticileriyle kullanılıyorsa, uygulamanın aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlanmasını sağlamak için **CONNECTIONNAMELIST** özelliğinin tek bir giriş içermesi gerekir.

CCDTURL özellik

Java EE ortamı JMS bağlantıları için bir bağlantı havuzu sağlarsa, bunun **CCDTURL** özelliğinin davranışını nasıl etkilediğine ilişkin bilgi için bkz. “Bağlantı havuzunda CONNECTIONNAMELIST ya da CCDT kullanılması” sayfa 287 .

Java EE ortamı bir JMS bağlantı havuzu sağlamazsa. Uygulama, **CCDTURL** özelliğini Java SE uygulamalarıyla aynı şekilde kullanır.

Uygulamalar XA hareketlerine katılan işlemsel veritabanı yöneticileriyle kullanılıyorsa ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, CCDT hem etkin hem de yedek kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

Bu, ortamın hareket kurtarma işlemi gerçekleştirmesi gerekiyorsa, ortamın bir hata sonrasında hangi kuyruk yöneticisine yeniden bağlandığına bakılmaksızın, Java EE ortamının kuyruk yöneticisi günlüklerine erişebileceği anlamına gelir.

Uygulamalar bağımsız kuyruk yöneticileriyle kullanılıyorsa, etkinleştirme belirtiminin bir hata sonrasında aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlandığından emin olmak için CCDT tek bir giriş içermelidir.

CLIENTRECONNECTOPTIONS özellik

Web taşıyıcısında çalışan EJB ' ler ya da uygulamalar tarafından kullanılan tüm JMS bağlantı üreticileri için **CLIENTRECONNECTOPTIONS** özelliğini *DISABLED* olarak ayarlamanız gerekir.

Yeni bir kuyruk yöneticisine otomatik olarak yeniden bağlanması gereken uygulamalar, kullandıkları kuyruk yöneticisi başarısız olursa, kendi yeniden bağlantı mantığını gerçekleştirmeleri gerekir. Ek bilgi için bkz. “Java EE uygulamasında yeniden bağlantı mantığını uygulama” sayfa 288 .

Senaryolar: WebSphere Application Server with IBM MQ
Senaryolar: IBM MQ ile WebSphere Application Server Liberty profili

İstemci taşıyıcıları içinde çalışan uygulamalar
WebSphere Application Servergibi bazı Java EE ortamları Java SE uygulamalarını çalıştırmak için kullanılabilecek bir istemci taşıyıcısı sağlar.

Bu ortamların içinde çalışan uygulamalar, IBM MQ kuyruk yöneticisine bağlanmak için bir JMS bağlantı üreticisi kullanır.

İstemci taşıyıcıları içinde çalışan uygulamalar için:

- **CONNECTIONNAMELIST** ve **CCDTURL** tam olarak desteklenir
- **CLIENTRECONNECTOPTIONS** tam olarak desteklenir

CONNECTIONNAMELIST özellik

Java EE ortamı JMS bağlantıları için bir bağlantı havuzu sağlarsa, bunun **CONNECTIONNAMELIST** özelliğinin davranışını nasıl etkilediğine ilişkin bilgi için bkz. “Bağlantı havuzunda CONNECTIONNAMELIST ya da CCDT kullanılması” sayfa 287 .

Java EE ortamı bir JMS bağlantı havuzu sağlamazsa. Uygulama, **CONNECTIONNAMELIST** özelliğini Java SE uygulamalarıyla aynı şekilde kullanır.

CCDTURL özellik

Java EE ortamı JMS bağlantıları için bir bağlantı havuzu sağlarsa, bunun **CCDTURL** özelliğinin davranışını nasıl etkilediğine ilişkin bilgi için bkz. “Bağlantı havuzunda CONNECTIONNAMELIST ya da CCDT kullanılması” sayfa 287 .

Java EE ortamı bir JMS bağlantı havuzu sağlamazsa. Uygulama, **CCDTURL** özelliğini Java SE uygulamalarıyla aynı şekilde kullanır.

Bağlantı havuzunda CONNECTIONNAMELIST ya da CCDT kullanılması
Bazı Java EE ortamları (örneğin, WebSphere Application Server) bir JMS bağlantı havuzu sağlar. Java SE uygulamalarını çalıştırmak için kullanılabilecek taşıyıcı.

Java EE ortamında tanımlanmış bir bağlantı üreticisini kullanarak bağlantı yaratan uygulamalar, bu bağlantı üreticisine ilişkin bağlantı havuzundan varolan bir boş bağlantıyı ya da bağlantı havuzunda uygun bir bağlantı yoksa yeni bir bağlantı elde edin.

Bağlantı üreticisi **CONNECTIONNAMELIST** ya da **CCDTURL** özelliğiyle yapılandırıldıysa, bunun bazı etkileri olabilir.

Bağlantı oluşturmak için bağlantı üreticisi ilk kez kullanıldığında, Java EE ortamı **CONNECTIONNAMELIST**ürününü kullanır. ya da IBM MQ sistemine yeni bir bağlantı yaratmak için **CCDTURL** adresini kullanın. Bu bağlantı artık gerekli olmadığında, bağlantının yeniden kullanılabilir hale geldiği bağlantı havuzuna döndürülür.

Bağlantı üreticisinden başka bir bağlantı yaratılırsa, Java EE ortamı bağlantıyı yeni bir bağlantı yaratmak için **CONNECTIONNAMELIST** ya da **CCDTURL** özelliklerini kullanmak yerine bağlantı havuzundan döndürür.

Bir kuyruk yöneticisi yönetim ortamı başarısız olduğunda bağlantı kullanılıyorsa, bağlantı atılır. Ancak, bağlantı havuzunun içeriği olmayabilir; bu, havuzun artık çalışmayan bir kuyruk yöneticisine yönelik bağlantılar içerebileceği anlamına gelir.

Bu durumda, bağlantı üreticisinden bağlantı yaratmak için bir sonraki istekte bulunulduğunda, başarısız olan kuyruk yöneticisiyle bağlantı döndürülür. Kuyruk yöneticisi artık çalışmadığı için bu bağlantıyı kullanma girişimleri başarısız olur ve bağlantının atılmasına neden olur.

Yalnızca bağlantı havuzu boşsa, Java EE ortamı IBM MQile yeni bir bağlantı yaratmak için **CONNECTIONNAMELIST** ya da **CCDTURL** özelliklerini kullanır.

CONNECTIONNAMELIST ve CCDT ' lerin JMS bağlantıları yaratmak için kullanılma şekli nedeniyle, farklı IBM MQ sistemlerine bağlantı içeren bir bağlantı havuzu da olabilir.

Örneğin, **CONNECTIONNAMELIST** özelliği aşağıdaki değere ayarlanmış olarak bir bağlantı üreticisinin yapılandırıldığını varsayın:

```
CONNECTIONNAMELIST = hostname1(port1), hostname2(port2)
```

Bir uygulamanın bu bağlantı üreticisinden bağımsız bir kuyruk yöneticisine ilk kez bağlantı yaratmaya çalıştığında, sistemde çalışan kuyruk yöneticisine `hostname1(port1)` erişilemediğini varsayın. Bu, uygulamanın `hostname2(port2)` üzerinde çalışan kuyruk yöneticisiyle bir bağlantıyla sona ereceği anlamına gelir.

Şimdi başka bir uygulama geliyor ve aynı bağlantı üreticisinden bir JMS bağlantısı oluşturuyor. `hostname1(port1)` üzerindeki kuyruk yöneticisi artık kullanılabilir, bu nedenle bu IBM MQ sistemine yeni bir JMS bağlantısı yaratılır ve uygulamaya döndürülür.

Her iki uygulama da tamamlandığında, bağlantıların bağlantı havuzuna geri dönmesine neden olan JMS Connections uygulamalarını kapatır.

Sonuç olarak, bağlantı üreticimize ilişkin bağlantı havuzu şu anda iki JMS bağlantısı içerir:

- `hostname1(port1)` üzerinde çalışan kuyruk yöneticisine bir bağlantı
- `hostname2(port2)` üzerinde çalışan kuyruk yöneticisine bir bağlantı

Bu, işlem kurtarmayla ilgili sorunlara yol açabilir. Java EE sisteminin bir hareketi geriye işlemesi gerekiyorsa, hareket günlüklerine erişimi olan bir kuyruk yöneticisine bağlanması gerekir.

Java EE uygulamasında yeniden bağlantı mantığını uygulama
Bir kuyruk yöneticisinin başarısız olması durumunda otomatik olarak yeniden bağlanmak isteyen kurumsal JavaBeans ve web tabanlı uygulamalar kendi yeniden bağlantı mantığını uygulayabilir.

Aşağıdaki seçenekler, bunu nasıl başarabileceğinize ilişkin daha fazla bilgi sağlar.

Uygulamanın başarısız olmasına izin ver

Bu yaklaşım, uygulama değişikliği gerektirmez, ancak **CONNECTIONNAMELIST** özelliğini içermek için bağlantı üreticisi tanımının yönetimle yeniden yapılandırılmasını gerektirir. Ancak bu yaklaşım, çağıranın bir hatayı uygun şekilde işleyebilmesini gerektirir. Bunun, bağlantı hatasıyla ilgili olmayan MQRC_Q_FULL gibi hatalar için de gerekli olduğunu unutmayın.

Bu işleme ilişkin örnek kod:

```
public class SimpleServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
         try {
 // get connection factory/ queue
 InitialContext ic = new InitialContext();
 ConnectionFactory cf =
           (ConnectionFactory)ic.lookup("java:comp/env/jms/WMQCF");
 Queue q = (Queue) ic.lookup("java:comp/env/jms/WMQQueue");

 // send a message
 Connection c = cf.createConnection();
 Session s = c.createSession(false, Session.AUTO_ACKNOWLEDGE);
 MessageProducer p = s.createProducer(q);
 Message m = s.createTextMessage();
```

```
  p.send(m);

  // done, release the connection
  c.close();
  }
  catch (JMSException je) {
  // process exception
    }
  }
}
```

Önceki kod, bu sunucu uygulamacığının kullandığı bağlantı üreticisinin **CONNECTIONNAMELIST** özelliğini tanımladığını varsayar.

Sunucu uygulaması ilk kez işlem yaptığında, **CONNECTIONNAMELIST** özelliği kullanılarak, aynı kuyruk yöneticisine bağlanan diğer uygulamalarda havuza yollanmış bağlantı olmadığı varsayılarak yeni bir bağlantı yaratılır.

Bir `close()` çağrısıyla bağlantı serbest bırakıldığında, bu bağlantı havuza döndürülür ve sunucu uygulamacığının bir sonraki çalışmasında (**CONNECTIONNAMELIST** öğesine başvurmadan), bağlantı hatası oluşuncaya kadar yeniden kullanılır; bu noktada bir CONNECTION_ERROR_OCCURRED olayı oluşturulur. Bu olay, havuzun başarısız olan bağlantıyı yok etmesini ister.

Uygulama bir sonraki çalışmasında, havuza yollanmış bağlantı yoktur ve kullanılabilir ilk kuyruk yöneticisine bağlanmak için **CONNECTIONNAMELIST** kullanılır. Kuyruk yöneticisi hata durumunda yedek sisteme geçiş gerçekleştiyse (örneğin, başarısızlık geçici bir ağ hatası değildi), sunucu uygulaması kullanılabilir olduğunda yedek eşgörünüme bağlanır.

Uygulamada veritabanları gibi diğer kaynaklar varsa, uygulama sunucusunun hareketi geriye işlemesi gerektiğini belirtmesi uygun olabilir.

Uygulama içindeki yeniden bağlantıyı işle

Çağıran sunucu uygulamacığından gelen bir hatayı işleyemezse, yeniden bağlantı uygulama içinde işlenmelidir. Aşağıdaki örnekte gösterildiği gibi, uygulama içindeki bir yeniden bağlantıyı işlemek için uygulamanın JNDI 'den aradığı bağlantı üreticisini önbelleğe alabilmesi ve JMSCMQ0001: `WebSphere MQ çağrısı' 2 '('MQCC_FAILED') neden '2009' ('MQRC_CONNECTION_BROKEN ')`gibi bir `JMSException` işleyebilmesi için yeni bir bağlantı istemesi gerekir.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

  // get connection factory/ queue
  InitialContext ic = new InitialContext();
  ConnectionFactory cf = (ConnectionFactory)
              ic.lookup("java:comp/env/jms/WMQCF");
  Destination destination = (Destination) ic.lookup("java:comp/env/jms/WMQQueue");

  setupResources();

  // loop sending messages
  while (!sendComplete) {
    try {
      // create the next message to send
      msg.setText("message sent at "+new Date());
      // and send it
      producer.send(msg);
    }
    catch (JMSException je) {
        // drive reconnection
        setupResources();
    }
  }
```

Aşağıdaki örnekte `setupResources()` , JMS nesnelerini oluşturur ve anlık olmayan yeniden bağlantıyı işlemek için bir uyku ve yeniden deneme döngüsü içerir. Uygulamada bu yöntem birçok yeniden bağlanma girişimlerini önler. Çıkış koşullarının açıklık için örnekten çıkarıldığını unutmayın.

```
 private void setupResources() {
```

```
    boolean connected = false;
    while (!connected) {
      try {
        connection = cf.createConnection(); // cf cached from JNDI lookup
        session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
        msg = session.createTextMessage();
        producer = session.createProducer(destination); // destination cached from JNDI lookup
        // no exception? then we connected ok
        connected = true;
      }
      catch (JMSException je) {
        // sleep and then have another attempt
        try {Thread.sleep(30*1000);} catch (InterruptedException ie) {}
      }
    }
```

Uygulama yeniden bağlantıyı yönetiyorsa, uygulamanın diğer kaynaklara tutulan bağlantıları serbest bırakması önemlidir (bu kaynaklar diğer IBM MQ kuyruk yöneticileri ya da veritabanları gibi diğer arka uç hizmetleri olabilir). Yeni bir IBM MQ kuyruk yöneticisi yönetim ortamına yeniden bağlanma işlemi tamamlandığında bu bağlantıları yeniden kurmanız gerekir. Bağlantıları yeniden kurmazsanız, yeniden bağlanma girişimi sırasında uygulama sunucusu kaynakları gereksiz yere tutulur ve yeniden kullanıldıkları zamana kadar zamanaşımına uğramış olabilir.

WorkManager kullanımı

İşleme süresi birkaç saniyeden fazla olan uzun ömürlü uygulamalar (örneğin, toplu işleme) için WebSphere Application Server WorkManager kullanılabilir. WebSphere Application Server için bir kod parçası örneği:

```
public class BatchSenderServlet extends HttpServlet  {

  private WorkManager workManager = null;
  private MessageSender sender; // background sender WorkImpl

  public void init() throws ServletException {
    InitialContext ctx = new InitialContext();
    workManager = (WorkManager)ctx.lookup(java:comp/env/wm/default);
    sender = new MessageSender(5000);
    workManager.startWork(sender);
  }

  public void destroy() {
    sender.halt();
  }

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                              throws ServletException, IOException {
    res.setContentType("text/plain");
    PrintWriter out = res.getWriter();
    if (sender.isRunning()) {
      out.println(sender.getStatus());
    }
}
```

Burada `web.xml` aşağıdakileri içerir:

```
<resource-ref>
      <description>WorkManager</description>
      <res-ref-name>wm/default</res-ref-name>
      <res-type>com.ibm.websphere.asynchbeans.WorkManager</res-type>
      <res-auth>Container</res-auth>
      <res-sharing-scope>Shareable</res-sharing-scope>
   </resource-ref>
```

ve toplu iş şimdi iş arabirimi aracılığıyla uygulanır:

```
import com.ibm.websphere.asynchbeans.Work;

public class MessageSender implements Work {
```

```
public MessageSender(int messages) {numberOfMessages = messages;}

public void run() {
  // get connection factory/ queue
  InitialContext ic = new InitialContext();
  ConnectionFactory cf = (ConnectionFactory)
             ic.lookup("java:comp/env/jms/WMQCF");
  Destination destination = (Destination) ic.lookup("jms/WMQQueue");

  setupResources();

  // loop sending messages
  while (!sendComplete) {
    try {
      // create the next message to send
      msg.setText("message sent at "+new Date());
      // and send it
      producer.send(msg);
      // are we finished?
      if (sendCount == numberOfMessages) {sendComplete = true);
    }
    catch (JMSException je) {
        // drive reconnection
        setupResources();
    }
}

public boolean isRunning() {return !sendComplete;}

public void release() {sendComplete = true;}
```

Toplu iş işlemesinin çalıştırılması uzun sürerse (örneğin, büyük iletiler, yavaş ağ ya da kapsamlı veritabanı erişimi (özellikle yavaş yedek sisteme geçiş ile eşleştiğinde), sunucu, aşağıdaki örneğe benzer şekilde askıda kalmış iş parçacığı uyarılarının çıktısını almaya başlar:

WSVR0605W: İş parçacığı "`WorkManager.DefaultWorkManager` : 0" (00000035) `694061` milisaniye boyunca etkindir ve askıya alınabilir. Sunucuda askıda kalabilecek toplam 1 iş parçacığı var/var.

Bu uyarılar, toplu iş boyutu azaltılarak ya da askıda iş parçacığı zamanaşımı artırılarak en aza indirilebilir. Ancak, bu işlemi bir EJB ' de (toplu gönderme için) ya da ileti odaklı bean 'de (tüketme ya da tüketme ve yanıtlama için) gerçekleştirmeniz genellikle tercih edilir.

Uygulama tarafından yönetilen yeniden bağlantının çalıştırma zamanı hatalarını işlemek için genel bir çözüm sağlamadığını ve uygulamanın bağlantı hatasıyla ilgili olmayan hataları işlemesi gerektiğini unutmayın.

Örneğin, dolu olan bir kuyruğa ileti (2053 MQRC_Q_FULL) ya da geçerli olmayan güvenlik kimlik bilgilerini (2035 MQRC_NOT_AUTHORIZED) kullanarak bir kuyruk yöneticisine bağlanma girişiminde bulunuluyor.

Yedek sisteme geçiş işlemi devam ederken hemen kullanılabilecek eşgörünüm olmadığında, uygulama 2059 MQRC_Q_MGR_NOT_VAR olan hataları da işlemelidir. Bu, JMS kural dışı durumlarının oluştuğunda, sessiz bir şekilde yeniden bağlanmaya çalışmak yerine, bunları bildiren uygulama tarafından gerçekleştirilebilir.

IBM MQ classes for JMS nesne havuzlama
Java EE dışında bir bağlantı havuzu oluşturma biçiminin kullanılması, örneğin, çerçeveleri kullanan bazı bağımsız uygulamalardan ya da bulut ortamlarında devreye alınmasından ve `QueueManagers` ' e daha fazla sayıda istemci bağlantısından kaynaklanan genel yükün azaltılmasına yardımcı olur; bu da uygulamaların ve kuyruk yöneticilerinin sunucu birleştirmesinde artışa neden olur.

Java EE programlama modelinde, kullanılmakta olan çeşitli nesnelerin iyi tanımlanmış bir yaşam döngüsü vardır. Sunucu uygulamacıkları daha fazla özgürlük sağlarken, ileti odaklı fasulye (MDB) en kısıtlıdır. Bu nedenle, Java EE sunucularında bulunan havuzlama seçenekleri, kullanılan çeşitli programlama modellerine uyar.

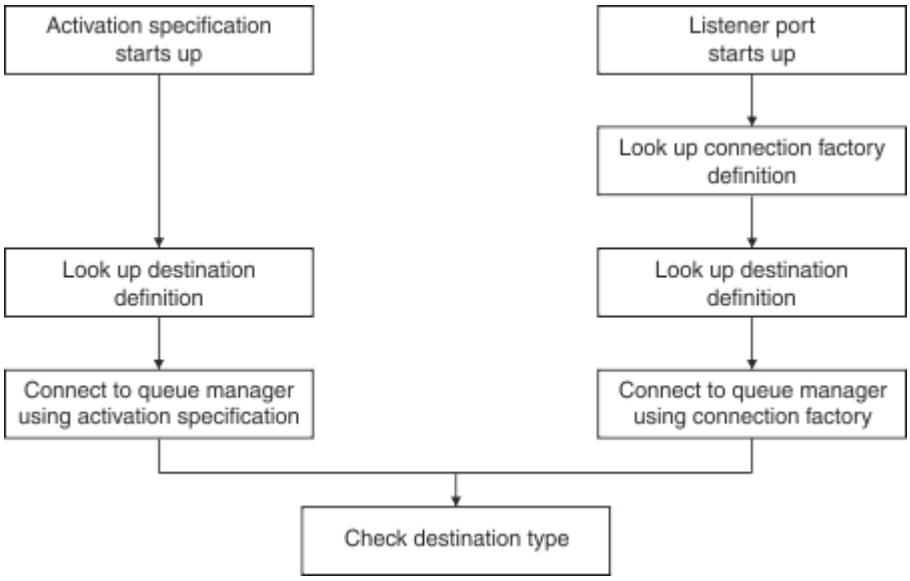
Java SE (ya da Spring gibi başka bir çerçeveyle) ile programlama modelleri son derece esnektir. Bu yüzden tek bir havuz oluşturma stratejisi her şeye uymuyor. Herhangi bir havuz oluşturacak bir çerçeve olup olmadığını düşünmelisiniz, örneğin, Bahar.

Kullanılacak havuz oluşturma stratejisi, uygulamanızın çalıştığı ortama bağlıdır.

Java EE ortamında nesne havuzu oluşturma
Java EE uygulama sunucuları, ileti odaklı Bean uygulamaları, Enterprise Java Beans ve Sunucu Uygulamaları tarafından kullanılabilen bağlantı havuzu oluşturma işlevselliği sağlar.

WebSphere Application Server , başarımı artırmak için bir JMS sağlayıcısına yönelik bir bağlantı havuzu sağlar. Bir uygulama JMS bağlantısı oluşturduğunda, uygulama sunucusu serbest bağlantı havuzunda bir bağlantının önceden var olup olmadığını belirler. Bu durumda, uygulamaya bağlantı döndürülür; tersi durumda, yeni bir bağlantı yaratılır.

Şekil 41 sayfa 292 , hem etkinleştirme belirtimlerinin hem de dinleyici kapılarının bir JMS bağlantısını nasıl kurduğunu ve normal kipteki iletilere ilişkin bir hedefi izlemek için bu bağlantıyı nasıl kullandığını gösterir.



Şekil 41. Normal kip

IBM MQ ileti alışverişi sağlayıcısını kullanırken, giden ileti alışverişi gerçekleştiren uygulamalar (kurumsal Java Bean ve sunucu uygulamaları gibi) ve iletiyle yönlendirilen Bean dinleyici kapısı bileşeni bu bağlantı havuzlarından yararlanabilir.

IBM MQ ileti alışverişi sağlayıcısı etkinleştirme belirtimleri, IBM MQ kaynak bağdaştırıcısı tarafından sağlanan bağlantı havuzu işlevini kullanır. Ek bilgi için WebSphere MQ kaynak bağdaştırıcısına ilişkin özelliklerin yapılandırılması başlıklı konuya bakın.

“Bağlantı havuzunu kullanma örnekleri” sayfa 296 içinde, giden ileti alışverişi gerçekleştiren uygulamaların ve dinleyici kapılarının JMS bağlantıları oluştururken boş havuzu nasıl kullandığı açıklanmaktadır.

“Serbest bağlantı havuzu bakım iş parçacıkları” sayfa 298 , bir uygulama ya da dinleyici kapısı bağlantılarla çalışmayı tamamladığında bu bağlantılara ne olacağını açıklar.

“Havuz bakımı iş parçacığı örnekleri” sayfa 300 , JMS bağlantılarının eski olmasını önlemek için serbest bağlantı havuzunun nasıl temizlendiğini açıklar.

WebSphere Application Server , Bağlantı Üreticisi 'nin *bağlantı sayısı üst sınırı* özelliğiyle belirtilen, bir üreticiden yaratılabilecek bağlantı sayısı üzerinde bir sınıra sahiptir. Bu özelliğin varsayılan değeri 10 'dur; bu, bir üreticiden bir kerede en çok 10 bağlantı oluşturulabileceği anlamına gelir.

Her üreticinin ilişkili bir boş bağlantı havuzu vardır. Uygulama sunucusu başlatıldığında, serbest bağlantı havuzları boş olur. Bir üretici için serbest havuzda bulunabilecek bağlantı sayısı üst sınırı, Bağlantı sayısı üst sınırı özelliği tarafından da belirtilir.

İpucu: JMS 2.0ile hem bağlantılar hem de bağlamlar oluşturmak için bir bağlantı üreticisi kullanılabilir. Sonuç olarak, hem bağlantıların hem de bağlamların karışımını içeren bir bağlantı üreticisiyle ilişkilendirilmiş bir bağlantı havuzu olabilir. Bir bağlantı üreticisinin yalnızca bağlantı yaratmak ya da

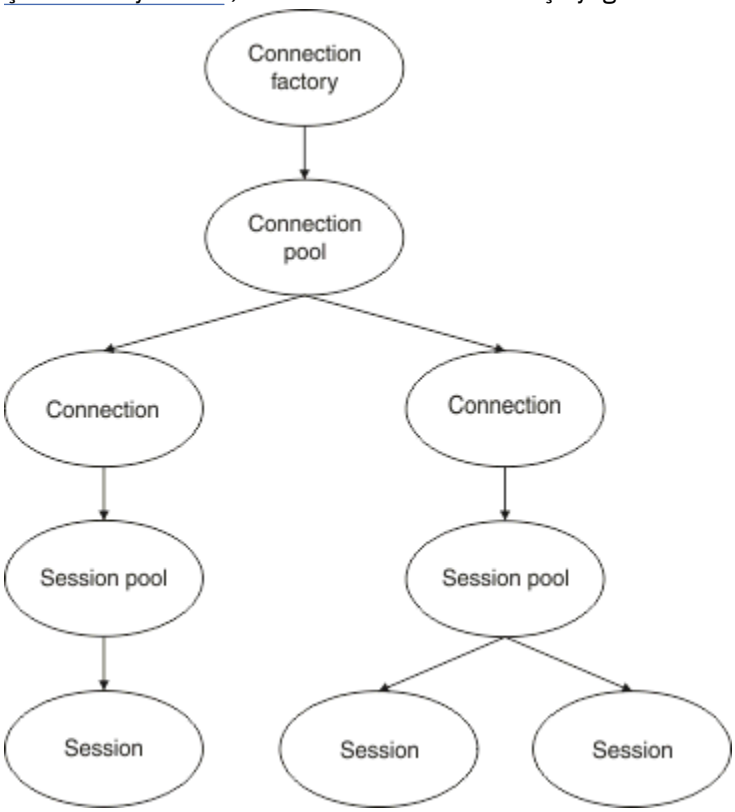
bağlam yaratmak için kullanılması önerilir. Bu, bağlantı üreticisine ilişkin bağlantı havuzunun yalnızca tek tipli nesneler içermesini sağlayarak havuzu daha verimli hale getirir.

WebSphere Application Server' de bağlantı havuzlama özelliğinin nasıl çalıştığına ilişkin bilgi için JMS bağlantıları için bağlantı havuzlama yapılandırılmasıbaşlıklı konuya bakın. Diğer uygulama sunucuları için uygun uygulama sunucusu belgelerine bakın.

Bağlantı havuzunun nasıl kullanıldığını

Her JMS bağlantı üreticisiyle ilişkilendirilmiş bir bağlantı havuzu vardır ve bağlantı havuzu sıfır ya da daha fazla JMS bağlantısı içerir. Her JMS bağlantısının ilişkili bir JMS oturum havuzu vardır ve her JMS oturum havuzu sıfır ya da daha fazla JMS oturumu içerir.

Şekil 42 sayfa 293 , bu nesneler arasındaki ilişkiyi gösterir.



Şekil 42. Bağlantı havuzları ve oturum havuzları

Bir dinleyici kapısı başlatıldığında ya da giden ileti alışverişi yapmak isteyen bir uygulama bağlantı yaratmak için üreticiyi kullandığında, kapı ya da uygulama aşağıdaki yöntemlerden birini çağırır:

- **connectionFactory.createConnection()**
- **ConnectionFactory.createConnection(String, String)**
- **QueueConnectionFactory.createQueueConnection()**
- **QueueConnectionFactory.createQueueConnection(String, String)**
- **TopicConnectionFactory.createTopicConnection()**
- **TopicConnectionFactory.createTopicConnection(String, String)**

WebSphere Application Server bağlantı yöneticisi, bu üreticiye ilişkin boş havuzdan bir bağlantı elde etmeye çalışır ve bunu uygulamaya geri döndürür.

Havuzda boş bağlantı yoksa ve bu üreticiden oluşturulan bağlantı sayısı, o üreticinin *bağlantı sayısı üst sınırı* özelliğinde belirtilen sınıra ulaşmamışsa, Bağlantı Yöneticisi uygulamanın kullanması için yeni bir bağlantı yaratır.

Ancak, bir uygulama bağlantı yaratmayı denerse, ancak bu üreticiden oluşturulan bağlantı sayısı, üreticinin *bağlantı sayısı üst sınırı* özelliğine eşitse, uygulama bir bağlantının kullanılabilir olmasını (serbest havuza geri konması için) bekler.

Uygulamanın bekleyeceği süre, varsayılan değeri 180 saniye olan bağlantı havuzunun *bağlantı zamanaşımı* özelliğinde belirtilir. Bir bağlantı bu 180 saniyelik süre içinde serbest havuza geri konursa, Bağlantı Yöneticisi bağlantıyı hemen havuzdan çıkarır ve uygulamaya iletir. Ancak, zamanaşımı süresi geçerse, bir *ConnectionWaitTimeoutException* yayınlanır.

Bir uygulama bağlantıyı bitirdiğinde ve kapattığında aşağıdakileri arayarak:

- **`Connection.close()`**
- **`QueueConnection.close()`**
- **`TopicConnection.close()`**

Bağlantı gerçekten açık tutulur ve başka bir uygulama tarafından yeniden kullanılabilmesi için serbest havuza döndürülür. Bu nedenle, uygulama sunucusunda JMS uygulaması çalışmasa da WebSphere Application Server ile JMS sağlayıcısı arasında açık bağlantılar olabilir.

Gelişmiş bağlantı havuzu özellikleri
JMS bağlantı havuzlarının davranışını denetlemek için kullanılabilecek bazı gelişmiş özellikler vardır.

Dalgalanma koruması

"Giden ileti alışverişi gerçekleştiren uygulamaların bağlantı havuzunu kullanma şekli" sayfa 297 içinde, `connectionFactory.createConnection()` ile birleştirilen `sendMessage()` yönteminin kullanımı açıklanmaktadır.

`ejbCreate()` yönteminin bir parçası olarak aynı bağlantı üreticisinden JMS bağlantıları yaratan 50 EJB ' niz olduğunu göz önünde bulundurun.

Bu Bean 'lerin tümü aynı anda yaratılırsa ve üreticinin serbest bağlantı havuzunda bağlantı yoksa, uygulama sunucusu aynı JMS sağlayıcısına aynı anda 50 JMS bağlantısı oluşturmayı dener. Sonuç hem WebSphere Application Server hem de JMS sağlayıcısına önemli bir yüktür.

Dalgalanma koruması özellikleri, herhangi bir zamanda bir bağlantı üreticisinden oluşturulabilecek JMS bağlantılarının sayısını sınırlayarak ve ek bağlantı yaratılmasını aşamalandırarak bu durumu önleyebilir.

JMS bağlantılarının sayısının bir kerede sınırlanması iki özellik kullanılarak gerçekleştirilir:

- Dalgalanma eşiği
- Dalgalanma yaratma aralığı.

EJB uygulamaları bir bağlantı üreticisinden JMS bağlantısı yaratmayı denediğinde, bağlantı yöneticisi kaç bağlantının yaratıldığını denetler. Bu sayı `surge threshold` özelliğinin değerinden küçük ya da ona eşitse, bağlantı yöneticisi yeni bağlantılar açmaya devam eder.

Ancak, yaratılmakta olan bağlantı sayısı `surge threshold` özelliğini aşarsa, bağlantı yöneticisi yeni bir bağlantı yaratmadan ve açmadan önce `surge creation interval` özelliğinin belirlediği süreyi bekler.

Sıkışmış bağlantılar

Bir JMS uygulaması bu bağlantıyı JMS sağlayıcısına bir istek göndermek için kullanıyorsa ve sağlayıcı belirli bir süre içinde yanıt vermiyorsa, JMS bağlantı `stuck`olarak kabul edilir.

WebSphere Application Server , `stuck` JMS bağlantılarını saptamanın bir yolunu sağlar. Bu işlevi kullanmak için üç özellik ayarlamanız gerekir:

- Sıkışmış Zaman Zamanlayıcısı
- Sıkışmış Saat
- Sıkışmış Eşik

“Havuz bakımı iş parçacığı örnekleri” sayfa 300 , havuz bakımı iş parçacığının belirli aralıklarla nasıl çalıştığını ve bir bağlantı üreticisinin serbest havuzunun içeriğini nasıl denetlediğini, belirli bir süre kullanılmayan ya da çok uzun süredir var olan bağlantıları aradığını açıklar.

Sıkışmış bağlantıları saptamak için uygulama sunucusu, JMS sağlayıcısından yanıt bekleyip beklemediğini görmek için bir bağlantı üreticisinden oluşturulan tüm etkin bağlantıların durumunu denetleyen sıkışmış bir bağlantı iş parçacığını da yönetir.

Sıkışmış bağlantı iş parçacığı çalıştığında, `Stuck time timer` özelliği tarafından belirlenir. Bu özelliğin varsayılan değeri sıfırdır; bu, sıkışmış bağlantı algılamanın hiçbir zaman çalışmadığı anlamına gelir.

İş parçacığı yanıt bekleyen bir yanıt bulursa, yanıtı ne kadar beklediğini belirler ve bu süreyi `Stuck time` özelliğinin değeriyle karşılaştırır.

JMS sağlayıcısının yanıt vermesi için geçen süre `Stuck time` özelliğinin belirlediği süreyi aşarsa, uygulama sunucusu JMS bağlantısını sıkışmış olarak işaretler.

Örneğin, `jms/CF1` bağlantı üreticisinin `Stuck time timer` özelliğini 10 olarak ve `Stuck time` özelliğini 15 olarak ayarladığını varsayın.

Sıkışmış bağlantı iş parçacığı her 10 saniyede bir etkin duruma gelir ve `jms/CF1` 'tan oluşturulan herhangi bir bağlantının IBM MQ' den yanıt almak için 15 saniyeden uzun bir süre bekleyip beklemediğini denetler.

Bir EJB ' nin `jms/CF1`kullanarak IBM MQ ile JMS bağlantısı yarattığını ve daha sonra, `Connection.createSession()` çağrısıyla bu bağlantıyı kullanarak bir JMS oturumu yaratmayı denediğini varsayın.

Ancak bir şey, JMS sağlayıcısının isteğe yanıt vermesini engelliyor. Makine donmuş olabilir ya da JMS sağlayıcısında çalışan bir işlem ölümcül kilitlenerek yeni işlerin işlenmesini önlemiş olabilir:

EJB 'nin `Connection.createSession()` çağrısından on saniye sonra, sıkışmış bağlantı zamanlayıcısı etkin duruma gelir ve `jms/CF1`' den oluşturulan etkin bağlantılara bakar.

Yalnızca bir etkin bağlantı olduğunu varsayın; örneğin, `c1`. İlk EJB, `c1`' e gönderdiği bir isteğe yanıt için 10 saniye beklemektedir; bu, `Stuck time`değerinden küçük bir değerdir, bu nedenle sıkışan bağlantı süreölçeri bu bağlantıyı yoksayar ve devre dışı kalır.

10 saniye sonra, sıkışmış bağlantı iş parçacığı yeniden etkin duruma gelir ve `jms/CF1`için etkin bağlantılara bakar. Daha önce olduğu gibi, yalnızca tek bir bağlantının (`c1`) olduğunu varsayın.

`createSession()` adlı ilk EJB ' nin çağrılmasından bu yana 20 saniye geçti ve EJB hala yanıt bekliyor. 20 saniye, `Stuck time` özelliğinde belirtilen süreden daha uzun olduğundan, sıkışmış bağlantı iş parçacığı `c1` ' i sıkışmış olarak işaretler.

Beş saniye sonra IBM MQ nihayet yanıt verirse ve ilk EJB ' nin bir JMS Oturumu yaratmasına izin verirse, bağlantı yeniden kullanılır.

Uygulama sunucusu, sıkışmış bir bağlantı üreticisinden oluşturulan JMS bağlantılarının sayısını sayar. Bir uygulama yeni bir JMS Connection oluşturmak için bu bağlantı üreticisini kullandığında ve o üreticinin serbest havuzunda serbest bağlantı olmadığında, bağlantı yöneticisi sıkışmış bağlantı sayısını `Stuck threshold` özelliğinin değeriyle karşılaştırır.

Sıkışmış bağlantıların sayısı `Stuck threshold`özelliği için ayarlanan değerden azsa, bağlantı yöneticisi yeni bir bağlantı yaratır ve bunu uygulamaya verir.

Ancak, sıkışmış bağlantı sayısı `Stuck threshold` özelliğinin değerine eşitse, uygulama bir kaynak kural dışı durumu alır.

Havuz bölümleri

WebSphere Application Server , bir bağlantı üreticisi için serbest bağlantı havuzunu bölümlemenizi sağlayan iki özellik sağlar:

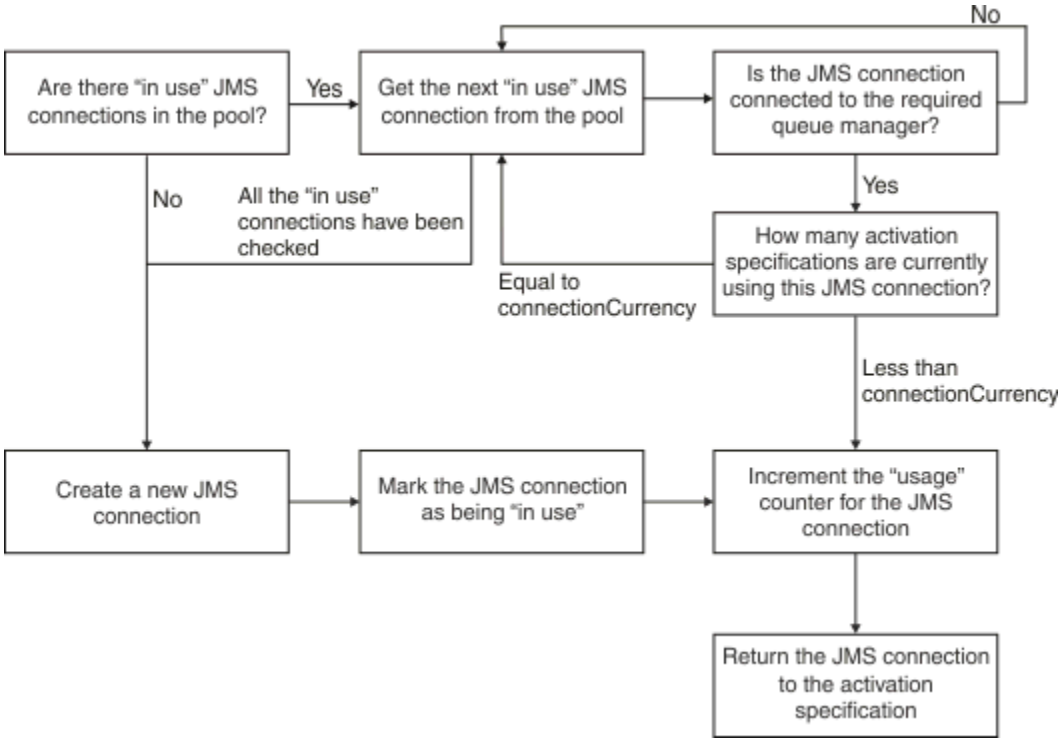
- `Number of free pool partitions` , uygulama sunucusuna serbest bağlantı havuzunu kaç bölüme ayırmak istediğinizi bildirir.
- `Free pool distribution table size` , bölümlerin nasıl dizinleneceğini belirler.

IBM Destek Merkeziniz tarafından bu özellikleri değiştirmeniz istenmedikçe, bu özellikleri varsayılan sıfır değerlerinde bırakın.

WebSphere Application Server ' in `Number of shared partitions`adlı ek bir gelişmiş bağlantı havuzu özelliği olduğunu unutmayın. Bu özellik, paylaşılan bağlantıları saklamak için kullanılan bölüm sayısını belirtir. Ancak, JMS bağlantıları her zaman paylaşılmadığı için bu özellik geçerli değildir.

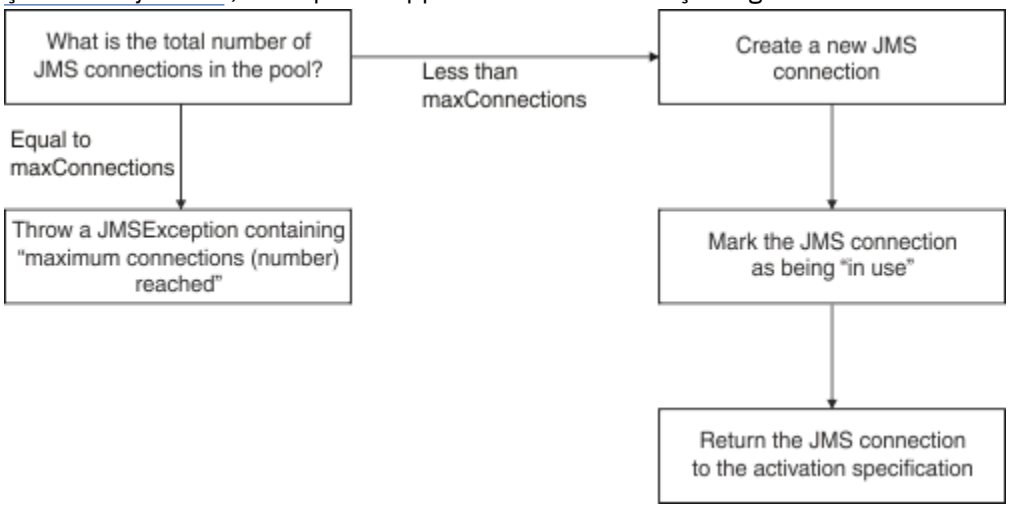
Bağlantı havuzunu kullanma örnekleri
İletiyle yönlendirilen Bean dinleyici kapısı bileşeni ve giden ileti alışverişi gerçekleştiren uygulamalar bir JMS bağlantı havuzu kullanır.

Şekil 43 sayfa 296 içinde, WebSphere Application Server V7.5 ve V8.0için bağlantı havuzunun nasıl çalıştığı gösterilmektedir.



Şekil 43. WebSphere Application Server V7.5 ve V8.0 -bağlantı havuzunun çalışma şekli

Şekil 44 sayfa 296 , WebSphere Application Server V8.5için bağlantı havuzunun nasıl çalıştığını gösterir.



Şekil 44. WebSphere Application Server V8.5 -Bağlantı havuzunun çalışma şekli

MDB dinleyici kapılarının bağlantı havuzunu kullanma şekli

JMS sağlayıcısı olarak IBM MQ kullanan bir WebSphere Application Server Network Deployment sisteminde konuşlandırılmış bir MDB ' nin olduğunu varsayın. MDB, *bağlantı sayısı üst sınırı* özelliği 2 olarak ayarlanmış `jms/CF1`gibi, çağrılan bir bağlantı üreticisini kullanan bir dinleyici kapısına konuşlandırılır; bu, aynı anda bu üreticiden yalnızca iki bağlantı yaratılabileceği anlamına gelir.

Dinleyici kapısı başlatıldığında, kapı`jms/CF1` bağlantı üreticisini kullanarak IBM MQile bağlantı oluşturmayı dener.

Bunu yapmak için kapı, bağlantı yöneticisinden bir bağlantı isteğinde bulunur. `jms/CF1` bağlantı üreticisi ilk kez kullanıldığından, `jms/CF1` serbest bağlantı havuzunda bağlantı yoktur, bu nedenle bağlantı yöneticisi `c1`gibi çağrılan yeni bir bağlantı yaratır. Bu bağlantının, dinleyici kapısının tüm ömrü boyunca var olduğunu unutmayın.

Şimdi, WebSphere Application Server denetim konsolunu kullanarak dinleyici kapısını durduracağınız durumu göz önünde bulundurun. Bu durumda, bağlantı yöneticisi bağlantıyı alır ve serbest havuza geri koyar. Ancak, IBM MQ bağlantısı açık kalır.

Dinleyici kapısını yeniden başlatacaksanız, kapı bir kez daha bağlantı yöneticisinden kuyruk yöneticisiyle bağlantı kurmasını ister. Artık serbest havuzda bir bağlantınız (`c1`) olduğu için, bağlantı yöneticisi bu bağlantıyı havuzdan çıkarır ve dinleyici kapısı tarafından kullanılabilir kılar.

Şimdi, uygulama sunucusunda dağıtılmış ikinci bir MDB ' nin olduğunu ve bunun farklı bir dinleyici kapısı kullandığını varsayın.

Daha sonra, `jms/CF1` bağlantı üreticisini kullanacak şekilde yapılandırılmış üçüncü bir dinleyici kapısı başlatmayı denediğinizi varsayın. Üçüncü dinleyici kapısı, `jms/CF1` için boş havuza bakan bağlantı yöneticisinden bir bağlantı ister ve boş olduğunu bulur. Daha sonra, `jms/CF1` üreticisinden önceden kaç bağlantı oluşturulduğunu denetler.

`jms/CF1` için bağlantı sayısı üst sınırı özelliği 2 olarak ayarlandığından ve bu üreticiden iki bağlantı yarattığından, bağlantı yöneticisi bir bağlantının kullanılabilir olması için 180 saniye bekler (bağlantı zamanaşımı özelliğinin varsayılan değeri).

Ancak, ilk dinleyici kapısını durdurursanız, `c1` bağlantısı `jms/CF1`için boş havuza konur. Bağlantı yöneticisi bu bağlantıyı alır ve üçüncü dinleyiciye verir.

Şimdi ilk dinleyiciyi yeniden başlatmayı denerseniz, bu dinleyicinin, ilk dinleyicinin yeniden başlatılabilmesi için diğer dinleyici kapılarından birinin durmasını beklemesi gerekir. Çalışan dinleyici kapılarından hiçbiri 180 saniye içinde durdurulmazsa, ilk dinleyici bir `ConnectionWaitTimeoutException` hatası alır ve durur.

Giden ileti alışverişi gerçekleştiren uygulamaların bağlantı havuzunu kullanma şekli

Bu seçenek için, uygulama sunucusunda `EJB1`gibi tek bir EJB ' nin kurulu olduğunu varsayın. Bean, aşağıdaki yöntemlerle `sendMessage()` adlı bir yöntem gerçekleştirir:

- `connectionFactory.createConnection()`kullanarak IBM MQ üreticisinden `jms/CF1`bir JMS bağlantısı oluşturma.
- Bağlantıdan bir JMS oturumu oluşturma.
- Oturumdan bir ileti üreticisi yaratılıyor.
- Mesaj gönderiyorum.
- Yapımcı kapatılıyor.
- Oturum kapatılıyor.
- `connection.close()`' i arayarak bağlantı kapatılıyor.

`jms/CF1` üreticisine ilişkin boş havuzun boş olduğunu varsayın. EJB ilk kez çağrıldığında, Bean `jms/CF1`üreticisinden IBM MQ ile bağlantı yaratmayı dener. Üreticiye ilişkin boş havuz boş olduğundan, bağlantı yöneticisi yeni bir bağlantı yaratır ve bunu EJB1' e verir.

Yöntem çıkmadan hemen önce `connection.close()` yöntemini çağırır. Bağlantı yöneticisi, `c1`' i kapatmak yerine bağlantıyı alır ve `jms/CF1`için boş havuza yerleştirir.

`sendMessage()` ' in bir sonraki çağrılışında, `connectionFactory.createConnection()` yöntemi uygulamaya `c1` değerini döndürür.

İlk eşgörünümle aynı anda çalışan EJB ' nin ikinci bir eşgörünümünün olduğunu varsayın. Her iki yönetim ortamı da `sendMessage()` komutunu çağırırken, `jms/CF1` bağlantı üreticisinden iki bağlantı yaratılır.

Şimdi, bean 'in üçüncü bir eşgörünümünün yaratıldığını varsayın. Üçüncü Bean `sendMessage()`'i çağırdığında, yöntem `jms/CF1`' den bağlantı yaratmak için `connectionFactory.createConnection()` ' i çağırır.

Ancak, şu anda `jms/CF1`' dan oluşturulan ve bu üretici için bağlantı sayısı üst sınırı değerine eşit olan iki bağlantı vardır. Bu nedenle `createConnection()` yöntemi, bir bağlantının kullanılabilir olması için 180 saniye (bağlantı zamanaşımı özelliğinin varsayılan değeri) bekler.

Ancak, ilk EJB çağrıları için `sendMessage()` yöntemi `connection.close()` ve çıkış yaparsa, kullandığı bağlantı (`c1`) serbest bağlantı havuzuna geri konar. Bağlantı yöneticisi bağlantıyı serbest havuzdan geri alır ve üçüncü EJB ' ye verir. Bu Bean 'den `connectionFactory.createConnection()` ' e çağrı, `sendMessage()` yönteminin tamamlanmasına izin vererek geri döner.

Aynı bağlantı havuzunu kullanan MDB dinleyici kapıları ve EJB ' ler

Önceki iki örnek, dinleyici kapılarının ve EJB ' lerin bağlantı havuzunu yalıtılmış olarak nasıl kullanabileceğini gösterir. Ancak, aynı uygulama sunucusunda çalışan bir dinleyici kapısı ve bir EJB olabilir ve aynı bağlantı üreticisini kullanarak JMS bağlantıları oluşturabilirsiniz.

Bu durumun etkilerini göz önünde bulundurmalısın.

Hatırlamanız gereken anahtar şey, bağlantı üreticisinin dinleyici kapısı ile EJB arasında paylaşılmasıdır.

Örneğin, aynı anda çalışan bir dinleyiciniz ve EJB ' iniz olduğunu varsayın. Her ikisi de `jms/CF1` bağlantı üreticisini kullanıyor; bu, o üretici için bağlantı sayısı üst sınırı özelliğinin belirlediği bağlantı sınırına ulaşıldığı anlamına gelir.

Başka bir dinleyici kapısını ya da EJB ' nin başka bir eşgörünümünü başlatmayı denerseniz, `jms/CF1`için serbest bağlantı havuzuna bir bağlantı döndürülmesini beklemeniz gerekir.

Serbest bağlantı havuzu bakım iş parçacıkları
Her bir boş bağlantı havuzuyla ilişkilendirilmiş bir havuz bakımı iş parçacığıdır. Bu iş parçacığı, havuzdaki bağlantıların hala geçerli olduğundan emin olmak için boş havuzu izler.

Havuz bakımı iş parçacığı, serbest havuzdaki bir bağlantının atılması gerektiğine karar verirse, iş parçacığı JMS bağlantısını IBM MQile fiziksel olarak kapatır.

Havuz bakımı iş parçacığı nasıl çalışır

Havuz bakımı iş parçacığının davranışı, bağlantı havuzunun dört özelliğinin değerine göre belirlenir:

Eskilerarası zamanaşımı
Bir bağlantının açık kaldığı süre.

Bağlantı sayısı alt sınırı
Bağlantı yöneticisinin bir bağlantı üreticisinin boş havuzunda tuttuğu bağlantı sayısı alt sınırı.

Toplama süresi
Havuz bakım iş parçacığının çalışma sıklığı.

Kullanılmayan zamanaşımı
Bir bağlantının kapatılmadan önce boş havuzda ne kadar süre kalacağı.

Varsayılan olarak, havuz korunan iş parçacığı 180 saniyede bir çalışır, ancak bu değer bağlantı havuzu **`Reap time`** özelliği ayarlanarak değiştirilebilir.

Bakım iş parçacığı havuzdaki her bağlantıya bakar, havuzda ne kadar süre olduğunu ve yaratıldığından ve son kullanıldığından bu yana geçen süreyi denetler.

Bağlantı, bağlantı havuzu için **Unused timeout** özelliğinin değerinden daha uzun bir süre kullanılmamışsa, bakım iş parçacığı, serbest havuzdaki bağlantı sayısını denetler. Bu sayı:

- **Minimum connections**değerinden büyük olan bağlantı yöneticisi bağlantıyı kapatır.
- **Minimum connections**değerine eşittir, bağlantı kapatılmaz ve boş havuzda kalır.

Minimum connections özelliğinin varsayılan değeri *1*' dir; bu, performans nedeniyle bağlantı yöneticisinin her zaman en az bir bağlantıyı boş havuzda tutmaya çalıştığı anlamına gelir.

Unused timeout özelliğinin varsayılan değeri 1800 saniyedir. Varsayılan olarak, bir bağlantı serbest havuza geri konursa ve en az 1800 saniye kullanılmazsa, bağlantı kapatılırsa, serbest havuzda en az bir bağlantı kalır.

Bu yordam, kullanılmayan bağlantıların eskimiş olmasını önler. Bu özelliği kapatmak için **Unused timeout** özelliğini sıfır olarak ayarlayın.

Bir bağlantı boş havuzdaysa ve yaratılmasından bu yana geçen süre, bağlantı havuzu için **Aged timeout** özelliğinin değerinden büyükse, son kullanılmasından bu yana ne kadar süre geçtiğinden bağımsız olarak kapatılır.

Varsayılan olarak, **Aged timeout** özelliği sıfır olarak ayarlanır; bu, bakım iş parçacığının bu denetimi hiçbir zaman gerçekleştirmediği anlamına gelir. **Aged timeout** özelliğinden daha uzun süredir var olan bağlantılar, serbest havuzda kaç bağlantının kalacağına bakılmaksızın atılır. **Minimum connections** özelliğinin bu durumda herhangi bir etkisi olmadığını unutmayın.

Havuz bakımı iş parçacığının devre dışı bırakılması

Önceki açıklamadan, özellikle bağlantı üreticisinin serbest havuzunda çok sayıda bağlantı varsa, havuz bakımı iş parçacığının etkin olduğunda çok fazla iş yaptığını görebilirsiniz.

Örneğin, her bir fabrika için **Maximum connections** özelliği 10 olarak ayarlanmış üç JMS bağlantı üreticisi olduğunu varsayın. Her 180 saniyede bir, üç havuz bakımı iş parçacığı etkinleşir ve her bağlantı üreticisi için serbest havuzları tarar. Serbest havuzların çok sayıda bağlantısı varsa, bakım iş parçacıklarının yapacak çok işi vardır ve bu da performansı önemli ölçüde etkileyebilir.

Reap time özelliğini sıfıra ayarlayarak tek bir serbest bağlantı havuzu için havuz bakımı iş parçacığını devre dışı bırakabilirsiniz.

Bakım iş parçacığının devre dışı bırakılması, **Unused timeout** geçmiş olsa bile bağlantıların hiçbir zaman kapatılmadığı anlamına gelir. Ancak **Aged timeout** geçtiyse, bağlantılar yine de kapatılabilir.

Bir uygulama bağlantıyı bitirdiğinde, bağlantı yöneticisi bağlantının ne kadar süre var olduğunu denetler ve bu süre **Aged timeout** özelliğinin değerinden uzunsa, bağlantı yöneticisi bağlantıyı boş havuza döndürmek yerine kapatır.

Esneme zamanaşımının işlemsel etkileri

Önceki kısımda açıklandığı gibi, **Aged timeout** özelliği, bağlantı yöneticisi bağlantıyı kapatmadan önce JMS sağlayıcısıyla bağlantının ne kadar süreyle açık kalacağını belirtir.

Aged timeout özelliğinin varsayılan değeri sıfırdır; bu, bağlantının çok eski olduğu için hiçbir zaman kapatılamayacağını gösterir. **Aged timeout** özelliğini bu değerde bırakmanız gerekir; **Aged timeout** 'in etkinleştirilmesi, EJB' lerin içinde JMS ' yi kullanırken işlemsel etkilere sahip olabilir.

JMSiçinde bir hareket birimi, JMS *bağlantısından*oluşturulan bir JMS *oturumudur*. Bu, JMS *bağlantısına*değil, hareketlere eklenen JMS *oturumudur* .

Due to the design of the application server, JMS connections can be closed because the **Aged timeout** has elapsed, even if JMS sessions created from that connection are involved in a transaction.

Bir JMS bağlantısının kapatılması, JMS belirtiminde açıklandığı gibi JMS oturumlarındaki işlem bekleyen işlerin geriye işlenmesine neden olur. Ancak, uygulama sunucusu bağlantıdan yaratılan JMS oturumlarının

artık geçerli olmadığını bilmez. Sunucu bir hareketi kesinleştirmek ya da geriye işlemek için oturumu kullanmayı denediğinde `IllegalStateException` oluşur.

Önemli: Aged timeout komutunu EJB ' ler içinden JMS bağlantılarıyla kullanmak istiyorsanız, JMS işlemlerini gerçekleştiren EJB yönteminden önce, JMS oturumunda herhangi bir JMS işinin belirtik olarak kesinleştirildiğini doğrulayın.

Havuz bakımı iş parçacığı örnekleri
Havuz bakımı iş parçacığının nasıl çalıştığını anlamak için Enterprise JavaBean (EJB) örneğini kullanma. İhtiyacınız olan tek şey, boş havuzdaki bağlantıları almak için bir yol olduğundan, İleti Odaklı Bean (MDB) ve dinleyici kapılarını da kullanabilirsiniz.

`sendMessage()` yöntemine ilişkin ek ayrıntılar için bkz. "Giden ileti alışverişi gerçekleştiren uygulamaların bağlantı havuzunu kullanma şekli" sayfa 297 .

Bağlantı üreticisini aşağıdaki değerlerle yapılandırdınız:

- Varsayılan değeri 180 saniye olan **Reap time**
- **Aged timeout** varsayılan değeri olan sıfır saniye ile
- **Unused timeout** 300 saniye olarak ayarlandı

Uygulama sunucusu başlatıldıktan sonra `sendMessage()` yöntemi çağrılır.

Yöntem, örneğin, `c1`fabrikayı kullanarak `jms/CF1`adlı bir bağlantı yaratır, bir ileti göndermek için bu üreticiyi kullanır ve daha sonra, `c1` 'un serbest havuza konmasına neden olan `connection.close()`' i çağırır.

180 saniye sonra havuz bakımı iş parçacığı başlar ve `jms/CF1` serbest bağlantı havuzuna bakar. Havuzda boş bir bağlantı `c1` var, bu nedenle bakım iş parçacığı bağlantının geri konma zamanına bakar ve bunu geçerli saatle karşılaştırır.

Bağlantının boş havuza konmasından bu yana 180 saniye geçti; bu, `jms/CF1`için **Unused timeout** özelliğinin değerinden küçük. Bu nedenle, bakım iş parçacığı bağlantıyı yalnız bırakır.

180 saniye sonra havuz bakımı iş parçacığı yeniden çalışır. Bakım iş parçacığı `c1`bağlantısını bulur ve bağlantının 360 saniye boyunca havuzda olduğunu belirler; bu, **Unused timeout** değer kümesinden daha uzundur, bu nedenle bağlantı yöneticisi bağlantıyı kapatır.

`sendMessage()` yöntemini yeniden çalıştırırsanız, uygulama `connectionFactory.createConnection()` 'ı çağırdığında bağlantı yöneticisi, bağlantı üreticisine ilişkin serbest bağlantı havuzu boş olduğundan IBM MQ ile yeni bir bağlantı oluşturur.

Önceki örnekte, **Aged timeout** özelliği sıfıra ayarlandığında, bakım iş parçacığının eski bağlantıları önlemek için **Reap time** ve **Unused timeout** özelliklerini nasıl kullandığı gösterilmektedir.

Aged timeout özelliği nasıl çalışır?

Aşağıdaki örnekte, aşağıdakileri ayarladığınızı varsayın:

- **Aged timeout** özelliği-300 saniye
- **Unused timeout** özelliği sıfıra.

`sendMessage()` yöntemini çağırıyorsunuz ve bu yöntem `jms/CF1` bağlantı üreticisinden bağlantı yaratmayı deniyor.

Bu üreticiye ilişkin boş havuz boş olduğundan, bağlantı yöneticisi yeni bir bağlantı (`c1`) oluşturur ve bunu uygulamaya geri döndürür. `sendMessage()` `connection.close()`aradığında, `c1` serbest bağlantı havuzuna geri konir.

180 saniye sonra havuz bakımı iş parçacığı çalışır. İş parçacığı, serbest bağlantı havuzunda `c1` öğesini bulur ve ne kadar süre önce oluşturulduğunu denetler. Bağlantı, **Aged timeout**' dan daha kısa olan 180 saniye boyunca var oldu, bu nedenle havuz bakımı iş parçacığı bağlantıyı yalnız bırakır ve uykuya geri döner.

60 saniye sonra `sendMessage()` yeniden çağrılır. Bu kez, yöntem `connectionFactory.createConnection()`' i çağırdığında, bağlantı yöneticisi `jms/CF1`için serbest

havuzda bir bağlantı (c1) olduğunu keşfeder. Bağlantı yöneticisi c1 ' i serbest havuzdan çıkarır ve bu bağlantıyı uygulamaya verir.

sendMessage() ' tan çıkıldığında, bağlantı boş havuza döndürülür. 120 saniye sonra havuz bakımı iş parçacığı yeniden uyanır, jms/CF1 için boş havuzun içeriğini tarar ve c1öğesini keşfeder.

Bağlantı yalnızca 120 saniye önce kullanılsa da, **Aged timeout** özelliği için ayarladığınız 300 saniyelik değerden daha uzun olan toplam 360 saniye boyunca var olduğu için havuz bakımı iş parçacığı bağlantıyı kapatır.

Bağlantı alt sınırı özelliğinin havuz bakımı iş parçacığını nasıl etkilediği

"MDB dinleyici kapılarının bağlantı havuzunu kullanma şekli" sayfa 297 örneğini yeniden kullanarak, her biri farklı bir dinleyici kapısı kullanan, uygulama sunucunuzda konuşlandırılmış iki veritabanı olduğunu varsayın.

Her bir dinleyici kapısı, aşağıdaki ürünle yapılandırdığınız jms/CF1 bağlantı üreticisini kullanacak şekilde yapılandırılır:

- **Unused timeout** özelliği 120 saniye olarak ayarlandı
- **Reap time** özelliği 180 saniye olarak ayarlandı
- **Minimum connections** özelliği 1 olarak ayarlandı

İlk dinleyicinin durdurulduğunu ve c1 bağlantısının boş havuza konduğunu varsayın. 180 saniye sonra havuz bakımı iş parçacığı uyanır, jms/CF1için boş havuzun içeriğini tarar ve c1 ' un bağlantı üreticisine ilişkin **Unused timeout** özelliğinin değerinden daha uzun süredir boş havuzda olduğunu keşfeder.

Ancak, c1kapatılmadan önce havuz bakımı iş parçacığı, bu bağlantı atılırsa havuzda kaç bağlantının kalacağını görebilir. c1 , serbest bağlantı havuzundaki tek bağlantı olduğundan, bağlantı yöneticisi bunu kapatmaz; bunu yaparsa, boş havuzda kalan bağlantı sayısı, **Minimum connections**için ayarlanan değerden daha az olur.

Şimdi, ikinci dinleyicinin durdurulduğunu varsayın. Serbest bağlantı havuzu artık iki boş bağlantı içerir: c1 ve c2.

180 saniye sonra havuz bakımı iş parçacığı yeniden çalışır. Bu zamana kadar, c1 360 saniye ve c2 180 saniye boyunca serbest bağlantı havuzunda olmuştur.

Havuz bakımı iş parçacığı c1 değerini denetler ve havuzda **Unused timeout** özelliğinin değerinden daha uzun süre olduğunu keşfeder.

İş parçacığı daha sonra serbest havuzda kaç bağlantı olduğunu denetler ve bunu **Minimum connections** özelliğinin değeriyle karşılaştırır. Havuz iki bağlantı içerdiğinden ve **Minimum connections** 1 olarak ayarlandığından, bağlantı yöneticisi c1' yi kapatır.

Bakım iş parçacığı şimdi c2' e bakar. Bu, **Unused timeout** özelliğinin değerinden daha uzun bir süredir serbest bağlantı havuzunda bulunuyor. Ancak, c2 kapatıldığında serbest bağlantı havuzu belirlenmiş bağlantı sayısı alt sınırından az kalacağı için, bağlantı yöneticisi c2 ' i yalnız bırakır.

JMS bağlantıları ve IBM MQ
IBM MQ ' in JMS sağlayıcısı olarak kullanımına ilişkin bilgiler.

Bağ tanımları iletimini kullanma

Bir bağlantı üreticisi bağ tanımları iletimini kullanacak şekilde yapılandırıldıysa, her JMS bağlantısı IBM MQile bir etkileşim kurar (**hconn**olarak da bilinir). Etkileşim, kuyruk yöneticisiyle iletişim kurmak için işlemler arası iletişimi (ya da paylaşılan belleği) kullanır.

İstemci iletimini kullanma

Bir IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticisi istemci iletimini kullanacak şekilde yapılandırıldığında, o üreticiden oluşturulan her bağlantı IBM MQile yeni bir etkileşim (**hconn**olarak da bilinir) oluşturur.

IBM MQ ileti alışverişi sağlayıcısı normal kipini kullanarak bir kuyruk yöneticisine bağlanan bağlantı üreticileri için, bağlantı üreticisinden yaratılan birden çok JMS bağlantısının IBM MQile bir TCP/IP bağlantısını paylaşması mümkündür. Daha fazla bilgi için bkz. “IBM MQ classes for JMS içinde TCP/IP bağlantısının paylaşılması” sayfa 304.

Herhangi bir zamanda JMS bağlantıları tarafından kullanılan istemci kanalı sayısı üst sınırını saptamak için, aynı kuyruk yöneticisini işaret eden tüm bağlantı üreticileri için *Bağlantı sayısı üst sınırı* özelliğinin değerini ekleyin.

Örneğin, aynı IBM MQ kanalını kullanarak aynı IBM MQ kuyruk yöneticisine bağlanacak şekilde yapılandırılmış iki bağlantı üreticiniz (`jms/CF1` ve `jms/CF2`) olduğunu varsayın.

Bu üreticiler varsayılan bağlantı havuzu özelliklerini kullanıyorlar; bu, *Bağlantı sayısı üst sınırı* değerinin 10 olarak ayarlandığı anlamına gelir. Tüm bağlantılar aynı anda jms/CF1 ve jms/CF2 ' den kullanılıyorsa, uygulama sunucusu ile IBM MQarasında 20 etkileşim olur.

Bağlantı üreticisi IBM MQ ileti alışverişi sağlayıcısı normal kipini kullanarak kuyruk yöneticisine bağlanırsa, uygulama sunucusu ile bu bağlantı üreticileri için kuyruk yöneticisi arasında var olabilecek TCP/IP bağlantısı sayısı üst sınırı şöyledir:

```
20/the value of SHARECNV for the IBM MQ channel
```

Bağlantı üreticisi IBM MQ ileti alışverişi sağlayıcısı geçiş kipi kullanılarak bağlanacak şekilde yapılandırıldıysa, bu bağlantı üreticileri için uygulama sunucusu ile IBM MQ arasındaki TCP/IP bağlantısı sayısı üst sınırı 20 olur (iki üreticiye ilişkin bağlantı havuzlarındaki her JMS bağlantısı için bir adet).

İlgili kavramlar

“IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma” sayfa 78
IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , IBM MQile verilen Java ileti alışverişi sağlayıcılarıdır. JMS ve Jakarta Messaging belirtimlerinde tanımlanan arabirimleri gerçekleştirmenin yanı sıra, bu ileti alışverişi sağlayıcıları Java ileti sistemi API 'sine iki uzantı kümesi ekler.

Java SE ortamında nesne havuzu oluşturma

Java SE (ya da Spring gibi başka bir çerçeveyle) ile programlama modelleri son derece esnektir. Bu yüzden tek bir havuz oluşturma stratejisi her şeye uymuyor. Herhangi bir havuz oluşturma biçimi yapabilen bir çerçeve olup olmadığını düşünmelisiniz, örneğin, Bahar.

Aksi takdirde, uygulama mantığı bunu kaldırabilir. Uygulamanın kendisinin ne kadar karmaşık olduğunu kendinize sorun. En iyisi, uygulamayı anlamak ve ileti sistemi bağlanırlığından ne gerektirdiğini anlamaktır. Uygulamalar genellikle temel JMS API 'si çevresinde kendi sarıcı kodlarında da yazılır.

Bu çok mantıklı bir yaklaşım olsa da ve karmaşıklığı gizleyebilse de, sorunlara yol açılabileceğini unutmamaya değer. Örneğin, sık olarak adlandırılan soysal bir `getMessage()` yöntemi, yalnızca tüketicileri açıp kapatmamalıdır.

Dikkate almanız gereken noktalar:

- Uygulamanın IBM MQ' e ne kadar süre erişmesi gerekecek? Her zaman ya da ara sıra.
- Mesajlar ne sıklıkta gönderilecek? Daha az sıklıkta, IBM MQ ile tek bir bağlantı paylaşılabilir.
- Bağlantının kesilmesi kural dışı durumu genellikle, havuza yollanmış bir bağlantının yeniden yaratılması gerekmesinin işaretidir. Peki ya:
  - Güvenlik kural dışı durumları ya da anasistem kullanılamıyor
  - Kuyruk tam kural dışı durumları
- Bağlantı kopuk bir kural dışı durum oluşursa, havuzdaki diğer serbest bağlantılara ne olması gerekir? Kapatılıp yeniden mi yaratılmalılar?
- TLS kullanılıyorsa, örneğin, tek bir bağlantının ne kadar süre açık kalmasını istiyorsunuz?
- Havuza yollanmış bir bağlantı, kuyruk yöneticisi tarafından saptanacak ve bağlantıyı geriye doğru izleyebilecek şekilde kendini nasıl tanıtır?

Havuzlama için tüm JMS nesnelerini göz önünde bulundurmanız ve mümkün olduğunda bu nesneyi havuza almanız gerekir. Nesneler şunlardır:

- JMS bağlantılar
- Oturum
- Bağlamlar
- Tüm farklı türlerin üreticileri ve tüketicileri

İstemci iletimi kullanılırken, JMS bağlantıları, oturumları ve bağlamları, IBM MQ kuyruk yöneticisiyle iletişim kurarken yuvaları kullanır. Bu nesneleri havuza yollayarak, tasarruflar kuyruk yöneticisine gelen IBM MQ bağlantılarının (`hConns`) sayısında ve kanal eşgörünümlerinin sayısında azalma olur.

Kuyruk yöneticisine bağ tanımları iletimi kullanıldığında ağ katmanı tamamen kaldırılır. Ancak, birçok uygulama, daha yüksek düzeyde kullanılabilir ve iş yükü dengeli bir yapılandırma sağlamak için istemci aktarımını kullanır.

JMS üreticileri ve tüketicileri kuyruk yöneticisinde açık hedefler. Daha az sayıda kuyruk ya da konu açılırsa ve uygulamanın birden çok bölümü bu nesneleri kullanıyorsa, bunları havuza iletme yararlı olabilir.

IBM MQ perspektifinden, bu işlem bir MQOPEN ve MQCLOSE işlemleri dizisini kaydeder.

Bağlantılar, oturumlar ve bağlamlar

Bu nesnelerin tümü IBM MQ bağlantı tanıtıcılarını kuyruk yöneticisine kapsüller ve bir `ConnectionFactory`' den oluşturulur. Tek bir bağlantı üreticisinden oluşturulan bağlantı sayısını ve diğer nesneleri belirli bir sayıyla sınırlamak için uygulamaya mantık ekleyebilirsiniz.

Oluşturulan bağlantıları içermek için uygulamada basit bir veri yapısı kullanabilirsiniz. Bu veri yapılarından birini kullanması gereken uygulama kodu, kullanılacak bir nesneyi *dışarı alabilir* .

Aşağıdaki etkenleri göz önünde bulundurun:

- Bağlantılar ne zaman havuzdan kaldırılmalıdır? Genellikle, bağlantıda bir kural dışı durum dinleyicisi oluşturun. Bir kural dışı durumu işlemek için o dinleyici çağrıldığında, bağlantıyı ve o bağlantıdan yaratılan oturumları yeniden yaratmanız gerekir.
- İş yükü dengeleme için bir CCDT kullanılırsa, bağlantılar farklı kuyruk yöneticilerine gidebilir. Bu, havuz oluşturma gereksinimleri için geçerli olabilir.

JMS belirtiminin, birden çok iş parçacığının bir oturuma ya da bağlama aynı anda erişmesi için bir programlama hatası olduğunu belirttiğini unutmayın. IBM MQ JMS kodu, iş parçacıklarının işlenmesinde titiz olmayı dener. Ancak, bir oturumun ya da bağlam nesnesinin aynı anda yalnızca bir iş parçacığı tarafından kullanıldığından emin olmak için uygulamaya mantık eklemelisiniz.

Üreticiler ve tüketiciler

Yaratılan her üretici ve tüketici, kuyruk yöneticisinde bir hedef açar. Aynı hedef çeşitli görevler için kullanılacaksa, tüketici ya da üretici nesnelerini açık tutmak mantıklı olur. Yalnızca tüm iş yapıldığında nesneyi kapatın.

Bir hedefin açılması ve kapatılması kısa işlemlerdir, ancak sık sık yapıldığında geçen süre bir arada sayılabilir.

Bu nesnelerin kapsamı, yaratıldığı oturum ya da bağlam içindedir; bu nedenle, nesnelerin bu kapsamda tutulması gerekir. Genel olarak, uygulamalar bu oldukça kolay olacak şekilde yazılır.

İzleme

Uygulamalar nesne havuzlarını nasıl izleyecek? Bunun cevabı büyük ölçüde uygulanan havuzlama çözümünün karmaşıklığı tarafından belirlenir.

Bir JavaEE havuzlama uygulamasını göz önünde bulundurursanız, aşağıdakiler de içinde olmak üzere çok sayıda seçenek vardır:

- Havuzların yürürlükteki büyüklüğü
- Nesnelerin harcadığı süre
- Havuzların temizlenmesi
- Bağlantıların yenilenmesi

Yeniden kullanılan tek bir oturumun kuyruk yöneticisinde nasıl göründüğünü de göz önünde bulundurmalısınız. Yararlı olabilecek uygulamayı (appNamegibi) tanımlamak için bağlantı üreticisi özellikleri vardır.

"IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma" sayfa 78
IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , IBM MQile verilen Java ileti alışverişi sağlayıcılarıdır. JMS ve Jakarta Messaging belirtimlerinde tanımlanan arabirimleri gerçekleştirmenin yanı sıra, bu ileti alışverişi sağlayıcıları Java ileti sistemi API 'sine iki uzantı kümesi ekler.

IBM MQ classes for JMS içinde TCP/IP bağlantısının paylaşılması
Tek bir TCP/IP bağlantısını paylaşmak için birden çok MQI kanalı eşgörünümü yapılabilir.

Aynı Java yürütme ortamı içinde çalışan ve CLIENT iletimini kullanarak bir kuyruk yöneticisine bağlanmak için IBM MQ classes for JMS ya da IBM MQ kaynak bağdaştırıcısını kullanan uygulamalar, bir kanal eşgörünümünü paylaşmak için yapılabilir.

Bir kanal, **SHARECNV** parametresi 1değerinden büyük bir değere ayarlanarak tanımlanırsa, bu etkileşim sayısı bir kanal örneğini paylaşabilir. Bu işlevi kullanmak üzere bir bağlantı üreticisini ya da etkinleştirme belirtimini etkinleştirmek için **SHARECONVALLOWED** özelliğini YESdeğerine ayarlayın.

Bir JMS uygulaması tarafından yaratılan her JMS bağlantısı ve JMS oturumu, kuyruk yöneticisiyle kendi etkileşmesini yaratır.

Bir etkinleştirme belirtimi başlatıldığında, IBM MQ kaynak bağdaştırıcısı, kullanılacak etkinleştirme belirtimi için kuyruk yöneticisiyle bir etkileşim başlatır. Etkinleştirme belirtimiyle ilişkili sunucu oturumu havuzundaki her sunucu oturumu, kuyruk yöneticisiyle de bir etkileşim başlatır.

SHARECNV özniteliği, bağlantı paylaşımına yönelik en iyi çalışma yaklaşımıdır. Bu nedenle, IBM MQ classes for JMSile 0 'dan büyük bir **SHARECNV** değeri kullanıldığında, yeni bir bağlantı isteğinin her zaman önceden kurulmuş bir bağlantıyı paylaştığı garanti edilmez.

TCP/IP bağlantılarının paylaşılma şekli

TCP/IP bağlantılarını paylaşmak için kullanılabilecek iki strateji vardır:

GLOBAL stratejisi
Bu strateji, TCP/IP bağlantılarının paylaşılmasına ilişkin varsayılan stratejidir. Herhangi bir JMS bağlantısı ya da oturumu, uygun bir TCP/IP bağlantısında etkileşim kullanabilir. Uygunluk, anasistem adresi, kapı numarası, kullanıcı kimliği ve parola ve TLS/SSL parametreleri gibi faktörlerle belirlenir.

TCP/IP bağlantılarının paylaşılmasına ilişkin bu yaklaşım, kullanılmakta olan kanal eşgörünümlerinin sayısını en aza indirir; ancak, TCP/IP bağlantılarının genel havuzuna erişim için gereken çekişme pahasına.

CONNECTION stratejisi
Bu stratejiyle, kanal eşgörünümleri yalnızca ilgili JMS nesneleri arasında paylaşılır. Özellikle, bir JMS bağlantısı oluşturulduğunda, bunun için bir kanal örneği oluşturulur ve bu kanal örneğinde ek etkileşimler yalnızca o JMS bağlantısı tarafından oluşturulan JMS oturumları için kullanılabilir.

SHARECNV özniteliğinin belirttiği etkileşimden daha fazla etkileşim yaratılırsa, yalnızca özgün JMS bağlantısı tarafından yaratılan JMS oturumları tarafından kullanılabilecek yeni bir kanal eşgörünümü yaratılır.

Kanal örneklerini paylaşmaya yönelik bu yaklaşım, çok daha fazla kanal örneği gerektirecek pahasına konuşmalar için çekişmeyi azaltır.

Kanal eşgörünümü paylaşım stratejisinin belirtik olarak belirtilmesi

V 9.3.2

Uygulamalar yeniden bağlanamıyorsa, varsayılan olarak GLOBAL stratejisi kullanılır. Yeniden bağlanabilir uygulamalar her zaman CONNECTION stratejisini kullanır.

IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingkullanan uygulamalar için CONNECTION stratejisi, uygulama genelinde yeniden bağlanamayan uygulamalar için etkinleştirilebilir. `com.ibm.mq.jms.channel.sharing` sistem özelliğini `CONNECTION`değerine ayarlayarak CONNECTION stratejisini etkinleştirebilirsiniz. Bu değer büyük ve küçük harfe duyarlı değildir ve `CONNECTION` dışındaki herhangi bir değer yoksayılır.

`com.ibm.mq.jms.channel.sharing` sistem özelliğini aşağıdaki yollardan biriyle ayarlayabilirsiniz:

- "-D" komut satırı seçeneğini kullanarak JVM kullanıma hazırlamanın bir parçası olarak özelliği ayarlayın:

```
-Dcom.ibm.mq.jms.channel.sharing=CONNECTION
```

- `System.setProperty()` komutunu kullanarak IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging ürününü kullanmadan önce özelliği ayarlayın

Küresel paylaşım stratejisine ilişkin kanal örneklerinin sayısının hesaplanması

Bir uygulama tarafından oluşturulan kanal örneği sayısı üst sınırını belirlemek için aşağıdaki formülleri kullanın:

Etkinleştirme belirtimleri

Kanal örneği sayısı = (*maxPoolDepth_value* + 1)/ *SHARECNV_value*

Burada *maxPoolDepth_value* , **maxPoolDepth** özelliğinin ve *SHARECNV_value* , etkinleştirme belirtimi tarafından kullanılan kanaldaki **SHARECNV** özelliğinin değeridir.

Diğer JMS uygulamaları

Kanal örneği sayısı = (*jms_connections* + *jms_sessions*)/ *SHARECNV_value*

Burada *jms_connections* , uygulama tarafından oluşturulan bağlantı sayısıdır; *jms_sessions* , uygulama tarafından oluşturulan JMS oturumlarının sayısıdır ve *SHARECNV_value* , etkinleştirme belirtimi tarafından kullanılan kanaldaki **SHARECNV** özelliğinin değeridir.

CONNECTION paylaşım stratejisi için kanal eşgörünümlerinin sayısının hesaplanması

Kanal yönetim ortamlarının sayısı, uygulamadaki JMS bağlantıları arasında JMS oturumlarının dağıtımına bağlıdır.

JMS bağlantısı için bir etkileşimin ve bu JMS bağlantısı altında her JMS oturumu için bir etkileşimin olmasına izin verin, ardından **SHARECNV** değerine göre bölün ve yukarı yuvarlayarak. Bu hesaplama, o JMS bağlantısı için gereken kanal eşgörünümlerini verir.

Aynı ilke aktivasyon özelliklerine uygulanabilir. Etkinleştirme belirtimini JMS bağlantısı ve **maxPoolDepth** özelliğini JMS oturumu sayısı olarak kabul edin.

Örnekler

Aşağıdaki örneklerde, IBM MQ classes for JMS ya da IBM MQ kaynak bağdaştırıcısı kullanılarak uygulamalar tarafından bir kuyruk yöneticisinde yaratılan kanal eşgörünümlerinin sayısını hesaplamak için formüllerin nasıl kullanılacağı gösterilmektedir.

JMS uygulaması örneği

JMS uygulaması bağlantısı, CLIENT iletimini kullanarak bir kuyruk yöneticisine bağlanır ve bir JMS bağlantısı ve üç JMS oturumu yaratır. Uygulamanın kuyruk yöneticisine bağlanmak için kullandığı kanalda **SHARECNV** özelliği 10 değerine ayarlıdır. Uygulama çalışırken, uygulama ile kuyruk yöneticisi ve bir kanal eşgörünümü arasında dört etkileşim vardır. Dört konuşmanın hepsi kanal örneğini paylaşıyor.

Etkinleştirme belirtimi örneği

Etkinleştirme belirtimi, CLIENT iletimini kullanarak bir kuyruk yöneticisine bağlanır. Etkinleştirme belirtimi, **maxPoolDepth** özelliği 10 olarak ayarlanarak yapılandırılır. Etkinleştirme belirtiminin kullanmak üzere yapılandırıldığı kanal için **SHARECNV** özelliği 10 olarak ayarlanmış. Etkinleştirme belirtimi çalışırken ve 10 iletiyi eşzamanlı olarak işlerken, etkinleştirme belirtimi ile kuyruk yöneticisi arasındaki etkileşim sayısı 11 'dir (sunucu oturumları için 10 etkileşim ve etkinleştirme belirtimi için bir etkileşim). Etkinleştirme belirtimi tarafından kullanılan kanal eşgörünümlerinin sayısı 2 'dir.

Etkinleştirme belirtimi örneği

Etkinleştirme belirtimi, CLIENT iletimini kullanarak bir kuyruk yöneticisine bağlanır. Etkinleştirme belirtimi, **maxPoolDepth** özelliği 5 olarak ayarlanarak yapılandırılır. Etkinleştirme belirtiminin kullanmak üzere yapılandırıldığı kanalda **SHARECNV** özelliği 0 olarak ayarlanmış. Etkinleştirme belirtimi çalışırken ve 5 iletiyi eşzamanlı olarak işlerken, etkinleştirme belirtimi ile kuyruk yöneticisi arasındaki etkileşim sayısı 6 'dır (sunucu oturumları için beş etkileşim ve etkinleştirme belirtimi için bir etkileşim). Kanaldaki **SHARECNV** özelliği 0 olarak ayarlandığından, her etkileşim kendi kanal örneğini kullandığından, etkinleştirme belirtimi tarafından kullanılan kanal eşgörünümlerinin sayısı 6 'dır.

İlgili görevler

"WebSphere Application Server ile IBM MQ arasında yaratılan TCP/IP bağlantılarının sayısının belirlenmesi" sayfa 482

Paylaşım etkileşimleri özelliğini kullanarak birden çok etkileşim MQI kanalı eşgörünümlerini paylaşabilir; bu, TCP/IP bağlantısı olarak da bilinir.

IBM MQ classes for JMS içinde istemci bağlantıları için bir kapı aralığı belirtme

Uygulamanızın bağlanabileceği kapı aralığını belirtmek için LOCALADDRESS özelliğini kullanın.

Bir IBM MQ classes for JMS uygulaması istemci kipinde bir IBM MQ kuyruk yöneticisine bağlanmaya çalıştığında, güvenlik duvarı yalnızca belirtilen kapılardan ya da kapı aralığından kaynaklanan bağlantılara izin verebilir. Bu durumda, uygulamanın bağlanabileceği bir kapı ya da kapı aralığını belirtmek için ConnectionFactory, QueueConnectionFactory ya da TopicConnectionFactory nesnesinin LOCALADDRESS özelliğini kullanabilirsiniz.

LOCALADDRESS özelliğini IBM MQ JMS yönetim aracını kullanarak ya da bir JMS uygulamasında setLocalAddress () yöntemini çağırarak ayarlayabilirsiniz. Aşağıda, bir uygulama içinden özelliğin ayarlanmasına ilişkin bir örnek verilmiştir:

```
mqConnectionFactory.setLocalAddress("192.0.2.0(2000,3000)");
```

Uygulama daha sonra bir kuyruk yöneticisine bağlandığında, uygulama 192.0.2.0(2000) ile 192.0.2.0(3000) aralığında yerel bir IP adresine ve kapı numarasına bağlanır.

Birden çok ağ arabirimi olan bir sistemde, bağlantı için hangi ağ arabiriminin kullanılması gerektiğini belirtmek için LOCALADDRESS özelliğini de kullanabilirsiniz.

Bir aracıya gerçek zamanlı bağlantı için, LOCALADDRESS özelliği yalnızca çoklu yayın kullanıldığında anlamlıdır. Bu durumda, bir bağlantı için hangi yerel ağ arabiriminin kullanılması gerektiğini belirtmek için özelliği kullanabilirsiniz, ancak özelliğin değeri bir kapı numarası ya da bir kapı numarası aralığı içermemelidir.

Kapı aralığını kısıtlarsanız bağlantı hataları oluşabilir. Bir hata oluşursa, MQRC_Q_MGR_NOT_KULLANILABILIR IBM MQ neden kodunu ve aşağıdaki iletiyi içeren yerleşik bir MQException ile JMSException yayınlanır:

```
LOCAL_ADDRESS_PROPERTY kısıtlamaları nedeniyle yuva bağlantısı girişimi reddedildi
```

Belirtilen aralıktaki tüm kapılar kullanımdaysa ya da belirtilen IP adresi, anasistem adı ya da kapı numarası geçerli değilse (örneğin, negatif bir kapı numarası) bir hata oluşabilir.

IBM MQ classes for JMS , bir uygulama için gerekenden farklı bağlantılar oluşturabileceğinden, her zaman bir kapı aralığı belirtmeyi düşünebilirsiniz. Genel olarak, bir uygulama tarafından oluşturulan her oturum bir kapı gerektirir ve IBM MQ classes for JMS üç ya da dört ek kapı gerektirebilir. Bir bağlantı hatası oluşursa, kapı aralığını artırın.

IBM MQ classes for JMSiçinde varsayılan olarak kullanılan bağlantı havuzlama, kapıların yeniden kullanılabileceği hız üzerinde bir etkiye sahip olabilir. Sonuç olarak, kapılar serbest bırakılırken bir bağlantı hatası oluşabilir.

IBM MQ classes for JMS içinde kanal sıkıştırma
IBM MQ classes for JMS uygulaması, bir ileti üstbilgisini ya da verilerini sıkıştırmak için IBM MQ olanaklarını kullanabilir.

Bir IBM MQ kanalında akan verilerin sıkıştırılması, kanalın performansını artırabilir ve ağ trafiğini azaltabilir. IBM MQile verilen işlevi kullanarak, ileti kanallarında ve MQI kanallarında akan verileri sıkıştırabilirsiniz. Her iki kanal tipinde de, üstbilgi verilerini ve ileti verilerini birbirinden bağımsız olarak sıkıştırabilirsiniz. Varsayılan olarak, kanalda veri sıkıştırılmaz.

IBM MQ classes for JMS uygulaması, bir java.util.Collection nesnesi yaratarak bir bağlantıdaki üstbilgiyi ya da ileti verilerini sıkıştırmada kullanılabilecek teknikleri belirtir. Her sıkıştırma tekniği, derlemdeki bir Tamsayı nesnesidir ve uygulamanın derleme nesnesine sıkıştırma tekniklerini ekleme sırası, uygulama bağlantıyı yarattığında sıkıştırma tekniklerinin kuyruk yöneticisiyle kararlaştırıldığı sıradır. Uygulama, üstbilgi verileri için setHdrCompList() yöntemini ya da ileti verileri için setMsgCompList() yöntemini çağırarak derlemi bir ConnectionFactory nesnesine geçirebilir. Uygulama hazır olduğunda bağlantıyı yaratabilir.

Aşağıdaki kod parçaları, açıklanan yaklaşımı göstermektedir. İlk kod parçası, üstbilgi veri sıkıştırmasını nasıl uygulayacağınızı gösterir:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(WMQConstants.WMQ_COMPHDR_SYSTEM));
.
.
.
((MQConnectionFactory) cf).setHdrCompList(headerComp);
.
.
.
connection = cf.createConnection();
```

İkinci kod parçası, ileti verileri sıkıştırmasını nasıl uygulayacağınızı gösterir:

```
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();
```

İkinci örnekte, sıkıştırma teknikleri bağlantı oluşturulduğunda RLE, daha sonra ZLIBHIGH sırasıyla kararlaştırılır. Seçilen sıkıştırma tekniği, Connection nesnesinin yaşam süresi boyunca değiştirilemez. Bir bağlantıda sıkıştırma kullanmak için, Connection nesnesi yaratılmadan önce setHdrCompList() ve setMsgCompList() yöntemleri çağrılmalıdır.

İletileri IBM MQ classes for JMS içine zamanuyumsuz olarak koyma
Olağan durumda, bir uygulama bir hedefe ileti gönderdiğinde, uygulamanın kuyruk yöneticisinin isteği işlediğini doğrulamasını beklemesi gerekir. Bazı durumlarda iletileri zamanuyumsuz olarak yerleştirmeyi seçerek ileti başarımını artırabilirsiniz. Bir uygulama bir iletiyi zamanuyumsuz olarak yerleştirdiğinde, kuyruk yöneticisi her çağrıya ilişkin başarıyı ya da başarısızlığı döndürmez; bunun yerine belirli aralıklarla hata olup olmadığını denetleyebilirsiniz.

Bir hedefin, kuyruk yöneticisinin iletiyi güvenli bir şekilde alıp almadığını belirlemeden, denetimi uygulamaya döndürüp döndürmeyeceği aşağıdaki özelliklere bağlıdır:

JMS Hedef özellik PUTASYNCAL, (kısa ad-PAALD).
PUTASYNCALALLOWED, temel kuyruk ya da JMS hedefinin temsil ettiği konu tarafından bu seçeneğe izin veriliyorsa, JMS uygulamalarının iletileri zamanuyumsuz olarak yerleştirip yerleştiremeyeceğini denetler.

IBM MQ kuyruk ya da konu özelliği DEFPRESP (Varsayılan koyma yanıtı tipi).
DEFPRESP, iletileri kuyruğa koyan ya da iletileri konuya yayınlayan uygulamaların zamanuyumsuz koyma işlevini kullanıp kullanamayacağını belirler.

Aşağıdaki çizelge, PUTASYNCALLU ve DEFPRESP özelliklerine ilişkin olası değerleri ve zamanuyumsuz koyma işlevini etkinleştirmek için kullanılan değer birleşimlerini göstermektedir:

Çizelge 46. PUTASYNCALFED ve DEFPRESP özelliklerinin, iletilerin bir hedefe zamanuyumsuz olarak konup konmayacağını belirlemek için nasıl birleştirildiğini.

IBM MQ kuyruk özelliği	PUTASYNCALBYK = NO	PUTASYNCALBYK = EVET	Zamanuyumsuz koyma işlevi etkinleştirildi
DEFPRESP=SYNC	Zamanuyumsuz koyma işlevi etkinleştirilmedi	Zamanuyumsuz koyma işlevi etkinleştirildi	PUTASYNCAL, = AS_DEST ya da AS_Q_DEF ya da AS_T_DEF
DEFPRESP=ASYNC	Zamanuyumsuz koyma işlevi etkinleştirilmedi	Zamanuyumsuz koyma işlevi etkinleştirildi	PUTASYNCAL, = AS_DEST ya da AS_Q_DEF ya da AS_T_DEF

Tabloda gösterildiği gibi IBM MQ-JMS Hedef özelliğini "NO" ya da "YES" olarak belirterek davranışı değiştirebilirsiniz, ancak bu özellik JVM **SystemProperty** ve değeri kullanılarak tüm Java Virtual Machine için de geçersiz kılınabilir:

```
com.ibm.mq.cfg.Channels.Put1DefaultAlwaysSync=Y
```

İşlem uygulanan bir oturumda gönderilen iletiler için uygulama, kuyruk yöneticisinin `commit()'u` çağırdığında iletileri güvenli bir şekilde alıp almadığını belirler.

Bir uygulama işlemli bir oturumda kalıcı iletiler gönderirse ve iletilerden biri ya da daha fazlası güvenli bir şekilde alınmazsa, işlem kesinleştirilemez ve kural dışı durum üretir. Ancak, bir uygulama işlem yapılan bir oturumda kalıcı olmayan iletiler gönderirse ve bir ya da daha fazla ileti güvenli bir şekilde alınmazsa, hareket başarıyla kesinleştirilir. Uygulama, kalıcı olmayan iletilerin güvenli bir şekilde ulaşmamasına ilişkin herhangi bir geribildirim almıyor.

İşlem yapılmayan bir oturumda gönderilen kalıcı olmayan iletiler için, *ConnectionFactory* nesnesinin SENDCHECKCOUNT özelliği, IBM MQ classes for JMS kuyruk yöneticisinin iletileri güvenli bir şekilde aldığını denetlemeden önce kaç iletinin gönderileceğini belirtir.

Bir denetim bir ya da daha çok iletinin güvenli bir şekilde alınmadığını ve uygulamanın bağlantıya bir kural dışı durum dinleyicisi kaydettirdiğini keşfederse, IBM MQ classes for JMS kural dışı durum dinleyicisinin `onException()` yöntemini uygulamaya bir JMS kural dışı durumu geçirmesi için çağırır.

JMS kural dışı durumu JMSWMQ0028 hata kodunu içeriyor ve bu kod şu iletiyi görüntüler:

```
At least one asynchronous put message failed or gave a warning.
```

JMS kural dışı durumu, daha fazla ayrıntı sağlayan bağlantılı bir kural dışı duruma da sahiptir. SENDCHECKCOUNT özelliğinin varsayılan değeri sıfırdır; bu, böyle bir denetim yapılmadığı anlamına gelir.

Bu eniyileme, istemci kipinde bir kuyruk yöneticisine bağlanan ve bir dizi iletiyi hızlı bir şekilde göndermesi gereken, ancak gönderilen her ileti için kuyruk yöneticisinden anında geribildirim gerektirmeyen bir uygulamaya yarar sağlar. Ancak, bir uygulama bağ tanımlama kipinde bir kuyruk yöneticisine bağlansa da bu eniyilemeyi kullanmaya devam edebilir, ancak beklenen performans avantajı o kadar da iyi değildir.

Not: Bir hareket altında ileti göndermek için tanımlanmamış bir **MessageProducer** kullanıyorsanız, varsayılan olarak iletiler zamanuyumsuz koyma mekanizması kullanılarak kuyruğa yerleştirilir.

JMS API, **MessageProducer** ' nin bir Hedef belirtilmeden, sözdizimi kullanılarak oluşturulmasına izin verdiğinden bu durum oluşabilir:

```
javax.jms.MessageProducer messageProducer =  javax.jms.Session.createProducer(null);
  messageProducer.send(Destination destination, Message message, int deliveryMode, int priority, long
timeToLive);
```

Bu senaryoda, **MessageProducer** oluşturulduğunda ileti önceden değil, gönderildiğinde JMS Hedefi sağlanır. IBM MQ API açısından bu, iletiyi kuyruğa koymak için bir MQPUT1 yayınlanmasına neden olur.

Bunu bir IBM MQ syncpoint altında yaparsanız, yani (JMS terminolojisinde) iletiyi bir hareketin altına koyarsanız, işlemli JMS Oturumu kullanarak ya da XASession API 'si kullanılarak IBM MQ classes for JMS zamanuyumsuz koyma yöntemini kullanmaya geçer.

IBM MQ classes for JMS ile önden okuma özelliğinin kullanılması
IBM MQ tarafından sağlanan önden okuma işlevi, bir işlem dışında alınan kalıcı olmayan iletilerin, bir uygulama tarafından istenmeden önce IBM MQ classes for JMS ' e gönderilmesini sağlar. IBM MQ classes for JMS , iletileri bir iç arabellekte saklar ve uygulama bunları istediğinde iletileri uygulamaya iletir.

Bir hareketin dışındaki bir hedeften ileti almak için MessageConsumers ya da MessageListeners kullanan IBM MQ classes for JMS uygulamaları, önden okuma işlevini kullanabilir. Önden okuma özelliğinin kullanılması, bu nesneleri kullanan uygulamaların ileti aldıklarında daha yüksek başarımdan yararlanmalarını sağlar.

MessageConsumers ya da MessageListeners kullanan bir uygulamanın önden okuma özelliğini kullanıp kullanamayacağı aşağıdaki özelliklere bağlıdır:

JMS hedef özelliği READAHEADALLOWED (kısa ad-RAALD).
READAHEADALLOWED, JMS hedefinin gösterdiği temel kuyruk ya da konu bu seçeneğe izin veriyorsa, bir işlemin dışındaki kalıcı olmayan iletileri almak ya da göz atarken JMS uygulamalarının önden okuma özelliğini kullanıp kullanamayacağını denetler.

IBM MQ kuyruk ya da konu özelliği DEFREADA (Varsayılan önden okuma).
DEFREADA, bir işlemin dışında kalıcı olmayan iletileri alan ya da bu iletilere göz atan uygulamaların önden okuma özelliğini kullanıp kullanamayacağını belirtir.

Aşağıdaki çizelge, READAHEADALLOWED ve DEFREADA özelliklerine ilişkin olası değerleri ve önden okuma işlevini etkinleştirmek için kullanılan değer birleşimlerini göstermektedir:

Çizelge 47. READAHEADALLOWED ve DEFREADA özelliklerinin, bir işlem dışında kalıcı olmayan iletiler alınırken ya da bu iletilere göz atarken okuma özelliğinin kullanılıp kullanılmadığını belirlemek için nasıl birleştirildiğini.

IBM MQ kuyruk özelliği	READAHEADALLOWED = EVET	READAHEADALLOWED = HAYıR	AS_DEST ya da AS_Q_DEF ya da AS_T_DEF
DEFREADA = HAYıR	Önden okuma işlevi etkin	Önden okuma işlevi etkinleştirilmedi	Önden okuma işlevi etkinleştirilmedi
DEFREADA = EVET	Önden okuma işlevi etkin	Önden okuma işlevi etkinleştirilmedi	Önden okuma işlevi etkin
DEFREADA = DEVRE Dışı	Önden okuma işlevi etkinleştirilmedi	Önden okuma işlevi etkinleştirilmedi	Önden okuma işlevi etkinleştirilmedi

Önden okuma işlevi etkinleştirilirse, bir MessageConsumer ya da MessageListener bir uygulama tarafından oluşturulduğunda, IBM MQ classes for JMS MessageConsumer ya da MessageListener tarafından izlenmekte olan hedef için bir iç arabellek yaratır. Her MessageConsumer ya da MessageListeneriçin bir iç arabellek vardır. Uygulama aşağıdaki yöntemlerden birini çağırdığında, kuyruk yöneticisi IBM MQ classes for JMS ' e kalıcı olmayan iletiler göndermeye başlar:

- MessageConsumer.receive()

- `MessageConsumer.receive(long timeout)`
- `MessageConsumer.receiveNoWait()`
- `Session.setMessageListener(MessageListener listener)`

IBM MQ classes for JMS , uygulamanın yaptığı yöntem çağrısıyla ilk iletiyi otomatik olarak uygulamaya geri döndürür. Diğer kalıcı olmayan iletiler, IBM MQ classes for JMS tarafından hedef için oluşturulan iç arabellekte saklanır. Uygulama sonraki iletinin işlenmesini istediğinde, IBM MQ classes for JMS iç arabellekte sonraki iletiyi döndürür.

IBM MQ classes for JMS , iç arabellek boş olduğunda kuyruk yöneticisinden daha fazla kalıcı olmayan ileti ister.

IBM MQ classes for JMS tarafından kullanılan iç arabellek, bir uygulama bir `MessageConsumer`'yi ya da `MessageListener` ' in ilişkilendirildiği JMS oturumunu kapattığında silinir.

`MessageConsumer`için, iç arabellekteki işlenmemiş iletiler kaybolur.

`MessageListener`skullanılırken, iç arabellekteki iletilere ne olacağı, JMS hedef özelliği READAHEADCLOSEPOLICY (kısa ad-RACP) bağlıdır. Özelliğin varsayılan değeri DELIVER_ALL değeridir; diğer bir deyişle, `MessageListener` öğesini yaratmak için kullanılan JMS oturumu, iç arabellekteki tüm iletiler uygulamaya teslim edilinceye kadar kapanmaz. Özellik DELIVER_CURRENT olarak ayarlanırsa, yürürlükteki ileti uygulama tarafından işlendikten ve iç arabellekteki kalan tüm iletiler atıldıktan sonra JMS oturumu kapatılır.

IBM MQ classes for JMS içindeki yayınları alıkoyma
IBM MQ classes for JMS istemcisi, alıkonan yayınları kullanacak şekilde yapılandırılabilir.

Bir yayınlayıcı, bir yayının bir kopyasının, konuya ilgi duyan gelecekteki abonelere gönderilebilmesi için saklanması gerektiğini belirtebilir. Bu, IBM MQ classes for JMS içinde JMS_IBM_RETAIN tamsayı özelliği 1 değerine ayarlanarak yapılır. com.ibm.msg.client.jms.JmsConstants arabiriminde bu değerler için sabit değerler tanımlanmıştır. Örneğin, *msg*iletisini alıkonan bir yayın olarak ayarlamak için aşağıdaki kodu kullanın:

```
// set as a retained publication
msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION);
```

Artık iletiyi normal olarak gönderebilirsiniz. JMS_IBM_RETAIN, alınan bir iletide de sorgulanabilir. Bu nedenle, alınan bir iletinin alıkonan bir yayın olup olmadığı sorgulanebilir.

IBM MQ classes for JMS içinde XA desteği

JMS , bir JEE kapsayıcısı içinde desteklenen bir hareket yöneticisiyle bağ tanımlarında ve istemci kiplerinde XA uyumlu işlemleri destekler.

Bir uygulama sunucusu ortamında XA işlevine gereksiniminiz varsa, uygulamanızı uygun şekilde yapılandırmanız gerekir. Uygulamaların dağıtımlı hareketleri kullanacak şekilde nasıl yapılandırılacağını öğrenmek için uygulama sunucunuzun kendi belgelerine bakın.

Bir IBM MQ kuyruk yöneticisi, JMSiçin hareket yöneticisi olarak işlev görmez.

JMS iletileri için teslim gecikmesi

JMS 2.0 ya da daha sonraki bir yayın için, ileti gönderirken bir teslim gecikmesi belirtebilirsiniz. Kuyruk yöneticisi, belirtilen teslim gecikmesi geçinceye kadar iletiyi teslim etmez.

Bir uygulama, `MessageProducer.setDeliveryDelay(long deliveryDelay)` ya da `JMSProducer.setDeliveryDelay(long deliveryDelay)`kullanarak bir ileti gönderdiğinde milisaniye cinsinden bir teslim gecikmesi belirtebilir. Bu değer, iletinin gönderildiği zamana eklenir ve diğer herhangi bir uygulamanın bu iletiyi alabileceği en erken zamanı verir.

Teslim gecikmesi, tek bir iç hazırlama kuyruğu kullanılarak gerçekleştirilir. Teslim gecikmesi sıfır olmayan iletiler, teslim gecikmesini ve hedef kuyrukla ilgili bilgileri gösteren bir üstbilgiyle bu kuyruğa yerleştirilir. Teslim gecikme işlemcisi adı verilen kuyruk yöneticisinin bir bileşeni, konaklatma kuyruğundaki iletileri

izler. Bir iletinin teslim gecikmesi tamamlandığında, ileti hazırlama kuyruğundan alınır ve hedef kuyruğa yerleştirilir.

İleti alışverişi istemcileri

IBM MQ teslimat gecikmesi uygulaması yalnızca JMS istemcisini kullandığınızda kullanılabilir. IBM MQile teslimat gecikmesi kullanıyorsanız aşağıdaki kısıtlamalar geçerlidir. Bu kısıtlamalar `MessageProducers` ve `JMSProducer`siçin eşit olarak geçerlidir, ancak `JMSProducers`durumunda `JMSRuntimeExceptions` yayınlanır.

- IBM MQ 8.0'dan önceki bir kuyruk yöneticisine bağlanıldığında `MessageProducer.setDeliveryDelay` ' i sıfır dışında bir değerle çağırma girişimi, MQRC_FUNCTION_NOT_SUPPORTED iletisiyle bir `JMSException` ile sonuçlanır.
- Teslim gecikmesi, MQBND_BIND_NOT_FIXED dışında bir **DEFBIND** değeri olan kümelenmiş hedefler için desteklenmez. Bir `MessageProducer` sıfır olmayan bir teslim gecikmesi ayarına sahipse ve bu gereksinimi karşılamayan bir hedefe gönderme girişiminde bulunulursa, çağrı MQRC_OPTIONS_ERROR iletisiyle bir `JMSException` ile sonuçlanır.
- Daha önce belirlenmiş bir sıfır olmayan teslim gecikmesinden (ya da bunun tersi) daha kısa bir süre için etkin bir değer ayarlama girişimi, MQRC_EXPIRY_ERROR iletisiyle bir `JMSException` ile sonuçlanır. Bu denetim, seçilen tam işlem kümesine bağlı olarak `setTimeToLive` ya da `setDeliveryDelay` ya da `send` yöntemleri çağrılırken gerçekleştirilir.
- Alıkonan yayınların kullanımı ve teslim gecikmesi desteklenmez. İleti `msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION)` kullanılarak alıkondu olarak işaretlendiyse, teslim gecikmesiyle bir ileti yayınlanmaya çalışılması, MQRC_OPTIONS_ERROR iletisiyle bir `JMSException` ile sonuçlanır.
- Teslim gecikmesi ve ileti gruplaması desteklenmez ve bu birleşimi kullanma girişimi, `JMSException` MQRC_OPTIONS_ERROR iletisiyle sonuçlanır.

Teslim gecikmesi olan bir iletinin gönderilmemesi, istemcinin uygun bir hata iletisi (örneğin, kuyruk dolu) içeren bir `JMSException` yayınlamasıyla sonuçlanır. Bazı durumlarda, hata iletisi hedef, konaklatma kuyruğu ya da her ikisi için de geçerli olabilir.

Not: IBM MQ , bir iş birimine ileti koyan uygulamaların, iş birimi kesinleştirilmemiş olsa da aynı iletiyi yeniden almasına izin verir. Bu teknik, iş birimi kesinleştirilinceye kadar ileti hazırlama kuyruğuna yerleştirilmediği ve sonuç olarak hedef hedefe gönderilmediği için teslim gecikmesiyle çalışmaz.

Yetkilendirme

IBM MQ , uygulama sıfır dışında bir teslim gecikmesi olan bir ileti gönderdiğinde özgün hedef üzerinde yetkilendirme denetimleri yapar. Uygulama yetkili değilse, gönderme başarısız olur. Kuyruk yöneticisi, bir iletinin teslim gecikmesinin tamamlandığını saptadığında, hedef kuyruğu açar. Bu noktada hiçbir yetki denetimi gerçekleştirilmez.

SYSTEM.DDELAY.LOCAL.QUEUE

Bir sistem kuyruğu, SYSTEM.DDELAY.LOCAL.QUEUE, teslim gecikmesini uygulamak için kullanılır.

- Multi Çoklu platformlar' DE SYSTEM.DDELAY.LOCAL.QUEUE varsayılan olarak var. Sistem kuyruğunun MAXMSGL ve MAXDEPTH özniteliklerinin beklenen yük için yeterli olması için değiştirilmesi gerekir.
- z/OS IBM MQ for z/OS' DE SYSTEM.DDELAY.LOCAL.QUEUE , hem yerel hem de paylaşılan kuyruklara teslim gecikmesiyle gönderilen iletiler için bir konaklatma kuyruğu olarak kullanılır. z/OSüzerinde, kuyruk yaratılmalı ve MAXMSGL ve MAXDEPTH özniteliklerinin beklenen yük için yeterli olması için tanımlanmalıdır.

Bu kuyruk yaratıldığında, mümkün olan en az sayıda kullanıcının bu kuyruğa erişebilmesi için bu kuyruğun güvenli kılınması gerekir. Kuyruğa erişim yalnızca bakım ve izleme amacıyla olmalıdır.

JMS uygulaması tarafından sıfır olmayan bir teslim gecikmesiyle bir ileti gönderildiğinde, ileti yeni bir ileti tanıtıcısıyla bu kuyruğa yerleştirilir. Özgün ileti tanıtıcısı, iletinin ilinti tanıtıcısına yerleştirilir. Bu ilinti tanıtıcısı, uygulamanın gerektiğinde konaklatma kuyruğundan bir ileti almasına olanak sağlar; örneğin, büyük bir teslim gecikmesi yanlışlıkla kullanıldıysa.

z/OS ile ilgili dikkat edilecek noktalar

z/OS

Sisteminiz z/OSişletim sisteminde çalışıyorsa, teslim gecikmesini kullanmak istiyorsanız dikkate alınması gereken ek noktalar vardır.

Teslim gecikmesi kullanılacaksa, sistem kuyruğu SYSTEM.DDELAY.LOCAL.QUEUE tanımlanmalıdır. Beklenen yükü için yeterli bir depolama sınıfıyla ve INDXTYPE (NONE) ve MSGDLVSQ (FIFO) belirtilerek tanımlanmalıdır. CSQ4INSG JCL ' de sistem kuyruğuna ilişkin örnek bir tanımlama sağlanmıştır (açıklama satırı yapıldı).

Paylaşılan kuyruklar

Teslim gecikmesi, paylaşılan kuyruklara ileti göndermek için desteklenir. Ancak, hedef kuyruğun paylaşılıp paylaşılmamasından bağımsız olarak yalnızca tek bir özel hazırlama kuyruğu kullanılır. Gecikme tamamlandığında, geciken iletiyi hedef paylaşılan kuyruğuna göndermek için özel kuyruğun sahibi olan kuyruk yöneticisinin çalışıyor olması gerekir.

Not: Kalıcı olmayan bir ileti paylaşılan bir kuyruğa teslim gecikmesiyle yerleştirilirse ve hazırlama kuyruğunun sahibi olan kuyruk yöneticisi kapanırsa, özgün ileti kaybolur. Sonuç olarak, paylaşılan bir kuyruğa teslim gecikmesiyle gönderilen kalıcı olmayan iletiler, paylaşılan bir kuyruğa teslim gecikmesi olmadan gönderilen kalıcı olmayan iletilerden daha fazla kaybedilir.

Hedef çözünürlüğü

İleti bir kuyruğa gönderilirse, çözünürlük iki kez yönlendirilir; bir kez JMS uygulaması tarafından, bir kez de kuyruk yöneticisi iletiyi hazırlama kuyruğundan çıkardığında ve hedef kuyruğa gönderdiğinde.

JMS uygulaması gönderme yöntemini çağırdığında, yayınlara ilişkin hedef abonelikler eşleştirilir.

Bir ileti kuyruk tanımına göre kalıcılık ya da öncelikle gönderilirse, değer ikinci değil, ilk çözünürlükte ayarlanır.

Süre bitimi aralığı

Teslim gecikmesi, süre bitimi özelliğinin davranışını (**MQMD.Expiry**) korur. Örneğin, bir JMS uygulamasından 20.000 ms süre ve 5.000 ms teslimat gecikmesi ile bir ileti konduysa ve 10.000 ms geçen bir süreden sonra MQMD.expiry alanının değeri, saniyenin yaklaşık onda biri olabilir. Bu değer, iletinin konma zamanından iletinin gönderildiği zamana kadar geçen 15 saniyeyi gösterir.

Hazırlama kuyruğundayken bir iletinin süresi dolarsa ve MQRO_EXPIRATION_ * seçeneklerinden biri ayarlanırsa, oluşturulan rapor, uygulama tarafından gönderildiği şekilde özgün ileti içindir, teslim gecikmesi bilgilerini içermek için kullanılan üstbilgi kaldırılır.

Teslim gecikme işlemcisi durduruluyor ve başlatılıyor

z/OS z/OSişletim sistemi üzerinde, teslim gecikmesi işlemcisi kuyruk yöneticisi MSTR adres alanıyle bütünleştirilmiştir. Kuyruk yöneticisi başlatıldığında, teslim gecikme işlemcisi de başlar. Konaklatma kuyruğu kullanılabiliyorsa, kuyruğu açar ve iletilerin işlenmesini bekler. Hazırlama kuyruğu tanımlanmadıysa ya da alma işlemi için geçersiz kılındıysa ya da başka bir hata oluşursa, teslim gecikmesi işlemcisi kapanır. Konaklatma kuyruğu daha sonra tanımlanırsa ya da etkinleştirilmek üzere değiştirilirse, teslim gecikme işlemcisi yeniden başlatılır. Teslim gecikmesi işlemcisi başka bir nedenle kapanırsa, konaklatma kuyruğunun **PUT** özniteliği ETKINLEşTIRILMIş 'ten DEVRE Dışı ' ya değiştirilerek ve yeniden ETKINLEşTIRILMIş ' e geri dönilerek yeniden başlatılabilir. Herhangi bir nedenle teslim

gecikmesi işlemcisini durdurmanız gerekirse, hazırlama kuyruğunun PUT özniteliğini DISABLEDolarak ayarlayın.

Multi Çoklu platformlarişletim sistemi üzerinde, gecikme işlemcisi kuyruk yöneticisiyle başlar ve kurtarılabilir bir hata durumunda otomatik olarak yeniden başlatılır.

Hedef kuyruğa konamadı

Gecikmiş bir ileti, gecikmesi tamamlandıktan sonra hedef kuyruğa yerleştirilemezse, ileti rapor seçeneklerinde belirtildiği gibi ele alınmaktadır: atılır ya da teslim edilmeyen ileti kuyruğuna gönderilir. Bu işlem başarısız olursa, iletiyi daha sonra koymak için bir girişimde bulunun. İşlem başarılı olursa, rapor istenirse, bir kural dışı durum raporu oluşturulur ve belirtilen kuyruğa gönderilir. Rapor iletisi gönderilemezse, rapor iletisi gitmeyen ileti kuyruğuna gönderilir. Raporu teslim edilmeyen iletiler kuyruğuna gönderme başarısız olursa ve ileti kalıcı olursa, tüm değişiklikler atılır ve özgün ileti daha sonra geriye işlenir ve yeniden teslim edilir. İleti kalıcı değilse, rapor iletisi atılır, ancak diğer değişiklikler kesinleştirilir. Bir abone aboneliğini kaldırdığı için ya da kalıcı olmayan bir abone bağlantısı kesildiği için gecikmiş bir yayın teslim edilemezse, ileti sessiz bir şekilde atılır. Rapor iletileri, daha önce açıklandığı gibi yine de oluşturulur.

Gecikmiş bir yayın aboneye teslim edilemezse ve bunun yerine teslim edilmeyen iletiler kuyruğuna konursa ve teslim edilmeyen iletiler kuyruğuna konamazsa, ileti atılır.

Teslim gecikmesi tamamlandıktan sonra hedef kuyruğa koyma işleminin başarısız olma olasılığını azaltmak için kuyruk yöneticisi, JMS istemcisi sıfır olmayan bir teslim gecikmesiyle bir ileti gönderdiğinde bazı temel denetimleri gerçekleştirir. Bu denetimler, kuyruğun geçersiz kılınıp kılınmadığını, iletinin izin verilen ileti uzunluğu üst sınırından büyük olup olmadığını ve kuyruğun dolu olup olmadığını içerir.

Yayınla/abone ol

Bir yayının kullanılabilir aboneliklerle eşleştirilmesi, JMS uygulaması sıfır olmayan bir teslim gecikmesine sahip bir ileti gönderdiğinde oluşur. SYSTEM.DDELAY.LOCAL.QUEUE kuyruğu, teslim gecikmesi tamamlanıncaya kadar burada tutulur. Bu abonelerden biri başka bir kuyruk yöneticisi için yetkili sunucu aboneliğiyse, bu kuyruk yöneticisinde dağıtım gecikmesi tamamlandıktan sonra çıkış yayılır. Bu, diğer kuyruk yöneticisindeki abonelerin abone olmadan önce yayınlanan yayınları almasına neden olabilir. Bu, JMS 2.0 ya da sonraki belirtimden bir sapmadır.

Yayınlama/abone olma ile teslim gecikmesi yalnızca hedef konu (N) PMSGDLV = ALLAVAIL ile yapılandırıldıysa desteklenir. Diğer değerleri kullanma girişimi MQRC_PUBLICATION_FAILURE hatasıyla sonuçlanır. Teslim gecikmesi işlemcisi iletiyi hedef kuyruğa koyarken başarısız olursa, sonuç "Hedef kuyruğa konamadı" kısmında açıklanmıştır.

Rapor iletileri

Tüm rapor seçenekleri, yoksayılan, ancak hedef kuyruğa gönderildiğinde iletilen aşağıdaki seçenekler dışında teslim işlemcisi tarafından desteklenir ve üzerinde işlem gerçekleştirir:

- MQRO_COA*
- MQRO_COD*
- MQRO_PAN/MQRO_NAN
- MQRO_ACTIVITY

Klonlanan ve paylaşılan abonelikler

IBM MQ 8.0 ya da daha sonra, birden çok tüketicinin aynı aboneliğe erişmesine izin vermek için iki yöntem vardır. Bu iki yöntem, klonlanmış abonelikler kullanılarak ya da paylaşılan abonelikler kullanılarak sağlanır.

Eşkopyalanmış abonelikler

Eşkopyalanmış abonelik bir IBM MQ uzantısıdır. Klonlanmış abonelikler, farklı Java sanal makinelerinde (JVM) birden çok tüketicinin aboneliğe eşzamanlı olarak erişmesine olanak sağlar. Bu davranış, bir

ConnectionFactory nesnesinde **CLONESUPP** özelliği `Enabled` (Etkin) olarak ayarlanarak kullanılabilir. Varsayılan olarak **CLONESUPP** , `Disabled`(Devre Dışı) değeridir. Klonlanan abonelikler yalnızca sürekli aboneliklerde etkinleştirilebilir. **CLONESUPP** etkinleştirilirse, bu ConnectionFactory kullanılarak yapılan sonraki her bağlantı klonlanır.

Bu abonelikten ileti almak için bir ya da daha fazla tüketici yaratıldıysa, sürekli abonelik klonlanmış olarak kabul edilebilir; yani, bunlar aynı abonelik adını belirterek oluşturulmuştur. Bu yalnızca, tüketicilerin yaratıldığı bağlantının **CLONESUPP** MQConnectionFactoryüzerinde `Enabled` (Etkin) değerine ayarlı olması durumunda yapılabilir. Aboneliğin konusunda bir ileti yayınlandığında, o iletinin bir kopyası aboneliğe gönderilir. İleti herhangi bir tüketici tarafından kullanılabilir, ancak yalnızca bir kişi iletiyi alır.

Not: Klonlanmış aboneliklerin etkinleştirilmesi, JMS belirtimini genişletir.

Paylaşılan abonelikler

JMS 2.0 belirtimiyle tanıtılan paylaşılan abonelikler, bir konu aboneliğinden gelen iletilerin birden çok tüketici arasında paylaşılmasına izin verir. Abonelikten gelen her ileti, o abonelikteki tüketicilerden yalnızca birine teslim edilir. Paylaşılan abonelikler, JMS 2.0 ya da daha sonraki bir API ' ye yapılan ilgili çağrıyla etkinleştirilir.

API ' ler aşağıdaki yollardan biriyle çağrılabilir:

- Bir Java SE uygulamasından (ya da Java EE Client Container 'den).
- Bir sunucu uygulamacığından ya da MDB somutlamasından.

JMS 2.0 ya da sonraki belirtimler, bir MDB ' yi paylaşılan bir abonelikten çıkarmanın standart bir yolunu tanımlamaz; bu nedenle IBM MQ 8.0 ya da daha sonraki bir sürümü, bu amaçla **sharedSubscription** etkinleştirme belirtimi özelliğini sağlar. Bu özellik hakkında daha fazla bilgi için bkz. “Kaynak bağdaştırıcısının gelen iletişim için yapılandırılması” sayfa 435 ve “sharedSubscription özelliğinin nasıl tanımlanacağına ilişkin örnekler” sayfa 451.

Paylaşılan abonelik etkinleştirildiyse, paylaşılmayan bir abonelik olamaz.

Paylaşılan abonelikler, sürekli ya da kalıcı olmayan abonelikler olarak oluşturulabilir. Kuyruk yöneticisi tarafında olağan JMS yapılanışının ötesinde ayrı olarak nesne yaratma gereksinimi yoktur. Gereken nesneler dinamik olarak yaratılır.

Paylaşılan ya da klonlanan abonelikler arasında karar verilmesi

Paylaşılan ya da klonlanmış aboneliklerin kullanılıp kullanılmayacağına karar verirken, her ikisinin de avantajlarını göz önünde bulundurun. Olanaklıysa, paylaşılan abonelikleri IBM MQ uzantısından ziyade belirtim tanımlı bir davranış olarak kullanın.

Aşağıdaki tablo, paylaşılan ve klonlanan abonelikler arasında karar verirken göz önünde bulundurmanız gereken bazı noktaları içerir:

Çizelge 48. Paylaşılan abonelikler ve klonlanan abonelikler için dikkat edilecek noktaların karşılaştırılması

Paylaşılan Abonelikler	**Eşkopyalanmış Abonelikler**
Paylaşılan abonelikler, JMS 2.0 ya da sonraki belirtimin standart bir parçasıdır.	Eşkopyalanmış abonelikler IBM MQ ' a özgü bir uzantıdır.
Paylaşılan abonelikler, belirtik API yöntemi çağrıları kullanılarak oluşturulur.	Eşkopyalanmış abonelikler yönetimsel olarak ConnectionFactory düzeyinde denetlenir.
Paylaşılan abonelikler sürekli ya da sürekli olabilir.	Eşkopyalanmış abonelikler yalnızca kalıcı olabilir.
Paylaşılan abonelikler, tek tek abonelik temelinde belirtik olarak oluşturulur.	Eşkopyalanmış abonelikler, işlevin etkinleştirildiği bir bağlantı altında sürekli abonelik için kullanılır.

Çizelge 48. Paylaşılan abonelikler ve klonlanan abonelikler için dikkat edilecek noktaların karşılaştırılması (devamı var)	
Paylaşılan Abonelikler	**Eşkopyalanmış Abonelikler**
Bir abonelik paylaşılan olarak oluşturulursa, daha sonra paylaşılmayan olarak ya da paylaşılmayan olarak değiştirilemez.	Sahip olan bağlantının **CLONESUPP** özelliği değiştiyse, abonelik her yeniden açıldığında klonlanandan kopyalanmamış olarak değiştirilebilir.

İlgili kavramlar

Aboneler ve abonelikler

Abonelik dayanıklılığı

İlgili görevler

JMS 2.0 paylaşılan aboneliklerinin kullanılması

İlgili başvurular

“sharedSubscription özelliğinin nasıl tanımlanacağına ilişkin örnekler” sayfa 451

Bir WebSphere Liberty `server.xml` dosyasında etkinleştirme belirtiminin sharedSubscription özelliğini tanımlayabilirsiniz. Diğer bir seçenek olarak, ek açıklamaları kullanarak özelliği iletiyle yönlendirilen bir bean (MDB) içinde tanımlayabilirsiniz.

KLONESUPP

SupportMQExtensions özelliği

JMS 2.0 belirtimi, belirli davranışların çalışma şeklini değiştirdi. IBM MQ 8.0 ve daha sonra, değiştirilen bu davranışları önceki uygulamalara geri döndürmek için TRUE olarak ayarlanabilen **com.ibm.mq.jms.SupportMQExtensions**özelliğini içerir.

V 9.3.0 V 9.3.0 JM 3.0 **com.ibm.mq.jakarta.jms.SupportMQExtensions** özelliği (Jakarta Messaging 3.0), `com.ibm.mq.jakarta.client.jar`içinde bulunan IBM MQ classes for Jakarta Messagingtarafından desteklenir.

JMS 2.0 **com.ibm.mq.jms.SupportMQExtensions** (JMS 2.0) özelliği, `com.ibm.mq.allclient.jar` ya da `com.ibm.mqjms.jar`içinde bulunan IBM MQ classes for JMStarafından desteklenir.

SupportMQExtensions ayarı `True`olarak ayarlanarak üç işlev alanı geri çevrilir:

İleti önceliği
: İletilere 0-9 öncelik atanabilir. JMS 2.0' den önce iletiler, bir kuyruğun varsayılan önceliğinin kullanıldığını gösteren -1 değerini de kullanabilir. JMS 2.0 ve sonrası, -1 ileti önceliğinin ayarlanmasına izin vermez. **SupportMQExtensions** ' un açılması -1 değerinin kullanılmasına izin verir.

İstemci Tanıtıcısı
: JMS 2.0 ya da üstü belirtimi, boş olmayan istemci tanıtıcılarının bağlantı kurarken benzersiz olup olmadıklarının denetlenmesini gerektirir. **SupportMQExtensions** ' un açılması, bu gereksinimin göz ardı edileceği ve bir istemci tanıtıcısının yeniden kullanılabileceği anlamına gelir.

NoLocal
: JMS 2.0 ya da daha sonraki bir belirtim, bu değişmez açıldığında, bir tüketicinin aynı istemci tanıtıcısı tarafından yayınlanan iletileri alamamasını gerektirir. JMS 2.0öncesinde bu öznitelik, kendi bağlantısıyla yayınlanan iletileri almasını önlemek için bir abonede ayarlanmıştı. **SupportMQExtensions** ' un açılması bu davranışı önceki uygulamasına geri çevirir.

Bu özellik aşağıdaki gibi ayarlanabilir:

```
java -Dcom.ibm.mq.jms.SupportMQExtensions=true
```

Bu özellik, **java** komutunda standart bir JVM Sistemi özelliği olarak ayarlanabilir ya da IBM MQ classes for JMS yapılandırma dosyasında bulunabilir.

İlgili kavramlar

“IBM MQ classes for JMS/Jakarta Messaging yapılandırma dosyası” sayfa 94
IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging yapılandırma dosyaları, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging' yi yapılandırmak için kullanılan özellikleri belirtir.

İlgili başvurular

“JMS istemci davranışını yapılandırmak için kullanılan özellikler” sayfa 100
JMS istemcisinin davranışını yapılandırmak için bu özellikleri kullanın.

JMS uygulamalarında paylaşılan abonelikleri kullanma

Paylaşılan aboneliklerle, tek bir abonelik birden çok tüketici arasında paylaşılır ve tüketicilerden yalnızca biri herhangi bir zamanda yayın alır.

Paylaşılan abonelikler JMS 2.0 ' den itibaren kullanılabilir. IBM MQ 8.0 ya da daha sonraki bir yayın düzeyiyle ilgili bir JMS uygulaması geliştirirken, bu işlevin kuyruk yöneticiniz üzerindeki etkisini göz önünde bulundurmanız gerekebilir.

Paylaşılan aboneliklerin arkasındaki fikir, yükü birden çok tüketici arasında paylaşmaktır. Sürekli abonelik birden çok tüketici arasında da paylaşılabilir.

Örneğin, bir:

- SUBaboneliği, üç tüketici tarafından paylaşılan futbol maçı güncellemelerini almak için FIFA2014/UPDATES konusuna abone olma C1, C2ve C3
- FIFA2014/UPDATES konusunda üretici P1 yayınlama

FIFA2014/UPDATESüzerinde bir yayın yapıldığında, yayın üç tüketicinin (C1, C2ya da C3) yalnızca biri tarafından alınır, ancak tümü alınmaz.

Aşağıdaki örnek, paylaşılan aboneliklerin kullanımını gösterir ve yalnızca ileti gövdesini almak için JMS 2.0, `Message.receiveBody()` içinde ek API kullanımını gösterir.

Örnek, FIFA2014/UPDATES konusuna paylaşılan abonelik oluşturan üç abone iş parçacığı ve bir yayınlayıcı iş parçacığı yaratır.

V 9.3.0 V 9.3.0 JM 3.0

```
package mqv91Samples;

import jakarta.jms.JMSException;

import com.ibm.msg.client.jms.JmsConnectionFactory;
import com.ibm.msg.client.jms.JmsFactoryFactory;
import com.ibm.msg.client.wmq.WMQConstants;

import jakarta.jms.JMSContext;
import jakarta.jms.Topic;
import jakarta.jms.Queue;
import jakarta.jms.JMSConsumer;
import jakarta.jms.Message;
import jakarta.jms.JMSProducer;

/*
* Implements both Subscriber and Publisher
*/
class SharedNonDurableSubscriberAndPublisher implements Runnable {
   private Thread t;
   private String threadName;

   SharedNonDurableSubscriberAndPublisher( String name){
       threadName = name;
       System.out.println("Creating Thread:" +  threadName );
   }

   /*
    * Demonstrates shared non-durable subscription in JMS 2.0 and later
    */
    private void sharedNonDurableSubscriptionDemo(){
        JmsConnectionFactory cf = null;
        JMSContext msgContext = null;
```

```
        try {
            // Create Factory for WMQ JMS provider
            JmsFactoryFactory ff = JmsFactoryFactory.getInstance(WMQConstants.WMQ_PROVIDER);
            // Create connection factory
            cf = ff.createConnectionFactory();
            // Set MQ properties
            cf.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM3");
            cf.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE, WMQConstants.WMQ_CM_BINDINGS);
            // Create message context
            msgContext = cf.createContext();

            // Create a topic destination
            Topic fifaScores = msgContext.createTopic("/FIFA2014/UPDATES");

            // Create a consumer. Subscription name specified, required for sharing of subscription.
            JMSConsumer msgCons = msgContext.createSharedConsumer(fifaScores, "FIFA2014SUBID");

            // Loop around to receive publications
            while(true){

                String msgBody=null;

                // Use JMS 2.0 and later receiveBody method as we are interested in message body only.
                msgBody = msgCons.receiveBody(String.class);

                if(msgBody != null){
                    System.out.println(threadName + " : " + msgBody);
                }
            }
        }catch(JMSException jmsEx){
            System.out.println(jmsEx);
        }
    }
```

JMS 2.0

```
package mqv91Samples;

import javax.jms.JMSException;

import com.ibm.msg.client.jms.JmsConnectionFactory;
import com.ibm.msg.client.jms.JmsFactoryFactory;
import com.ibm.msg.client.wmq.WMQConstants;

import javax.jms.JMSContext;
import javax.jms.Topic;
import javax.jms.Queue;
import javax.jms.JMSConsumer;
import javax.jms.Message;
import javax.jms.JMSProducer;

/*
* Implements both Subscriber and Publisher
*/
class SharedNonDurableSubscriberAndPublisher implements Runnable {
   private Thread t;
   private String threadName;

   SharedNonDurableSubscriberAndPublisher( String name){
       threadName = name;
       System.out.println("Creating Thread:" +  threadName );
   }

   /*
    * Demonstrates shared non-durable subscription in JMS 2.0 and later
    */
    private void sharedNonDurableSubscriptionDemo(){
        JmsConnectionFactory cf = null;
        JMSContext msgContext = null;

        try {
            // Create Factory for WMQ JMS provider
            JmsFactoryFactory ff = JmsFactoryFactory.getInstance(WMQConstants.WMQ_PROVIDER);
            // Create connection factory
            cf = ff.createConnectionFactory();
            // Set MQ properties
            cf.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM3");
            cf.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE, WMQConstants.WMQ_CM_BINDINGS);
            // Create message context
            msgContext = cf.createContext();
```

```
            // Create a topic destination
            Topic fifaScores = msgContext.createTopic("/FIFA2014/UPDATES");

            // Create a consumer. Subscription name specified, required for sharing of subscription.
            JMSConsumer msgCons = msgContext.createSharedConsumer(fifaScores, "FIFA2014SUBID");

            // Loop around to receive publications
            while(true){

                String msgBody=null;

                // Use JMS 2.0 and later receiveBody method as we are interested in message body only.
                msgBody = msgCons.receiveBody(String.class);

                if(msgBody != null){
                    System.out.println(threadName + " : " + msgBody);
                }
            }
        }catch(JMSException jmsEx){
            System.out.println(jmsEx);
        }
    }
```

```
 /*
        * Publisher publishes match updates like current attendance in the stadium, goal score and ball
possession by teams.
        */
        private void matchUpdatePublisher(){
            JmsConnectionFactory cf = null;
            JMSContext msgContext = null;
            int nederlandsGoals = 0;
            int chileGoals = 0;
            int stadiumAttendence = 23231;
            int switchIndex = 0;
            String msgBody = "";
            int nederlandsHolding = 60;
            int chileHolding = 40;

            try {
                // Create Factory for WMQ JMS provider
                JmsFactoryFactory ff = JmsFactoryFactory.getInstance(WMQConstants.WMQ_PROVIDER);

                // Create connection factory
                cf = ff.createConnectionFactory();
                // Set MQ properties
                cf.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER, "QM3");
                cf.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE, WMQConstants.WMQ_CM_BINDINGS);

                // Create message context
                msgContext = cf.createContext();

                // Create a topic destination
                Topic fifaScores = msgContext.createTopic("/FIFA2014/UPDATES");

                // Create publisher to publish updates from stadium
                JMSProducer msgProducer = msgContext.createProducer();

                while(true){
                    // Send match updates
                    switch(switchIndex){
                        // Attendance
                    case 0:
                        msgBody ="Stadium Attendence " + stadiumAttendence;
                        stadiumAttendence += 314;
                        break;

                        // Goals
                    case 1:
                        msgBody ="SCORE: The Netherlands: " + nederlandsGoals + " - Chile:" + chileGoals;
                        break;

                        // Ball possession percentage
                    case 2:
                        msgBody ="Ball possession: The Netherlands: " + nederlandsHolding + "% - Chile:
" + chileHolding + "%";
                        if((nederlandsHolding > 60) && (nederlandsHolding < 70)){
                            nederlandsHolding -= 2;
                            chileHolding += 2;
                        }else{
                            nederlandsHolding += 2;
                            chileHolding -= 2;
```

```
                        }
                        break;
                    }

                    // Publish and wait for two seconds to publish next update
                    msgProducer.send (fifaScores, msgBody);
                    try{
                        Thread.sleep(2000);
                    }catch(InterruptedException iex){

                    }

                    // Increment and reset the index if greater than 2
                    switchIndex++;
                    if(switchIndex > 2)
                        switchIndex = 0;
                }
            }catch(JMSException jmsEx){
                System.out.println(jmsEx);
            }
        }

    /*
     * (non-Javadoc)
     * @see java.lang.Runnable#run()
     */
    public void run() {
        // If this is a publisher thread
        if(threadName == "PUBLISHER"){
            matchUpdatePublisher();
        }else{
            // Create subscription and start receiving publications
            sharedNonDurableSubscriptionDemo();
        }
    }

       // Start thread
   public void start (){
      System.out.println("Starting " +  threadName );
      if (t == null)
      {
         t = new Thread (this, threadName);
         t.start ();
      }
   }
}
```

```
/*
* Demonstrate JMS 2.0 and later simplified API using IBM MQ 91 JMS Implementation
*/
public class Mqv91jms2Sample {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        // Create first subscriber and start
        SharedNonDurableSubscriberAndPublisher subOne = new
SharedNonDurableSubscriberAndPublisher( "SUB1");
        subOne.start();

        // Create second subscriber and start
        SharedNonDurableSubscriberAndPublisher subTwo = new
SharedNonDurableSubscriberAndPublisher( "SUB2");
        subTwo.start();

        // Create third subscriber and start
        SharedNonDurableSubscriberAndPublisher subThree = new
SharedNonDurableSubscriberAndPublisher( "SUB3");
        subThree.start();

        // Create publisher and start
        SharedNonDurableSubscriberAndPublisher publisher = new
SharedNonDurableSubscriberAndPublisher( "PUBLISHER");
        publisher.start();
    }
}
```

İlgili kavramlar

IBM MQ Java dil arabirimleri

V 9.3.2 Modüler uygulamanızın IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging kullanacak şekilde yapılandırılması

V 9.3.2 Uygulamanızda uygun modülü ve modül yoluna uygun dizini ekleyerek IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging modüllerini modüler bir şekilde kullanabilirsiniz.

Modüler paketleme

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging için birleştirilmiş JAR dosyaları, JAR dosyası adlarından türetilen varsayılan adların yerini alan otomatik modül adları sağlar.

- IBM MQ classes for JMS (`com.ibm.mq.allclient.jar`), `com.ibm.mq.javax`modül adıyla sağlanır.
- IBM MQ classes for Jakarta Messaging (`com.ibm.mq.jakarta.client.jar`), `com.ibm.mq.jakarta`modül adıyla sağlanır.

Modüller aynı paketi içeremediğinden ve varsayılan dizin birden çok JAR içinde aynı paketleri içerdiğinden, varsayılan `MQ_HOME/java/lib` dizini modüler kullanım için uygun değildir. Bu nedenle, JAR dosyaları arasında paket yinelemesi olmadan, yalnızca gerekli JAR dosyalarını içeren yeni dizinler kullanılabilir. Bu dizinler, `module-path`üzerinde içerilmeye uygundur.

Not: Varsayılan modül adlarına dayanarak modüler bir bağlamda kullanılabilir JAR dosyalarını kullanan uygulamalarınız varsa, uygulamalarınızı yeni modül adlarını gerektirecek şekilde güncellemelisiniz. Varsayılan birim adları JAR dosyası adlarından türetilir.

Modüler uygulamanızın IBM MQ classes for JMS kullanacak şekilde yapılandırılması

Aşağıdaki adımları tamamlayarak modüler uygulamanızı IBM MQ classes for JMS (`com.ibm.mq.allclient.jar`) kullanacak şekilde yapılandırabilirsiniz:

- Uygulamayı, `com.ibm.mq.javax` modülünü gerektirecek şekilde yapılandırın.
- Uygulamayı, modül yolunda `MQ_HOME/java/lib/modules/javax` dizinini içerecek şekilde yapılandırın.

Modüler uygulamanızın IBM MQ classes for Jakarta Messaging kullanacak şekilde yapılandırılması

Aşağıdaki adımları tamamlayarak modüler uygulamanızı IBM MQ classes for Jakarta Messaging (`com.ibm.mq.jakarta.client.jar`) kullanacak şekilde yapılandırabilirsiniz:

- Uygulamayı, `com.ibm.mq.jakarta` modülünü gerektirecek şekilde yapılandırın.
- Uygulamayı, modül yolunda `MQ_HOME/java/lib/modules/jakarta` dizinini içerecek şekilde yapılandırın.

Modüler uygulamanızın IBM MQ classes for Java kullanacak şekilde yapılandırılması

IBM MQ classes for Java ' i modüler bir uygulamadan kullanmak için IBM MQ classes for JMS yapılandırmasını ya da her iki istemci JAR dosyası desteği olarak IBM MQ classes for Jakarta Messagingyapılandırmasını kullanabilirsiniz IBM MQ classes for Java. Ancak, uygulamanızın her ikisini birden değil, yalnızca bu yapılandırmalardan birini kullanması gerekir.

IBM MQ classes for JMS Uygulama Sunucusu Olanakları

Bu konuda, IBM MQ classes for JMS ' in Oturum sınıfında ConnectionConsumer sınıfını ve gelişmiş işlevselliği nasıl uyguladığı açıklanmaktadır. Bir sunucu oturumu havuzunun işlevini de özetler.

Önemli: Bu bilgiler yalnızca başvuru içindir. Bu arabirimi kullanmak için bir uygulama yazılmamalıdır: Java EE sunucularına bağlanmak için IBM MQ kaynak bağdaştırıcısı içinde kullanılır. Pratik bağlantı bilgileri için bkz. “IBM MQ kaynak bağdaştırıcısının kullanılması” sayfa 419.

IBM MQ classes for JMS , *Java Message Service Belirtimi* ' nde belirtilen Uygulama Sunucusu Tesislerini (ASF) destekler (bkz. Oracle Technology Network for Java Developers). Bu belirtim, bu programlama modeli içindeki üç rolü tanımlar:

- **JMS Sağlayıcı** , ConnectionConsumer ve gelişmiş Oturum işlevlerini sağlar.
- **Uygulama sunucusu** , ServerSessionPool ve ServerSession işlevlerini sağlar.
- **İstemci uygulaması** , JMS sağlayıcısının ve uygulama sunucusunun sağladığı işlevselliği kullanır.

Bir uygulama bir aracıya gerçek zamanlı bağlantı kullanıyorsa, bu konudaki bilgiler geçerli değildir.

JMS ConnectionConsumer

ConnectionConsumer arabirimi, iletileri bir iş parçacığı havuzuna eşzamanlı olarak teslim etmek için yüksek performanslı bir yöntem sağlar.

JMS belirtimi, `ConnectionConsumer` arabirimini kullanarak bir uygulama sunucusunun bir JMS uygulamasıyla yakından bütünleşmesini sağlar. Bu özellik, iletilerin eşzamanlı olarak işlenmesini sağlar. Genellikle, bir uygulama sunucusu bir iş parçacığı havuzu yaratır ve JMS uygulaması iletileri bu iş parçacıklarının kullanımına açar. JMSkullanan bir uygulama sunucusu (WebSphere Application Servergibi), ileti odaklı Bean 'ler gibi üst düzey ileti sistemi işlevselliği sağlamak için bu özelliği kullanabilir.

Normal uygulamalar ConnectionConsumeruygulamasını kullanmaz, ancak uzman JMS istemcileri bunu kullanabilir. Bu tür istemciler için ConnectionConsumer , iletileri bir iş parçacığı havuzuna eşzamanlı olarak teslim etmek için yüksek performanslı bir yöntem sağlar. Bir ileti bir kuyruğa ya da konuya ulaştığında, JMS havuzdan bir iş parçacığı seçer ve ona bir ileti kümesi gönderir. Bunu yapmak için JMS , ilişkili bir MessageListener' in `onMessage()` yöntemini çalıştırır.

Her biri kayıtlı bir MessageListenerolan birden çok Oturum ve MessageConsumer nesnesi oluşturarak aynı etkiyi elde edebilirsiniz. Ancak ConnectionConsumer , daha iyi performans, daha az kaynak kullanımı ve daha fazla esneklik sağlar. Özellikle, daha az Oturum nesnesi gerekir.

ASF ile uygulama planlanması

Bu bölümde, aşağıdakiler de dahil olmak üzere bir uygulamanın nasıl planlayacağınız açıklanmaktadır:

- “ASF kullanarak noktadan noktaya ileti sistemine ilişkin genel ilkeler” sayfa 321
- “ASF kullanarak yayınlama/abone olma ileti sistemine ilişkin genel ilkeler” sayfa 322
- “ASF ' deki kuyruktan ileti kaldırılması” sayfa 323
- ASF ' de zehirli mesajlarla uğraşıyorum. Bkz. “IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 225.

ASF kullanarak noktadan noktaya ileti sistemine ilişkin genel ilkeler

ASF kullanarak noktadan noktaya ileti sistemine ilişkin genel bilgiler için bu konuyu kullanın.

Bir uygulama QueueConnection nesnesinden ConnectionConsumer oluşturduğunda, bir JMS kuyruk nesnesi ve seçici dizgisini belirtir. ConnectionConsumer , ilişkili ServerSessionhavuzundaki oturumlara ileti sağlamaya başlar. İletiler kuyruğa gelir ve seçiciyle eşleşirse, ilişkili ServerSessionHavuzundaki oturumlara teslim edilir.

IBM MQ terimlerinde, kuyruk nesnesi yerel kuyruk yöneticisinde bir QLOCAL ya da QALIAS öğesine gönderme yapar. Bu bir QALIAS ise, bu QALIAS 'in bir QLOCAL' e başvurması gerekir. Tam olarak çözümlenen IBM MQ QLOCAL, *temeldeki QLOCAL*olarak bilinir. Kapatılmamışsa ve üst QueueConnection başlatıldıysa, ConnectionConsumer öğesinin *etkin* olduğu belirtilir.

Her biri farklı seçicilere sahip birden çok ConnectionConsumers'ın aynı temel QLOCAL' a karşı çalışması mümkündür. Başarımı korumak için istenmeyen iletiler kuyrukta toplanmamalıdır. İstenmeyen iletiler, etkin ConnectionConsumer kullanıcısı olmayan, eşleşen bir seçiciye sahip olan iletilerdir. QueueConnectionFactory olanağını, bu istenmeyen iletilerin kuyruktan kaldırılacağı şekilde ayarlayabilirsiniz (ayrıntılar için bkz. “ASF ' deki kuyruktan ileti kaldırılması” sayfa 323). Bu davranışı şu iki yoldan biriyle ayarlayabilirsiniz:

- JMS denetim aracını kullanarak QueueConnectionFactory öğesini MRET (NO) olarak ayarlayın.

- Programınızda aşağıdakileri kullanın:

```
MQQueueConnectionFactory.setMessageRetention(WMQConstants.WMQ_MRET_NO)
```

Bu ayarı değiştirmezseniz, varsayılan değer, bu tür istenmeyen iletilerin kuyrukta tutulması olur.

IBM MQ kuyruk yöneticisini ayarlarken aşağıdaki noktaları göz önünde bulundurun:

- Paylaşılan giriş için temel QLOCAL etkinleştirilmelidir. Bunu yapmak için aşağıdaki MQSC komutunu kullanın:

```
ALTER QLOCAL( your.qlocal.name ) SHARE GET(ENABLED)
```

- Kuyruk yöneticinizin etkinleştirilmiş bir ileti kuyruğuna sahip olması gerekir. Bir ConnectionConsumer , bir iletiyi gönderilmeyen ileti kuyruğuna koyarken bir sorunla karşılaşırsa, temeldeki QLOCAL ' dan ileti teslimi durur. Bir gönderilmeyen ileti kuyruğu tanımlamak için aşağıdakileri kullanın:

```
ALTER QMGR DEADQ( your.dead.letter.queue.name )
```

- ConnectionConsumer ürününü çalıştıran kullanıcının MQOO_SAVE_ALL_CONTEXT ve MQOO_PASS_ALL_CONTEXT ile MQOPEN gerçekleştirme yetkisi olmalıdır. Ayrıntılar için, altyapınıza ilişkin IBM MQ belgelerine bakın.
- İstenmeyen iletiler kuyrukta bırakılırsa, bu iletiler sistem başarımını düşürüyor. Bu nedenle, ileti seçicilerinizi, bunlar arasında ConnectionConsumers (Bağlantı Sağlayıcıları) kuyruktaki tüm iletileri kaldıracak şekilde planlayın.

MQSC komutlarıyla ilgili ayrıntılar için MQSC komutlarıkonusuna bakın.

ASF kullanarak yayınlama/abone olma ileti sistemine ilişkin genel ilkeler
ConnectionConsumers , belirtilen bir Konu için ileti alır. Bir ConnectionConsumer dayanıklı ya da dayanıklı olamaz. ConnectionConsumer tarafından kullanılan kuyruğu ya da kuyrukları belirtmeniz gerekir.

Bir uygulama bir TopicConnection nesnesinden ConnectionConsumer oluşturduğunda, bir Konu nesnesini ve seçici dizgisini belirtir. ConnectionConsumer , daha sonra, abone olunan konu için alıkonan yayınlar da içinde olmak üzere, o Konu üzerindeki seçiciyle eşleşen iletileri almaya başlar.

Diğer bir seçenek olarak, bir uygulama belirli bir adla ilişkilendirilmiş kalıcı bir ConnectionConsumer yaratabilir. Bu ConnectionConsumer , kalıcı ConnectionConsumer son etkinliğinden bu yana Konu üzerinde yayınlanan iletileri alır. Konu üzerindeki seçiciyle eşleşen tüm bu iletileri alır. Ancak, ConnectionConsumer önden okuma özelliğini kullanıyorsa, kapandığında istemci arabelleğindeki kalıcı olmayan iletileri kaybedebilir.

IBM MQ classes for JMS IBM MQ ileti alışverişi sağlayıcısı geçiş kipindeyse, kalıcı olmayan ConnectionConsumer abonelikleri için ayrı bir kuyruk kullanılır.TopicConnectionÜreticisi 'ndeki CCSUB yapılandırılır seçeneği, kullanılacak kuyruğu belirtir. Normalde, CCSUB, aynı TopicConnectionÜreticisi 'ni kullanan tüm ConnectionConsumers tarafından kullanılmak üzere tek bir kuyruk belirtir. Ancak, her ConnectionConsumer öğesinin bir kuyruk adı öneki ve ardından bir yıldız işareti (*) belirterek geçici bir kuyruk oluşturmasını sağlayabilirsiniz.

IBM MQ classes for JMS , IBM MQ ileti sistemi sağlayıcısı geçiş kipindeyse, Konunun CCDSUB özelliği, sürekli abonelikler için kullanılacak kuyruğu belirtir. Bu, önceden var olan bir kuyruk ya da bir kuyruk adı önekini izleyen bir yıldız işareti (*) olabilir. Önceden var olan bir kuyruk belirtirseniz, Konu 'ya abone olan tüm sürekli ConnectionConsumers bu kuyruğu kullanır. Bir kuyruk adı önekinin ardından bir yıldız işareti (*) belirtirseniz, belirli bir adla kalıcı bir ConnectionConsumer ilk kez oluşturulduğunda bir kuyruk oluşturulur. Bu kuyruk, daha sonra aynı adla kalıcı bir ConnectionConsumer yaratıldığında yeniden kullanılır.

IBM MQ kuyruk yöneticisini ayarlarken aşağıdaki noktaları göz önünde bulundurun:

- Kuyruk yöneticinizin etkinleştirilmiş bir ileti kuyruğuna sahip olması gerekir. Bir ConnectionConsumer , bir iletiyi gönderilmeyen ileti kuyruğuna koyarken bir sorunla karşılaşırsa, temeldeki QLOCAL ' dan ileti teslimi durur. Bir gönderilmeyen ileti kuyruğu tanımlamak için aşağıdakileri kullanın:

```
ALTER QMGR DEADQ( your.dead.letter.queue.name )
```

- ConnectionConsumer ürününü çalıştıran kullanıcının MQOO_SAVE_ALL_CONTEXT ve MQOO_PASS_ALL_CONTEXT ile MQOPEN gerçekleştirme yetkisi olmalıdır. Ayrıntılar için platformunuza ilişkin IBM MQ belgelerine bakın.
- Ayrı, özel olarak ayrılmış bir kuyruk oluşturarak tek bir ConnectionConsumer için performansı eniyileyebilirsiniz. Bu, ek kaynak kullanımı maliyetlidir.

ASF ' deki kuyruktan ileti kaldırılması
Bir uygulama ConnectionConsumerskomutunu kullandığında, JMS ' in bazı durumlarda kuyruktan iletileri kaldırması gerekebilir.

Bu durumlar aşağıdaki gibidir:

Hatalı biçimlendirilmiş ileti
JMS ' in ayrıştıramadığı bir ileti gelebilir.

Zehirli mesaj
Bir ileti geriletme eşiğine ulaşabilir, ancak ConnectionConsumer onu geriletme kuyruğunda yeniden kuyruğa alamaz.

İlgili ConnectionConsumer yok
Noktadan noktaya iletişim ileti sistemi için, QueueConnectionFactory, istenmeyen iletileri alıkoymayacak şekilde ayarlandığında, ConnectionConsumerstarafından istenmeyen bir ileti gelir.

Bu durumlarda, ConnectionConsumer iletiyi kuyruktan kaldırmayı dener. İletinin MQMD ' nin rapor alanındaki yok etme seçenekleri tam davranışı ayarladı. Bu seçenekler şunlardır:

MQRO_DEAD_LETTER_Q
İleti, kuyruk yöneticisinin teslim edilmeyen ileti kuyruğuna yeniden yerleştirilir. Bu varsayılandır.

MQRO_DISCARD_MSG
İleti atıldı.

ConnectionConsumer bir rapor iletisi oluşturur ve bu, iletinin MQMD ' nin rapor alanına da bağlıdır. Bu ileti, ReplyToQmgr 'de iletinin ReplyToQ ' ya gönderilir. Rapor iletisi gönderilirken bir hata oluşursa, ileti gitmeyen ileti kuyruğuna gönderilir. İletinin MQMD ' nin rapor alanındaki kural dışı durum raporu seçenekleri, rapor iletisinin ayrıntılarını ayarlar. Bu seçenekler şunlardır:

MQRO_EXCEPTION
Özgün iletiye ilişkin MQMD ' yi içeren bir rapor iletisi oluşturulur. Herhangi bir ileti gövdesi verisi içermiyor.

MQRO_EXCEPTION_WITH_DATA
MQMD, herhangi bir MQ üstbilgisini ve 100 baytlık gövde verilerini içeren bir rapor iletisi oluşturulur.

MQRO_EXCEPTION_WITH_FULL_DATA
Özgün iletideki tüm verileri içeren bir rapor iletisi oluşturulur.

varsayılan
Rapor iletisi oluşturulmadı.

Rapor iletileri oluşturulduğunda, aşağıdaki seçenekler yerine getirildiğinde:

- MQRO_NEW_MSG_ID
- MQRO_PASS_MSG_ID
- MQRO_COPY_MSG_ID_TO_CORREL_ID
- MQRO_PASS_CORREL_ID

Bir zehir iletisi yeniden kuyruğa alınamazsa, belki de ölü mektup kuyruğu dolu olduğu için ya da yetki yanlış belirtildiğinden, ne olduğu iletinin devamlılığına bağlıdır. İleti kalıcı değilse, ileti atılır ve rapor iletisi

oluşturulmaz. İleti kalıcıysa, bu hedefte dinleyen tüm bağlantı tüketicilerine ileti teslimi durur. Bu tür bağlantı tüketicilerinin yeniden oluşturulabilmesi ve ileti teslimi yeniden başlatılmadan önce kapatılması ve sorun çözülmesi gerekir.

Bir teslim etmeyen ileti kuyruğunun tanımlanması ve herhangi bir sorun oluşmadığından emin olmak için kuyruğun düzenli olarak denetlenmesi önemlidir. Özellikle, gitmeyen iletiler kuyruğunun derinlik üst sınırına ulaşmadığından ve ileti büyüklüğü üst sınırının tüm iletiler için yeterli olduğundan emin olun.

Bir ileti, teslim edilmeyen ileti kuyruğuna yeniden gönderildiğinde, iletiden önce bir IBM MQ teslim edilmeyen mektup üstbilgisi (MQDLH) gelir. MQDLH biçimiyle ilgili ayrıntılar için MQDLH-Dead-letter header konusuna bakın. Bir ConnectionConsumer tarafından teslim edilmeyen ileti kuyruğuna yerleştirilen iletileri ya da ConnectionConsumer tarafından oluşturulan rapor iletilerini aşağıdaki alanlara göre tanımlayabilirsiniz:

- PutApplTip: `MQAT_JAVA (0x1C)`
- PutApplAdı "`MQ JMS ConnectionConsumer`"

Bu alanlar, ileti kuyruğundaki iletilere ilişkin MQDLH 'de ve rapor iletilerine ilişkin MQMD' de bulunur. MQMD 'nin geribildirim alanı ve MQDLH' nin Neden alanı, hatayı açıklayan bir kod içeriyor. Bu kodlarla ilgili ayrıntılar için bkz. “ASF ' de neden ve geribildirim kodları” sayfa 325. Diğer alanlar MQDLH-Dead-letter header' da açıklanmıştır.

ASF ' de zehirli iletilerin işlenmesi
Uygulama Sunucusu Tesislerinde, zehirli ileti işleme IBM MQ classes for JMS' in başka yerlerine göre biraz daha farklı şekilde ele alınır.

IBM MQ classes for JMSiçinde zehirli iletilerin işlenmesine ilişkin bilgi için bkz. “IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 225.

Application Server Facilities (ASF) kullandığınızda, MessageConsumeryerine ConnectionConsumerzehirli iletileri işler. ConnectionConsumer , kuyruğun BackoutThreshold ve BackoutRequeueQName özelliklerine göre iletileri yeniden kuyruğa yollar.

Bir uygulama ConnectionConsumerskomutunu kullandığında, iletinin geriletilmesi, uygulama sunucusunun sağladığı oturuma bağlıdır:

- Oturum işlem yapılmadığında, AUTO_ONAY ya da DUPS_OK_ONAY ile, bir ileti yalnızca bir sistem hatasından sonra ya da uygulama beklenmedik bir şekilde sona ererse geriletir.
- Oturum CLIENT_ONAY ile işlem yapılmadığında, alınmamış iletiler Session.recover() aracını çağıran uygulama sunucusu tarafından geri çekilebilir.

  Genellikle, MessageListener istemci uygulaması ya da uygulama sunucusu Message.acknowledge() öğesini çağırır. Message.acknowledge(), şu ana kadar oturumda teslim edilen tüm iletileri onaylar.
- Oturum işlemden geçtiğinde, alınmamış iletiler Session.rollback() işlevini çağıran uygulama sunucusu tarafından yedeklenebilir.
- Uygulama sunucusu bir XASession sağladıysa, iletiler dağıtılmış bir harekete bağlı olarak kesinleştirilir ya da geri çekilir. Uygulama sunucusu, işlemin tamamlanmasına ilişkin sorumluluğu üstlenir.

İlgili kavramlar
“IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 225
Zehirli bir mesaj, alan bir uygulama tarafından işlenemeyen bir mesajdır. Bir zehirli ileti bir uygulamaya teslim edilir ve belirli bir sayıda geriye işlenirse, IBM MQ classes for JMS bunu bir geriletme kuyruğuna taşıyabilir.

Hata işleme

Bu bölümde, “ASF ' deki hata koşullarından kurtarma” sayfa 324 ve “ASF ' de neden ve geribildirim kodları” sayfa 325dahil olmak üzere hata işlemenin çeşitli yönleri ele alınmıştır.

ASF ' deki hata koşullarından kurtarma
Bir ConnectionConsumer ciddi bir hatayla karşılaşırsa, aynı QLOCAL ile ilgilenilen tüm ConnectionConsumers ' a ileti teslimi durdurulur. Bu durumda, etkilenen Bağlantıyla kayıtlı

ExceptionListener ' a bildirim gönderilir. Bir uygulamanın bu hata koşullarından kurtulması için iki yol vardır.

Genellikle, ConnectionConsumer bir iletiyi gitmeyen ileti kuyruğuna yeniden gönderemezse ya da QLOCAL ' den ileti okurken bir hatayla karşılaşırsa, bu tür ciddi bir hata oluşur.

Etkilenen Bağlantı ile kayıtlı herhangi bir ExceptionListener uyarıldığından, sorunun nedenini tanımlamak için bunları kullanabilirsiniz. Bazı durumlarda, sistem yöneticisinin sorunu çözmek için müdahale etmesi gerekir.

Bu hata koşullarından kurtulmak için aşağıdaki tekniklerden birini kullanın:

- Etkilenen tüm ConnectionConsumersiçin `close()` ' i arayın. Uygulama, yalnızca etkilenen tüm ConnectionConsumers kapatıldıktan ve sistem sorunları çözüldükten sonra yeni ConnectionConsumers oluşturabilir.
- Etkilenen tüm bağlantılarda `stop()` ' yı arayın. Tüm Bağlantılar durdurulduktan ve tüm sistem sorunları çözümlendikten sonra, uygulama `start()` Connections 'ı başarıyla uygulayabilir.

ASF ' de neden ve geribildirim kodları
Bir hatanın nedenini belirlemek için neden ve geribildirim kodlarını kullanın. ConnectionConsumer tarafından oluşturulan ortak neden kodları burada verilmiştir.

Bir hatanın nedenini belirlemek için aşağıdaki bilgileri kullanın:

- Herhangi bir rapor iletilerindeki geribildirim kodu
- MQDLH ' deki neden kodu, teslim mektubu kuyruğundaki tüm iletilere ilişkin

ConnectionConsumers , aşağıdaki neden kodlarını oluşturur.

MQRC_BACKOUT_THRESHOLD_ULAŞILDI (0x93A; 2362)

Neden
İleti, QLOCAL içinde tanımlanan geriletme eşiğine ulaştı, ancak geriletme kuyruğu tanımlanmadı.

Geriletme Kuyruğunu tanımlayamayacağınız altyapılarda, ileti JMStanımlı geriletme eşiği olan 20 'ye ulaştı.

Eylem
Bu istenmezse, ilgili QLOCAL için geriletme kuyruğunu tanımlayın. Ayrıca, birden çok geri tepmenin nedenini de arayın.

MQRC_MSG_NOT_MATCHED (0x93B; 2363)

Neden
Noktadan noktaya iletişim iletisinde, kuyruğu izleyen ConnectionConsumers (Bağlantı Sağlayıcıları) için seçicilerin hiçbiriyle eşleşmeyen bir ileti vardır. Başarımı korumak için ileti, teslim edilmeyen iletiler kuyruğuna yeniden gönderilir.

Eylem
Bu durumu önlemek için, kuyruğu kullanan ConnectionConsumers ' ın tüm iletilerle uğraşan bir seçici kümesi sağladığından emin olun ya da iletileri alıkoymak için QueueConnectionFactory 'yi ayarlayın.

Diğer bir seçenek olarak, iletinin kaynağını araştırın.

MQRC_JMS_FORMAT_ERROR (0x93C; 2364)

Neden
JMS , kuyruktaki iletiyi yorumlayamıyor.

Eylem
İletinin kökenini araştırın. JMS normalde, beklenmeyen biçimde iletileri BytesMessage ya da TextMessageolarak gönderir. Bazen, ileti çok kötü biçimlendirilirse bu başarısız olur.

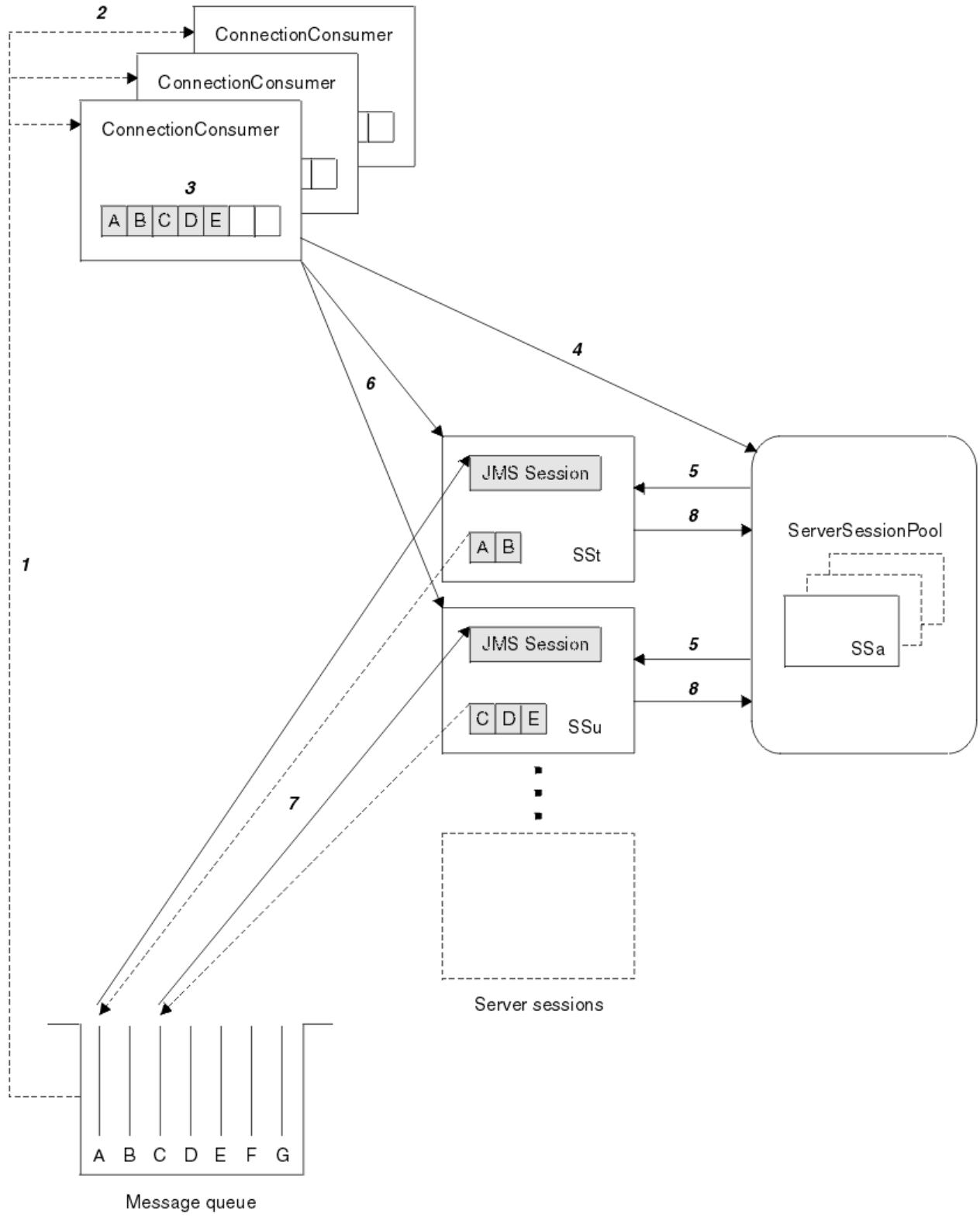
Bu alanlarda görüntülenen diğer kodlar, iletiyi geriletme kuyruğuna yeniden kuyruğa alma girişiminin başarısız oluşundan kaynaklanır. Bu durumda kod, yeniden istemenin başarısız olma nedenini açıklar. Bu hataların nedenini tanılamak için bkz. API> API tamamlama ve neden kodları.

Rapor iletisi ReplyToQ ' ya yerleştirilemezse, bu ileti gönderilmeyen iletiler kuyruğuna yerleştirilir. Bu durumda, MQMD ' nin geribildirim alanı bu konuda açıklandığı gibi tamamlanır. MQDLH 'deki neden alanı, rapor iletisinin ReplyToQ' ya neden yerleştirilemediğini açıklar.

AFS ' deki bir sunucu oturumu havuzunun işlevi

Bu konu, bir sunucu oturumu havuzunun işlevini özetler.

Şekil 45 sayfa 327 , ServerSessionPool ve ServerSession işlevselliğinin ilkelerini özetler.



Şekil 45. ServerSessionHavuzu ve ServerSession işlevselliği

1. ConnectionConsumers , kuyruktan ileti başvurularını alır.
2. Her ConnectionConsumer belirli ileti başvurularını seçer.
3. ConnectionConsumer arabelleği, seçilen ileti başvurularını tutar.
4. ConnectionConsumer , ServerSessionhavuzundan bir ya da daha çok ServerSessions isteğinde bulunur.

5. ServerSessions , ServerSessionhavuzundan ayrılır.
6. ConnectionConsumer , ServerSessions olanağına ileti başvuruları atar ve çalışan ServerSession iş parçacıklarını başlatır.
7. Her ServerSession , başvurulan iletilerini kuyruktan alır. Bunları, JMS Oturumu ile ilişkili MessageListener ' dan onMessage yöntemine iletir.
8. İşlemeyi tamamladıktan sonra, ServerSession havuza döndürülür.

Normalde bir uygulama sunucusu ServerSessionPool ve ServerSession işlevlerini sağlar.

z/OS IBM MQ classes for JMS ' in CICS Liberty JVM sunucusunda kullanılması

IBM MQ 9.1.0'den, bir CICS Liberty JVM sunucusunda çalışan Java programları, IBM MQ' e erişmek için IBM MQ classes for JMS ' i kullanabilir.

IBM MQ kaynak bağdaştırıcısının IBM MQ 9.1.0ya da sonraki bir sürümünü kullanıyor olmanız gerekir. Kaynak bağdaştırıcısını Fix Central adresinden edinebilirsiniz (bkz. “Kaynak bağdaştırıcısının Liberty içine kurulması” sayfa 428).

CICS 5.3 'de ve daha sonra kullanılabilir Liberty Profil JVM' lerinin iki çeşidi vardır; IBM MQ ile mümkün olan bağlantı tipleri aşağıdaki gibi kısıtlanmıştır:

CICS Liberty Standart

- IBM MQ kaynak bağdaştırıcısı, CLIENT kipinde herhangi bir hizmet içi IBM MQ sürümüne bağlanabilir
- IBM MQ kaynak bağdaştırıcısı, aynı CICS bölgesinden aynı kuyruk yöneticisine CICS bağlantısı (etkin CICS MQCONN kaynak tanımlaması) olmadığında, BINDINGS kipindeki herhangi bir IBM MQ for z/OS hizmet içi sürümüne bağlanabilir.

CICS Liberty Tümleşik

- IBM MQ kaynak bağdaştırıcısı, CLIENT kipinde herhangi bir hizmet içi IBM MQ sürümüne bağlanabilir.
- BINDINGS kipi bağlantısı desteklenmiyor.

Sisteminizi ayarlamaya ve yapılandırmaya ilişkin ayrıntılar için CICS belgelerinde Using IBM MQ classes for JMS in a Liberty JVM server başlıklı konuya bakın.

IMS içinde IBM MQ classes for JMS/ Jakarta Messaging kullanılması

Bir IMS ortamında standartlara dayalı ileti sistemi desteği, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingkullanımı aracılığıyla sağlanır.

Kuruluşunuzun kullandığı IMS sistemine ilişkin sistem gereksinimlerini denetleyin. Daha fazla bilgi için bkz. IMS 15.2 .

Bu konu kümesi, IMS ortamında IBM MQ classes for JMS ' in nasıl ayarlanacağını ve klasik (JMS 1.1) ve basitleştirilmiş (JMS 2.0) arabirimler kullanılırken geçerli olan API kısıtlamalarını açıklar. API ' ye özgü bilgilerin listesi için bkz. “JMS API kısıtlamaları” sayfa 333 .

Not: Benzer sınırlamalar, eski (JMS 1.0.2) etki alanına özgü arabirimler için de geçerlidir, ancak burada özel olarak açıklanmaz.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Aynı uygulamada hem Jakarta Messaging 3.0 API hem de JMS 2.0 API ' nin kullanılması desteklenmez. Daha fazla bilgi için, bkz. JMS/Jakarta İleti Sistemi için IBM MQ sınıflarının kullanılması.

Desteklenen IMS bağımlı bölgeleri

Aşağıdaki bağımlı bölge tipleri desteklenir:

- MPR
- BMP
- IFP
- JMP 31 ve 64 bit Java sanal makineleri (JVM)
- JBP 31 ve 64 bit JVM ' ler

Aşağıdaki konularda özellikle belirtilmedikçe, IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging tüm bölge tiplerinde aynı şekilde davranır.

Desteklenen Java Sanal Makineleri

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging için IBM Runtime Environment, Java Technology Edition 8 gereklidir. IBM Semeru Runtime Certified Edition for z/OS, Sürüm 11 desteklenmez.

Diğer kısıtlamalar

IMS ortamında IBM MQ classes for JMS kullanılırken aşağıdaki kısıtlamalar geçerlidir:

- İstemci kipi bağlantıları desteklenmiyor.
- Bağlantılar yalnızca IBM MQ ileti alışverişi sağlayıcısı `Normal`kipi kullanılarak IBM MQ 8.0 kuyruk yöneticilerine desteklenir.

  Bağlantı üreticisinde **PROVIDERVERSION** özniteliği belirtilmemiş ya da yediden büyük ya da yediye eşit bir değer olmalıdır.
- Herhangi bir XA bağlantı üreticisinin (örneğin, `com.ibm.mq.jms.MQXAConnectionFactory`) kullanılması desteklenmez.

İlgili görevler

IBM MQ öğesini IMS olarak tanımlama

IMS bağdaştırıcısının IBM MQ classes for JMS/Jakarta Messaging ile kullanılmak üzere ayarlanması

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , diğer programlama dilleri tarafından kullanılan aynı IBM MQ-IMS bağdaştırıcısını kullanır. Bu bağdaştırıcı, IMS Dış Altsistem Bağlantı Olanağı 'nı (ESAF) kullanır.

Başlamadan önce

Aşağıdaki yordamı tamamlamadan önce, ilgili kuyruk yöneticileri için IMS bağdaştırıcısını ve IMS bağdaştırıcısının ayarlanmasıkonusunda açıklandığı gibi IMS denetim ve bağımlı bölgelerini yapılandırmanız gerekir.

Uyarı: Dinamik sınırlı kod öbeğine başka amaçlar için gereksinim duymadığınız sürece, dinamik kod öbeğinin oluşturulmasını açıklayan adımı gerçekleştirmenize gerek yoktur.

IMS bağdaştırıcısını yapılandırdıktan sonra aşağıdaki yordamı gerçekleştirin.

Yordam

1. Bağımlı bölgenizdeki JCL ' de (örneğin, DFSJVMEV) ENVIRON parametresi tarafından başvurulan IMS PROCLIB üyenizdeki LIBPATH değişkenini, IBM MQ classes for JMS yerel kitaplıklarını içerecek şekilde güncelleyin.

   Yani, `libmqjims.so`dizinini içeren zFS dizini. Örneğin, DFSJVMEV aşağıdaki gibi görünebilir; burada son satır, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging yerel kitaplıklarını içeren dizindir:

```
LIBPATH=>
/java/latest/bin/j9vm:>
```

```
/java/latest/bin:>
/ims/latest/dbdc/imsjava/classic/lib:>
/ims/latest/dbdc/imsjava/lib:>
/mqm/latest/java/lib
```

2. `java.class.path` seçeneğini güncelleyerek, IMS bağımlı bölgeniz tarafından kullanılan JVM ' nin sınıf yoluna IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging ekleyin.

   Bunu, IMS PROCLIB veri kümesinin DFSJVMMS üyesiiçindeki yönergeleri izleyerek yapın.

   Örneğin, kalın çizginin güncellemeyi gösterdiği yerde, aşağıdakileri kullanabilirsiniz:

V 9.3.0 V 9.3.0 JM 3.0
```
-Djava.class.path=/ims/latest/dbdc/imsjava/imsutm.jar:/ims/latest/dbdc/imsjava/imsudb.jar:
/mqm/latest/java/lib/com.ibm.mq.jakarta.client.jar
```

JMS 2.0
```
-Djava.class.path=/ims/latest/dbdc/imsjava/imsutm.jar:/ims/latest/dbdc/imsjava/imsudb.jar:
/mqm/latest/java/lib/com.ibm.mq.allclient.jar
```

   Not: IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingiçeren dizinde birçok farklı jar dosyası bulunsa da, yalnızca `com.ibm.mq.allclient.jar` (JMS 2.0) ya da `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0)gerekir.

3. IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingürününü kullanacak IMS bağımlı bölgeleri durdurun ve yeniden başlatın.

Sonraki adım

Bağlantı üreticilerini ve hedeflerini oluşturun ve yapılandırın.

Bağlantı üreticileri ve hedeflerinin IBM MQ uygulamalarının somutlaştırılması için üç olası yaklaşım vardır. Ayrıntılar için bkz. “Bağlantı üreticileri ve hedefleri oluşturma ve yapılandırma” sayfa 197.

Bu üç yaklaşımın tümünün bir IMS ortamında geçerli olduğunu unutmayın.

İlgili kavramlar

IMS bağdaştırıcısının ayarlanması

IBM MQ öğesini IMS olarak tanımlama

İşlemsel davranış

Bir IMS ortamında IBM MQ classes for JMS tarafından gönderilen ve alınan iletiler her zaman geçerli görevde etkin olan IMS iş birimiyle (UOW) ilişkilendirilir.

Bu UOW, yalnızca `com.ibm.ims.dli.tm.Transaction` nesnesinin bir örneğinde kesinleştirme ya da geriye işleme yöntemleri çağrılarak ya da UOW örtük olarak kesinleştirildiğinde olağan şekilde sona eren IMS göreviyle tamamlanabilir. IMS görevi olağandışı biterse, UOW geriye işlenir.

Bunun sonucu olarak, `Connection.createSession`ya da `ConnectionFactory.createContext` yöntemlerinden herhangi biri çağrılırken **`transacted`** ve **`acknowledgeMode`** bağımsız değişkenlerinin değerleri yoksayılır. Ayrıca, aşağıdaki yöntemler desteklenmez. Aşağıdaki yöntemlerden herhangi birinin çağrılması, oturum vakasında `IllegalStateException` ile sonuçlanır:

- `javax.jms.Session.commit()`
- `javax.jms.Session.recover()`
- `javax.jms.Session.rollback()`

ve JMS bağlam durumundaki bir `IllegalStateRuntimeSession` :

- `javax.jms.JMSContext.commit()`
- `javax.jms.JMSContext.recover()`
- `javax.jms.JMSContext.rollback()`

Bu davranışın bir istisnası vardır. Bir oturum ya da JMS bağlamı aşağıdaki mekanizmalardan biri kullanılarak yaratıldıysa:

- `Connection.createSession(false, Session.AUTO_ACKNOWLEDGE)`
- `Connection.createSession(Session.AUTO_ACKNOWLEDGE)`
- `ConnectionFactory.createContext(JMSContext.AUTO_ACKNOWLEDGE)`

o oturumun ya da JMS bağlamının davranışı aşağıdaki gibidir:

- Gönderilen iletiler, IMS UOW dışına gönderilir. Başka bir deyişle, bunlar hedef hedefte hemen kullanılabilir olacak ya da sağlanan teslim gecikme aralığı tamamlandığında.
- Oturumu ya da JMS bağlamını yaratan bağlantı üreticisine SYNCPOINTALLRECEIVED özelliği belirtilmedikçe, kalıcı olmayan iletiler IMS UOW dışında alınır.
- Kalıcı iletiler her zaman IMS UOW içinde alınır.

Bu, örneğin, UOW geri alınsa bile bir denetim iletisini bir kuyruğa yazmak istiyorsanız yararlı olabilir.

IMS eşitleme noktalarının etkileri

ESAF ' yi kullanan var olan IBM MQ bağdaştırıcısı desteği üzerine IBM MQ classes for JMS oluşturması. Bu, bir eşitleme noktası oluştuğunda IMS bağdaştırıcısı tarafından kapatılan tüm açık tutamaçlar da dahil olmak üzere, belgelenmiş davranışın geçerli olduğu anlamına gelir.

Ek bilgi için bkz. “IMS uygulamalarında eşitleme noktaları” sayfa 67 .

Bu noktayı göstermek için, JMP ortamında çalışan aşağıdaki kodu göz önünde bulundurun. `mp.send()` ' e yapılan ikinci çağrı, `messageQueue.getUnique(inputMessage)` kodunun tüm açık IBM MQ bağlantısıyla ve nesne tanıtıcılarının kapatılması ile sonuçlandıkça `JMSException` ile sonuçlanır.

`getUnique()` çağrısı `Transaction.commit()`ile değiştirildiyse, ancak `Transaction.rollback()` kullanılıyorsa, benzer davranış gözlemlenir.

```
//Create a connection to queue manager MQ21.
MQConnectionFactory cf = new MQConnectionFactory();
cf.setQueueManager("MQ21");

Connection c = cf.createConnection();
Session s = c.createSession();

//Send a message to MQ queue Q1.
Queue q = new MQQueue("Q1");
MessageProducer mp = s.createProducer(q);
TextMessage m = s.createTextMessage("Hello world!");
mp.send(m);

//Get a message from an IMS message queue. This results in a GU call
//which results in all MQ handles being closed.
Application a = ApplicationFactory.createApplication();
MessageQueue messageQueue = a.getMessageQueue();
IOMessage inputMessage = a.getIOMessage(MESSAGE_CLASS_NAME);
messageQueue.getUnique(inputMessage);

//This attempt to send another message will result in a JMSException containing a
//MQRC_HCONN_ERROR as the connection/handle has been closed.
mp.send(m);
```

Bu senaryoda kullanılacak doğru kod aşağıdaki gibidir. Bu durumda, `getUnique()`çağrılmadan önce IBM MQ bağlantısı kapatılır. Daha sonra, başka bir ileti göndermek için bağlantı ve oturum yeniden yaratılır.

```
//Create a connection to queue manager MQ21.
MQConnectionFactory cf = new MQConnectionFactory();
cf.setQueueManager("MQ21");

Connection c = cf.createConnection();
Session s = c.createSession();

//Send a message to MQ queue Q1.
Queue q = new MQQueue("Q1");
MessageProducer mp = s.createProducer(q);
```

```
TextMessage m = s.createTextMessage("Hello world!");
mp.send(m);

//Close the connection to MQ, which closes all MQ object handles.
//The send of the message will be committed by the subsequent GU call.
c.close();
c = null;
s = null;
mp = null;

//Get a message from an IMS message queue. This results in a GU call.
Application a = ApplicationFactory.createApplication();
MessageQueue messageQueue = a.getMessageQueue();
IOMessage inputMessage = a.getIOMessage(MESSAGE_CLASS_NAME);
messageQueue.getUnique(inputMessage);

//Re-create the connection to MQ and send another message;
c = cf.createConnection();
s = c.createSession();
mp = s.createProducer(q);
m = s.createTextMessage("Hello world 2!");
mp.send(m);
```

IMS bağdaştırıcısı kullanılırken dikkat edilmesi gereken noktalar

Aşağıdaki kısıtlamaları dikkate almanız gerekir. Her kuyruk yöneticisi için tek bir bağlantı tanıtıcısı olabilir. Hem JMS hem de yerel kod kullanılırken IBM MQ ile etkileşimde bazı sonuçlar vardır. Bağlantı kimlik doğrulaması ve yetkilendirmesi sınırlamaları vardır.

Her kuyruk yöneticisi için bir bağlantı tanıtıcısı

IMS bağımlı bölgelerinde belirli bir kuyruk yöneticisiyle aynı anda yalnızca bir bağlantı tanıtıcısı kullanılabilir. Aynı kuyruk yöneticisine bağlanma girişimleri varsa, var olan tanıtıcıyı yeniden kullanın.

Bu davranış, yalnızca IBM MQ classes for JMSkullanan bir uygulamada herhangi bir soruna neden olmamalıdır; bu davranış, COBOL ya da C gibi dillerde yazılmış yerel kodda hem IBM MQ classes for JMS hem de MQI kullanıldığında IBM MQile etkileşim kuran uygulamalarda sorunlara neden olabilir.

Hem JMS hem de yerel kod kullanılırken IBM MQ ile etkileşimde bulunmanın etkileri

Yerli ya da Java kodundan ayrılmadan önce, hem IBM MQ işlevini kullanan Java kodu, hem de yerel kod ve IBM MQ bağlantısı kapatılmadığında sorunlar oluşabilir.

Örneğin, aşağıdaki sözde kodda, bir kuyruk yöneticisiyle bağlantı tanıtıcısı başlangıçta Java kodunda IBM MQ classes for JMSkullanılarak oluşturulur. Bağlantı tanıtıcısı COBOL kodunda yeniden kullanılır ve MQDISC çağrısı tarafından geçersiz kılınır.

IBM MQ classes for JMS , bir sonraki kez MQRC_HCONN_ERROR neden koduyla bir `JMSException` bağlantı tanıtıcısını kullanır.

```
COBOL code running in message processing region
    Use the Java Native Interface (JNI) to call Java code
       Create MQ connection and session - this creates an MQ connection handle
       Send message to MQ queue
       Store connection and session in static variable
       Return to COBOL code

    MQCONN - picks up MQ connection handle established in Java code
    MQDISC - invalidates connection handle

    Use the Java Native Interface (JNI) to call Java code
       Get session from static variable
       Create a message consumer - fails as connection handle invalidated
```

MQRC_HCONN_ERROR ile sonuçlanabilecek benzer kullanım örüntüleri vardır.

IBM MQ bağlantı tanıtıcıları yerel kod ile Java kod arasında paylaşılabilirken (örneğin, MQDISC çağrısı olmasaydı önceki örnek işe yarar), en iyi uygulama Java ' dan yerel koda ya da tersi yönde geçmeden önce bağlantı tanıtıcılarının kapatılmasıdır.

Bağlantı kimlik doğrulaması ve yetkilendirmesi

JMS belirtimi, bir bağlantı ya da JMS bağlam nesnesi oluşturulurken kimlik doğrulama ve yetkilendirme için bir kullanıcı adı ve parola belirtilmesini sağlar.

Bu, IMS ortamında desteklenmez. Bir kullanıcı adı ve parola belirtirken bağlantı yaratma girişimi `JMS Exception` yayınlanmasına neden olur. Bir kullanıcı adı ve parola belirtirken JMS bağlamı yaratma girişimi, bir `JMSRuntimeException` yayınlanmasına neden olur.

Bunun yerine, bir IMS ortamından IBM MQ ' e bağlanırken kimlik doğrulama ve yetkilendirme için var olan mekanizmalar kullanılmalıdır.

Daha fazla bilgi için bkz. z/OSüzerinde güvenliği ayarlama. Özellikle, kullanılabilecek kullanıcı kimliklerini açıklayan Güvenlik denetimi için kullanıcı kimliklerikonusuna bakın.

İlgili görevler

z/OS üzerinde güvenliğin ayarlanması

JMS API kısıtlamaları

JMS belirtimi perspektifinden, IBM MQ classes for JMS IMS her zaman devam eden bir JTA işlemi olan Java EE ya da Jakarta EE uyumlu bir uygulama sunucusu olarak işlem görür.

Örneğin, JMS belirtimi JEE EJB 'de ya da Web kapsayıcısında JTA işlemi devam ederken çağıramayacağınız belirttiğinden, IMS' de `javax.jms.Session.commit()` ' i hiçbir zaman çağıramazsınız.

Bu, “İşlemsel davranış” sayfa 330içinde açıklananlara ek olarak JMS API 'si için aşağıdaki kısıtlamalarla sonuçlanır.

Klasik API kısıtlamaları

- `javax.jms.Connection.createConnectionConsumer(javax.jms.Destination, String, javax.jms.ServerSessionPool, int)` her zaman bir `JMSException`atar.
- `javax.jms.Connection.createDurableConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` her zaman bir `JMSException`atar.
- Bağlantının etkin bir oturumu varsa, `javax.jms.Connection.createSession` değişkeninin üçü de her zaman bir `JMSException` oluşturur.
- `javax.jms.Connection.createSharedConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` her zaman bir `JMSException`atar.
- `javax.jms.Connection.createSharedDurableConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` her zaman bir `JMSException`atar.
- `javax.jms.Connection.setClientID()` her zaman bir `JMSException`atar.
- `javax.jms.Connection.setExceptionListener(javax.jms.ExceptionListener)` her zaman bir `JMSException`atar.
- `javax.jms.Connection.stop()` her zaman bir `JMSException`atar.
- `javax.jms.MessageConsumer.setMessageListener(javax.jms.MessageListener)` her zaman bir `JMSException`atar.
- `javax.jms.MessageConsumer.getMessageListener()` her zaman bir `JMSException`atar.
- `javax.jms.MessageProducer.send(javax.jms.Destination, javax.jms.Message,javax.jms.CompletionListener)` her zaman bir `JMSException`atar.
- `javax.jms.MessageProducer.send(javax.jms.Destination, javax.jms.Message, int, int, long, javax.jms.CompletionListener)` her zaman bir `JMSException`atar.
- `javax.jms.MessageProducer.send(javax.jms.Message, int, int, long, javax.jms.CompletionListener)` her zaman bir `JMSException`atar.
- `javax.jms.MessageProducer.send(javax.jms.Message,javax.jms.CompletionListener )` her zaman bir `JMSException`atar.

- `javax.jms.Session.run()` her zaman bir `JMSRuntimeException`atar.
- `javax.jms.Session.setMessageListener(javax.jms.MessageListener)` her zaman bir `JMSException`atar.
- `javax.jms.Session.getMessageListener()` her zaman bir `JMSException`atar.

Basitleştirilmiş API kısıtlamaları

- `javax.jms.JMSContext.createContext(int)` her zaman bir `JMSRuntimeException`atar.
- `javax.jms.JMSContext.setClientID(String)` her zaman bir `JMSRuntimeException`atar.
- `javax.jms.JMSContext.setExceptionListener(javax.jms.ExceptionListener)` her zaman bir `JMSRuntimeException`atar.
- `javax.jms.JMSContext.stop()` her zaman bir `JMSRuntimeException`atar.
- `javax.jms.JMSProducer.setAsync(javax.jms.CompletionListener)` her zaman bir `JMSRuntimeException`atar.

kullanmaIBM MQ classes for Java

Bir Java ortamında IBM MQ kullanın. IBM MQ classes for Java , bir Java uygulamasının IBM MQ istemcisi olarak IBM MQ ' e bağlanmasına ya da IBM MQ kuyruk yöneticisine doğrudan bağlanmasına izin verir.

Not:

Stabilized IBM , IBM MQ classes for Java üzerinde başka geliştirme yapmayacaktır ve bunlar IBM MQ 8.0içinde gönderilen düzeyde işlevsel olarak sabitlenecektir. IBM MQ classes for Java kullanan var olan uygulamalar tam olarak desteklenmeye devam eder, ancak yeni özellikler eklenmez ve geliştirme istekleri reddedilir. Tam olarak desteklenen, IBM MQ Sistem Gereksinimlerinde yapılan değişikliklerle birlikte hataların düzeltileceği anlamına gelir.

IBM MQ classes for Java , IMSiçinde desteklenmez.

IBM MQ classes for Java , WebSphere Libertyiçinde desteklenmez. Bunlar IBM MQ Liberty ileti sistemi özelliğiyle ya da genel JCA desteğiyle birlikte kullanılmamalıdır. Daha fazla bilgi için bakınız: Using WebSphere MQ Java Interfaces in J2EE/JEE Environments.

IBM MQ classes for Java , Java uygulamalarının IBM MQ kaynaklarına erişmek için kullanabileceği üç alternatif API ' den biridir. Diğer API ' ler şunlardır:

- V 9.3.0 V 9.3.0 JM 3.0 IBM MQ classes for Jakarta Messaging
- JMS 2.0 IBM MQ classes for JMS

Daha fazla bilgi için bkz “IBM MQ 'e Java ' dan erişme-API Seçimi” sayfa 81.

IBM MQ 9.3' den IBM MQ classes for Java , Java 8ile oluşturulur. Java 8 yürütme ortamı, daha önceki sınıf dosyası sürümlerinin çalıştırılmasını destekler.

IBM MQ classes for Java , yerel IBM MQ API 'si olan İleti Kuyruğu Arabirimi 'ni (MQI) kapsülleyin ve IBM MQile C++ ve .NET arabirimlerine benzer bir nesne modeli kullanın.

Programlanabilir seçenekler, IBM MQ classes for Java 'in IBM MQ ' e aşağıdaki yollardan biriyle bağlanmasını sağlar:

- İletim Denetimi İletişim Kuralı/Internet Protocol (TCP/IP) kullanılarak istemci kipinde IBM MQ MQI client olarak
- bağ tanımlama kipinde, Java Native Interface (JNI) kullanılarak doğrudan IBM MQ ile bağlantı kurulması

Not: Otomatik istemci yeniden bağlantısı IBM MQ classes for Javatarafından desteklenmez.

İstemci kipi bağlantısı

IBM MQ classes for Java uygulaması, istemci kipini kullanarak desteklenen herhangi bir kuyruk yöneticisine bağlanabilir.

İstemci kipinde bir kuyruk yöneticisine bağlanmak için IBM MQ classes for Java uygulaması, kuyruk yöneticisinin çalıştığı sistemde ya da farklı bir sistemde çalışabilir. Her bir durumda IBM MQ classes for Java , TCP/IP üzerinden kuyruk yöneticisine bağlanır.

İstemci kipi bağlantılarını kullanmak üzere uygulamaların nasıl yazılacağı hakkında daha fazla bilgi için bkz. “IBM MQ classes for Java bağlantı kipleri” sayfa 359.

Bağ tanımları kipi bağlantısı

Bağ tanımlama kipinde kullanıldığında IBM MQ classes for Java , bir ağ üzerinden iletişim kurmak yerine, var olan kuyruk yöneticisi API 'sine doğrudan çağırmak için Java Native Interface (JNI) olanağını kullanır. Çoğu ortamda, bağ tanımlama kipinde bağlanmak, TCP/IP iletişimi maliyetini önleyerek istemci kipinde bağlanmaya kıyasla IBM MQ classes for Java uygulamaları için daha iyi performans sağlar.

Bağ tanımlama kipinde bağlanmak için IBM MQ classes for Java kullanan uygulamaların, bağlandıkları kuyruk yöneticisiyle aynı sistemde çalışması gerekir.

IBM MQ classes for Java uygulamasını çalıştırmak için kullanılan Java Runtime Environment, IBM MQ classes for Java kitaplıklarını yüklemek için yapılandırılmalıdır; ek bilgi için bkz. “IBM MQ classes for Java Kitaplıklar” sayfa 344 .

Bağ tanımlama kipi bağlantılarını kullanmak üzere uygulamaların nasıl yazılacağı hakkında daha fazla bilgi için bkz. “IBM MQ classes for Java bağlantı kipleri” sayfa 359.

İlgili kavramlar

IBM MQ Java dil arabirimleri

“IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma” sayfa 78

IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , IBM MQile verilen Java ileti alışverişi sağlayıcılarıdır. JMS ve Jakarta Messaging belirtimlerinde tanımlanan arabirimleri gerçekleştirmenin yanı sıra, bu ileti alışverişi sağlayıcıları Java ileti sistemi API 'sine iki uzantı kümesi ekler.

İlgili görevler

IBM MQ classes for Java uygulamalarını izleme

Java ve JMS sorunlarının giderilmesi

Neden IBM MQ classes for Javakullanmalıyım?

Bir Java uygulaması, IBM MQ kaynaklarına erişmek için IBM MQ classes for Java ya da IBM MQ classes for JMS kullanabilir.

Not: V 9.3.0 V 9.3.0 IBM MQ classes for Java kullanan var olan uygulamalar tam olarak desteklenmeye devam etse de, yeni uygulamalar IBM MQ classes for Jakarta Messagingkullanmalıdır. IBM MQ' e zamanuyumsuz tüketim ve otomatik yeniden bağlanma gibi yakın zamanda eklenen özellikler IBM MQ classes for Javaiçinde bulunmaz, ancak IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messagingiçinde kullanılabilir. Daha fazla bilgi için bkz. “Neden IBM MQ classes for JMSkullanmalıyım?” sayfa 80 ve “Neden IBM MQ classes for Jakarta Messagingkullanmalıyım?” sayfa 79.

Not:

Stabilized IBM , IBM MQ classes for Java üzerinde başka geliştirme yapmayacaktır ve bunlar IBM MQ 8.0içinde gönderilen düzeyde işlevsel olarak sabitlenecektir. IBM MQ classes for Java kullanan var olan uygulamalar tam olarak desteklenmeye devam eder, ancak yeni özellikler eklenmez ve geliştirme istekleri reddedilir. Tam olarak desteklenen, IBM MQ Sistem Gereksinimlerinde yapılan değişikliklerle birlikte hataların düzeltileceği anlamına gelir.

IBM MQ classes for Java , IMSiçinde desteklenmez.

IBM MQ classes for Java , WebSphere Libertyiçinde desteklenmez. Bunlar IBM MQ Liberty ileti sistemi özelliğiyle ya da genel JCA desteğiyle birlikte kullanılmamalıdır. Daha fazla bilgi için bakınız: Using WebSphere MQ Java Interfaces in J2EE/JEE Environments.

İlgili kavramlar

“IBM MQ 'e Java ' dan erişme-API Seçimi” sayfa 81
IBM MQ , üç Java dil arabirimi sağlar.

IBM MQ classes for Java için önkoşullar

IBM MQ classes for Java' yi kullanmak için bazı diğer yazılım ürünlerine gereksinim duyarsınız.

IBM MQ classes for Javaile ilgili önkoşullar hakkında bilgi için IBM MQ için Sistem Gereksinimleri web sayfasına bakın.

IBM MQ classes for Java uygulamalarını geliştirmek için bir Java Development Kit (JDK) gerekir. İşletim sisteminizle desteklenen JDK ' lerin ayrıntılarını IBM MQ için Sistem Gereksinimleri bilgilerinde bulabilirsiniz.

IBM MQ classes for Java uygulamalarını çalıştırmak için aşağıdaki yazılım bileşenlerine gereksinim duyarsınız:

- Bir kuyruk yöneticisine bağlanan uygulamalar için IBM MQ kuyruk yöneticisi
- Uygulamaları çalıştırdığınız her sistem için bir Java Runtime Environment (JRE). IBM MQile uygun bir JRE sağlanır.
- IBM i İşletim sisteminin 30. seçeneği olan IBM i, QShell için
- z/OS z/OSiçin, z/OS UNIX System Services (z/OS UNIX)

FIPS 140-2 sertifikalı şifreleme modüllerini kullanmak için TLS bağlantılarına gereksinim duyarsanız, IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) gerekir. 1.4.2 sürümünde ya da sonraki sürümlerde her IBM JDK ve JRE IBMJSSEFIPS ' i içerir.

Java sanal makineniz (JVM) ve işletim sisteminizdeki TCP/IP uygulaması tarafından desteklenen IBM MQ classes for Java uygulamalarınızda IPv6 Internet Protocol sürüm 6 (IPv6) adreslerini kullanabilirsiniz.

Java EE içinde IBM MQ classes for Java uygulamalarının çalıştırılması

Java EEiçinde IBM MQ classes for Java kullanılmadan önce dikkate alınması gereken bazı kısıtlamalar ve tasarım konuları vardır.

IBM MQ classes for Java , bir Java Platform, Enterprise Edition (Java EE) ortamında kullanıldığında kısıtlamalar içerir. Java EE ortamında çalışan bir IBM MQ classes for Java uygulamasını tasarlarken, uygularken ve yönetirken dikkate alınması gereken ek noktalar da vardır. Bu kısıtlamalar ve dikkat edilmesi gereken noktalar aşağıdaki bölümlerde açıklanmıştır.

JTA işlemleri kısıtlamaları

IBM MQ classes for Java kullanan uygulamalar için desteklenen tek hareket yöneticisi IBM MQ ' in kendisidir. JTA denetimi altındaki bir uygulama IBM MQ classes for Java' dan yararlanabilse de, bu sınıflar aracılığıyla gerçekleştirilen her iş JTA iş birimleri tarafından kontrol edilmez. Bunun yerine, yerel iş birimlerini JTA arabirimleri aracılığıyla uygulama sunucusu tarafından yönetilenlerden ayrı olarak oluştururlar. Özellikle, JTA hareketinin geriye işlenmesi, gönderilen ya da alınan iletilerin geriye işlenmesiyle sonuçlanmaz. Bu kısıtlama, uygulama ya da bean tarafından yönetilen işlemler, taşıyıcı tarafından yönetilen işlemler ve tüm Java EE kapsayıcıları için geçerlidir. İleti alışverişi çalışmasını doğrudan doğruya uygulama sunucusu eşgüdümlü hareketler içinde IBM MQ ile gerçekleştirmek için IBM MQ classes for JMS kullanılmalıdır.

İş parçacığı yaratma

IBM MQ classes for Java , çeşitli işlemler için içeride iş parçacıkları oluşturur. Örneğin, yerel bir kuyruk yöneticisinde doğrudan çağırmak için BINDINGS kipinde çalışırken, çağrılar IBM MQ classes for Javatarafından dahili olarak oluşturulan bir 'worker' iş parçacığında yapılır. Diğer iş parçacıkları dahili olarak oluşturulabilir; örneğin, bir bağlantı havuzundan kullanılmayan bağlantıları temizlemek ya da sonlandırılan yayınlama/abone olma uygulamalarına ilişkin abonelikleri kaldırmak için.

Bazı Java EE uygulamaları (örneğin, EJB ve Web taşıyıcılarında çalışan) yeni iş parçacıkları yaratmamalıdır. Bunun yerine, uygulama sunucusu tarafından yönetilen ana uygulama iş parçacıklarında tüm işin gerçekleştirilmesi gerekir. Uygulamalar IBM MQ classes for Javakullandığında, uygulama sunucusu uygulama kodu ile IBM MQ classes for Java kodunu ayırt edemeyebilir, bu nedenle daha önce açıklanan iş parçacıkları uygulamanın kapsayıcı belirtimiyle uyumlu olmamasına neden olur. IBM MQ classes for JMS , bu Java EE belirtimlerini bozmaz ve bunun yerine kullanılabilir.

Güvenlik kısıtlamaları

Bir uygulama sunucusu tarafından uygulanan güvenlik ilkeleri, yeni denetim iş parçacıkları yaratma ve çalıştırma (önceki bölümlerde açıklandığı gibi) gibi IBM MQ classes for Java API tarafından üstlenilen bazı işlemleri önleyebilir.

Örneğin, uygulama sunucuları genellikle varsayılan olarak Java Security devre dışı bırakılmış olarak çalışır ve bazı uygulama sunucusuna özgü yapılandırmayla etkinleştirilmesine izin verir (bazı uygulama sunucuları, Java Security içinde kullanılan ilkelerin daha ayrıntılı yapılandırılmasına da izin verir). Java Güvenlik etkinleştirildiğinde IBM MQ classes for Java , uygulama sunucusu için tanımlanan Java Güvenlik ilkesi iş parçacığı kurallarını bozabilir ve API, çalışabilmesi için gerek duyduğu tüm iş parçacıklarını yaratamayabilir. İş parçacığı yönetimiyle ilgili sorunları önlemek için, Java güvenliğinin etkinleştirildiği ortamlarda IBM MQ classes for Java kullanımı desteklenmez.

Uygulama yalıtımında dikkate alınması gerekenler

Java EE ortamında uygulama çalıştırmanın amaçlanan bir yararı, uygulama yalıtımıdır. IBM MQ classes for Java uygulamasının tasarımı ve uygulanması, Java EE ortamından önce olur. IBM MQ classes for Java , uygulama yalıtımı kavramını desteklemeyen bir şekilde kullanılabilir. Bu alanda dikkat edilmesi gereken belirli örnekler şunlardır:

- MQEnvironment sınıfında durağan (JVM işlem çapında) ayarların kullanılması; örneğin:
  - bağlantı tanıtıcısı ve kimlik doğrulaması için kullanılacak kullanıcı kimliği ve parola
  - istemci bağlantıları için kullanılan anasistem adı, kapı ve kanal
  - Güvenli istemci bağlantıları için TLS yapılandırması

  Bir uygulamanın yararına MQEnvironment özelliklerinden herhangi birinin değiştirilmesi, aynı özellikleri kullanan diğer uygulamaları da etkiler. Java EEgibi bir çoklu uygulama ortamında çalışırken, süreç genelinde MQEnvironment sınıfında yapılandırılan özellikleri varsayılan olarak kullanmak yerine, her uygulamanın belirli bir özellik kümesiyle MQQueueManager nesneleri yaratarak kendi ayrı yapılanışını kullanması gerekir.
- MQEnvironment sınıfı, aynı JVM süreci içinde IBM MQ classes for Java kullanan tüm uygulamalar üzerinde genel olarak işlem yapan bir dizi durağan yöntem sunar ve belirli uygulamalar için bu davranışı geçersiz kılmanın bir yolu yoktur. Örnekler:
  - anahtar deposunun yeri gibi TLS özelliklerini yapılandırma
  - istemci kanalı çıkışlarını yapılandırma
  - tanılama izlemesini etkinleştirme ya da devre dışı bırakma
  - kuyruk yöneticilerine yönelik bağlantıların kullanımını eniyilemek için kullanılan varsayılan bağlantı havuzunu yönetme

  Bu yöntemlerin çağrılması, aynı Java EE ortamında çalışan tüm uygulamaları etkiler.

- Aynı kuyruk yöneticisine birden çok bağlantı yapma işlemini eniyilemek için bağlantı havuzlama etkinleştirilir. Varsayılan bağlantı havuzu yöneticisi işlem çapında ve birden çok uygulama tarafından paylaşılır. Bağlantı havuzu yapılanışında yapılan değişiklikler; örneğin, MQEnvironment.setDefaultConnectionManager() yöntemini kullanarak bir uygulama için varsayılan bağlantı yöneticisinin değiştirilmesi, aynı Java EE uygulama sunucusunda çalışan diğer uygulamaları etkiler.
- TLS, MQEnvironment sınıfı ve MQQueueManager nesne özellikleri kullanılarak IBM MQ classes for Java kullanılarak uygulamalar için yapılandırılır. Uygulama sunucusunun kendisinin yönetilen güvenlik yapılandırmasıyla bütünleştirilmez. Gerekli güvenlik düzeyinizi sağlamak ve uygulama sunucusu yapılandırmasını kullanmak için değil, IBM MQ classes for Java ' i uygun şekilde yapılandırdığınızdan emin olmanız gerekir.

Bağ tanımları kipi kısıtlamaları

IBM MQ ve WebSphere Application Server aynı makineye kurulabilir; böylece, WebSphere Application Server içinde gönderilen kuyruk yöneticisinin ve IBM MQ kaynak bağdaştırıcısının (RA) ana sürümleri farklı olur.

Kuyruk yöneticisi ve kaynak bağdaştırıcısı ana sürümleri farklıysa, bağ tanımları bağlantıları kullanılamaz. Kaynak bağdaştırıcısını kullanarak WebSphere Application Server ' den kuyruk yöneticisine yapılan her bağlantı istemci tipi bağlantılarını kullanmalıdır. Sürümler aynıysa bağ tanımları bağlantıları kullanılabilir.

IBM MQ classes for Java içinde karakter dizgisi dönüştürmeleri

IBM MQ classes for Java , karakter dizgisi dönüştürmesi için doğrudan CharsetEncoders ve CharsetDecoders karakterlerini kullanır. Karakter dizilimi dönüştürmesi için varsayılan davranış, iki sistem özellğiyle yapılandırılabilir. Eşlenebilir karakterler içeren iletilerin işlenmesi `com.ibm.mq.MQMD`aracılığıyla yapılandırılabilir.

IBM MQ 8.0öncesinde, IBM MQ classes for Java içindeki dizgi dönüştürmeleri `java.nio.charset.Charset.decode(ByteBuffer)` ve `Charset.encode(CharBuffer)` yöntemleri çağrılarak gerçekleştiriliyordu.

Bu yöntemlerden birinin kullanılması, bozuk biçimli ya da çevrilemeyen verilerin varsayılan olarak değiştirilmesine (REPLACE) neden olur. Bu davranış, uygulamalardaki hataları gizleyebilir ve çevrilmiş verilerde beklenmeyen karakterlere (örneğin, ?) yol açabilir.

IBM MQ 8.0' den bu tür sorunları daha erken ve daha etkili bir şekilde saptamak için IBM MQ classes for Java , CharsetEncoders ve CharsetDecoders öğelerini doğrudan kullanır ve bozuk biçimli ve çevrilemeyen verilerin işlenmesini belirtik olarak yapılandırır. Varsayılan davranış, uygun bir `MQException`yayınlayarak REPORT bu tür sorunlardır.

Yapılandırılıyor

UTF-16 ' dan (Javaiçinde kullanılan karakter gösterimi) UTF-8gibi bir yerel karakter kümesine çevrilmesi *kodlama*olarak adlandırılırken, tersi yönde çevirme *kod çözme*olarak adlandırılır.

Kod çözme işlemi, `CharsetDecoders`için varsayılan davranışı alır ve kural dışı durum yayınlanarak hataları raporlar.

Bir ayar, hem kodlama hem de kod çözme sırasında hata işlemeyi denetlemek üzere bir `java.nio.charset.CodingErrorAction` belirtmek için kullanılır. Kodlama sırasında, yerine koyma baytını ya da baytlarını denetlemek için başka bir ayar kullanılır. Kod çözme işlemlerinde varsayılan Java yerine koyma dizgisi kullanılır.

IBM MQ classes for Java içinde çevrilemeyen veri işlemenin yapılandırılması

IBM MQ 8.0' den `com.ibm.mq.MQMD` aşağıdaki iki alanı içerir:

byte [] unMappableDeğiştirme
Bir giriş karakteri çevrilemezse ve REPLACEbelirttiyseniz, kodlanmış dizgiye yazılacak bayt sırası.

Varsayılan: "?".getBytes()
Kod çözme işlemlerinde varsayılan Java yerine koyma dizgisi kullanılır.

java.nio.charset.CodingErrorAction unmappableAction
Kodlama ve kod çözme sırasında çevrilemeyen veriler için yapılacak işlemi belirtir:

Varsayılan: CodingErrorAction.REPORT;

Sistem varsayılanlarını ayarlamak için sistem özellikleri

IBM MQ 8.0' den, karakter dizgisi dönüşümüne ilişkin varsayılan davranışı yapılandırmak için aşağıdaki iki Java sistem özelliği kullanılabilir.

com.ibm.mq.cfg.jmqi.UnmappableCharacterAction
Kodlama ve kod çözme sırasında çevrilemeyen veriler için yapılacak işlemi belirtir. Değer REPORT, REPLACEya da IGNOREolabilir.

com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement
Kodlama işleminde bir karakter eşlenemediğinde uygulanacak yerine koyma baytlarını belirler ya da alır. Kod çözme işlemlerinde varsayılan Java yerine koyma dizgisi kullanılır.

Java karakteri ile yerli byte gösterimleri arasındaki karışıklığı önlemek için,
yerel karakter takımındaki yerine koyma byte 'ını gösteren ondalık sayı olarak
`com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement` belirtmeniz gerekir.

Örneğin, yerel karakter takımı ASCII tabanlıysa (örneğin, ISO-8859-1), yerel karakter takımı EBCDIC ise 111 ise ?ondalık değeri 63 'tür.

Not: Bir MQMD ya da MQMessage nesnesinde **unmappableAction** ya da **unMappableReplacement** alanları ayarlandıysa, bu alanların değerlerinin Java sistem özelliklerinden öncelikli olduğunu unutmayın. Bu, Java sistem özellikleri tarafından belirtilen değerlerin, gerekirse her bir ileti için geçersiz kılınmasına olanak sağlar.

İlgili kavramlar
“IBM MQ classes for JMS içinde karakter dizgisi dönüştürmeleri” sayfa 133
IBM MQ classes for JMS , karakter dizgisi dönüştürmesi için doğrudan CharsetEncoders ve CharsetDecoders karakterlerini kullanır. Karakter dizilimi dönüştürmesi için varsayılan davranış, iki sistem özellğiyle yapılandırılabilir. Eşlenebilir karakterler içeren iletilerin işlenmesi, UnmappableCharacterişlemini ve değiştirme baytlarını ayarlamak için ileti özellikleriyle yapılandırılabilir.

Kurma ve YapılandırmaIBM MQ classes for Java

Bu bölümde, IBM MQ classes for Javaürününü kurduğunuzda oluşturulan dizinler ve dosyalar açıklanır ve kuruluştan sonra IBM MQ classes for Java ' un nasıl yapılandırılacağı anlatılır.

IBM MQ classes for Java için kurulu olan

En son IBM MQ classes for Java sürümü IBM MQile kurulur. Bunun yapıldığından emin olmak için varsayılan kuruluş seçeneklerini geçersiz kılmanız gerekebilir.

IBM MQ kuruluşu hakkında daha fazla bilgi için bkz:

- Multi Kuruluş IBM MQ
- z/OS IBM MQ for z/OS ürününün kurulması

IBM MQ classes for Java , Java arşiv (JAR) dosyaları, `com.ibm.mq.jar`ve `com.ibm.mq.jmqi.jar`içinde bulunur.

`com.ibm.mq.headers.jar`JAR dosyasında, Programlanabilir Komut Biçimi (PCF) gibi standart ileti üstbilgileri için destek bulunur.

`com.ibm.mq.pcf.jar`JAR dosyasında Programlanabilir Komut Biçimi (PCF) desteği bulunur.

Not: IBM MQ classes for Java ' in bir uygulama sunucusu içinde kullanılması önerilmez. Bu ortamda çalışırken geçerli olan kısıtlamalar hakkında bilgi için bkz. “Java EE içinde IBM MQ classes for Java

uygulamalarının çalıştırılması” sayfa 336. Daha fazla bilgi için bkz. WebSphere MQ Java Arabirimlerinin J2EE/JEE Ortamlarında Kullanılması.

Önemli: “IBM MQ classes for Java yeniden dizinlenebilir JAR dosyaları” sayfa 340içinde açıklanan yeniden taşınabilir JAR dosyalarının yanı sıra, IBM MQ classes for Java JAR dosyalarının ya da yerel kitaplıkların başka makinelere kopyalanması ya da IBM MQ classes for Java ' in kurulu olduğu bir makinede farklı bir yere kopyalanması desteklenmez.

IBM MQ classes for Java yeniden dizinlenebilir JAR dosyaları
Yeniden taşınabilir JAR dosyaları, IBM MQ classes for Javakomutunu çalıştırması gereken sistemlere taşınabilir.

Önemli:

- Relocatable JAR dosyalarındaaçıklanan yeniden taşınabilir JAR dosyalarının yanı sıra, IBM MQ classes for Java JAR dosyalarının ya da yerel kitaplıkların başka makinelere kopyalanması ya da IBM MQ classes for Java ' in kurulu olduğu bir makinede farklı bir yere kopyalanması desteklenmez.
- Sınıf yükleyici çakışmalarını önlemek için, relocatable JAR dosyalarının aynı Java yürütme ortamı içindeki birden çok uygulama içinde paketlenmesi önerilmez. Bu senaryoda, IBM MQ relocatable JAR dosyalarını Java yürütme ortamının sınıf yolunda (classpath) kullanılabilir kılmayı düşünün.
- Yeniden yerleştirilebilir JAR dosyalarını, WebSphere Application Servergibi Java EE uygulama sunucularında konuşlandırılan uygulamalara eklemeyin. Bu ortamlarda, bunun yerine IBM MQ kaynak bağdaştırıcısı konuşlandırılmalı ve kullanılmalıdır; bu, IBM MQ classes for Javaöğesini içerir. WebSphere Application Server ' in IBM MQ kaynak bağdaştırıcısını yerleştirdiğini, bu nedenle bu ortama el ile yerleştirmeye gerek olmadığını unutmayın. Buna ek olarak, IBM MQ classes for Java WebSphere Libertyiçinde desteklenmez. Daha fazla bilgi için bkz “Liberty ve IBM MQ kaynak bağdaştırıcısı” sayfa 425.
- Yeniden dizinlenebilir JAR dosyalarını uygulamalarınızda paketliyorsanız, tüm önkoşul JAR dosyalarını Relocatable JAR filesbaşlıklı konuda açıklandığı gibi eklemeyi doğrulayın. IBM MQ classes for Java ' in güncel ve bilinen sorunların yeniden aracılık edildiğinden emin olmak için, paket JAR dosyalarını uygulama bakımının bir parçası olarak güncellemek için uygun yordamlara sahip olduğunuzdan da emin olmanız gerekir.

Yeniden dizinlenebilir JAR dosyaları

Bir işletme içinde, aşağıdaki dosyalar IBM MQ classes for Java uygulamalarını çalıştırması gereken sistemlere taşınabilir:

- JMS 2.0 `com.ibm.mq.allclient.jar` “1” sayfa 341
- V 9.3.0 V 9.3.0 JM 3.0 `com.ibm.mq.jakarta.client.jar` “2” sayfa 341
- Removed V 9.3.3 `com.ibm.mq.traceControl.jar`
- V 9.3.5 `bcpkix-jdk18on.jar` “3” sayfa 341
- V 9.3.0 V 9.3.0 `bcpkix-jdk15to18.jar` “4” sayfa 341
- V 9.3.5 `bcprov-jdk18on.jar` “3” sayfa 341
- V 9.3.0 V 9.3.0 `bcprov-jdk15to18.jar` “4” sayfa 341
- V 9.3.5 `bcutil-jdk18on.jar` “3” sayfa 341
- V 9.3.0 V 9.3.0 `bcutil-jdk15to18.jar` “4” sayfa 341
- V 9.3.3 `jackson-annotations.jar`
- V 9.3.3 `jackson-core.jar`
- V 9.3.3 `jackson-databind.jar`

- `org.json.jar`

Notlar:

1. JMS 2.0 ve JMS 1.1
2. Jakarta Messaging 3.0
3. Continuous Delivery Kaynak: IBM MQ 9.3.5
4. Long Term Support ve Continuous Delivery önce IBM MQ 9.3.5

Bouncy Castle güvenlik sağlayıcısı ve CMS destek JAR dosyaları

Bouncy Castle güvenlik sağlayıcısı ve CMS destek JAR dosyaları gereklidir. Daha fazla bilgi için bkz. Support for non-IBM JRE with AMS.

V 9.3.5 Continuous Delivery from IBM MQ 9.3.5için aşağıdaki JAR dosyaları gereklidir:

- `bcpkix-jdk18on.jar`
- `bcprov-jdk18on.jar`
- `bcutil-jdk18on.jar`

LTS IBM MQ 9.3.5öncesinde Long Term Support ve Continuous Delivery için aşağıdaki JAR dosyaları gereklidir:

- `bcpkix-jdk15to18.jar`
- `bcprov-jdk15to18.jar`
- `bcutil-jdk15to18.jar`

org.json.jar

IBM MQ classes for Java uygulamanız JSON biçiminde bir CCDT kullanıyorsa `org.json.jar` dosyası gereklidir.

com.ibm.mq.allclient.jar ve com.ibm.mq.jakarta.client.jar

`com.ibm.mq.allclient.jar` ve `com.ibm.mq.jakarta.client.jar` dosyaları, IBM MQ classes for JMS, IBM MQ classes for Javave PCF ve Üstbilgiler Sınıflarını içerir. Bu dosyaları yeni bir konuma taşırsanız, bu yeni konumu yeni IBM MQ Düzeltme Paketleriyle korumak için gereken adımları attığınızdan emin olun. Ayrıca, geçici bir düzeltme alıyorsanız, dosyaların kullanılmasının IBM Desteği tarafından bilindiğinden emin olun.

`com.ibm.mq.allclient.jar` dosyasının ya da `com.ibm.mq.jakarta.client.jar` dosyasının sürümünü belirlemek için aşağıdaki komutu kullanın:

V 9.3.0 V 9.3.0 JM 3.0

```
java -jar com.ibm.mq.jakarta.client.jar
```

JMS 2.0

```
java -jar com.ibm.mq.allclient.jar
```

Aşağıdaki örnek, bu komutun bazı örnek çıkışlarını göstermektedir:

```
C:\Program Files\IBM\MQ_1\java\lib>java -jar com.ibm.mq.allclient.jar
Name:       Java Message Service Client
Version:    9.3.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       IBM MQ classes for Java Message Service
Version:    9.3.0.0
```

```
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       IBM MQ JMS Provider
Version:    9.3.0.0
Level:      p000-L140428.1 mqjbnd=p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar

Name:       Common Services for Java Platform, Standard Edition
Version:    9.3.0.0
Level:      p000-L140428.1
Build Type: Production
Location:   file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```

jackson-annotations.jar, jackson-core.jar ve jackson-databind.jar

V 9.3.3

IBM MQ classes for Java uygulamanız bir kuyruk yöneticisine güvenli TLS bağlantıları oluşturursa üç Jackson JAR dosyası gerekir.

IBM MQ classes for Java için kuruluş dizinleri
IBM MQ classes for Java dosyaları ve örnekleri, platforma göre farklı konumlara kurulur. IBM MQ ile kurulan Java Runtime Environment (JRE) olanağının yeri de altyapıya göre değişir.

IBM MQ classes for Java dosyaları için kuruluş dizinleri

Çizelge 49 sayfa 342 , IBM MQ classes for Java dosyalarının nereye kurulduğunu gösterir.

Çizelge 49. IBM MQ classes for Java kuruluş dizinleri

Hizmet olarak sunulan	Dizin
AIX AIX	*MQ_INSTALLATION_PATH*/java/lib
	/QIBM/ProdData/mqm/java/lib
Linux Linux	*MQ_INSTALLATION_PATH*/java/lib
Windows Windows	*MQ_INSTALLATION_PATH*\java\lib
z/OS z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java /lib

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Örnekler için kuruluş dizinleri

Kuruluş Doğrulama Programları (IVPs) gibi bazı örnek uygulamalar IBM MQile birlikte sağlanır. Çizelge 50 sayfa 342 , örnek uygulamaların nereye kurulduğunu gösterir. IBM MQ classes for Java örnekleri, `wmqjavaadlı` bir alt dizinde bulunur. PCF örnekleri, `pcfadlı` bir alt dizinde bulunur.

Çizelge 50. Örnek dizinleri

Hizmet olarak sunulan	Dizin
AIX AIX	*MQ_INSTALLATION_PATH*/samp/wmqjava/
IBM i IBM i	/QIBM/ProdData/mqm/java/samples
Linux Linux	*MQ_INSTALLATION_PATH*/samp/wmqjava/

Çizelge 50. Örnek dizinleri (devamı var)	
Hizmet olarak sunulan	**Dizin**
Windows	*MQ_INSTALLATION_PATH*\tools\wmqjava\
z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/samples

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

JRE için kuruluş dizinleri

IBM MQ classes for JMS için bir Java 7 (ya da üstü) Java Runtime Environment (JRE) gereklidir. IBM MQile uygun bir JRE kurulur. Çizelge 51 sayfa 343 , bu JRE ' nin nereye kurulduğunu gösterir. Sağlanan örnekler gibi Java programlarını çalıştırmak için, bu JRE 'yi kullanarak *JRE_LOCATION*`/bin/java` ' u belirtik olarak çağırın ya da altyapınıza ilişkin PATH ortamına (ya da eşdeğerine) *JRE_LOCATION*`/bin` ekleyin; burada *JRE_LOCATION* , Çizelge 51 sayfa 343içinde verilen dizindir.

Çizelge 51. JRE dizinleri	
Hizmet olarak sunulan	**Dizin**
AIX	*MQ_INSTALLATION_PATH*/java/jre
IBM i	/QIBM/ProdData/mqm/java/jre
Linux	*MQ_INSTALLATION_PATH*/java/jre
Windows	*MQ_INSTALLATION_PATH*\java\jre
z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/jre

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

IBM MQ classes for Java ile ilgili ortam değişkenleri
IBM MQ classes for Java uygulamalarını çalıştırmak istiyorsanız, sınıf yollarının IBM MQ classes for Java dizinlerini ve örnek dizinlerini içermesi gerekir.

IBM MQ classes for Java uygulamalarının çalışması için sınıf yollarının uygun IBM MQ classes for Java dizinini içermesi gerekir. Örnek uygulamaları çalıştırmak için, sınıf yolu uygun örnek dizinlerini de içermelidir. Bu bilgiler, Java çağırma komutunda ya da **CLASSPATH** ortam değişkeninde sağlanabilir.

Önemli: Java seçenek `-Xbootclasspath` ' in IBM MQ classes for Java öğesini içerecek şekilde ayarlanması desteklenmez.

Çizelge 52 sayfa 343 , örnek uygulamalar da içinde olmak üzere IBM MQ classes for Java uygulamalarını çalıştırmak için her altyapıda kullanılacak uygun **CLASSPATH** ayarını gösterir.

*Çizelge 52. IBM MQ classes for Java örnek uygulamaları da içinde olmak üzere IBM MQ classes for Java uygulamalarını çalıştırmak için **CLASSPATH** ayarı*	
Hizmet olarak sunulan	**CLASSPATH ayar**
AIX	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar: /QIBM/ProdData/mqm/java/samples/wmqjava/samples:

*Çizelge 52. IBM MQ classes for Java örnek uygulamaları da içinde olmak üzere IBM MQ classes for Java uygulamalarını çalıştırmak için **CLASSPATH** ayarı (devamı var)*	
Hizmet olarak sunulan	**CLASSPATH ayar**
Linux	CLASSPATH= *MQ_INSTALLATION_PATH*/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/samp/wmqjava/samples:
Windows	CLASSPATH= *MQ_INSTALLATION_PATH*\Java\lib\com.ibm.mq.jar; *MQ_INSTALLATION_PATH*\tools\wmqjava\samples;
z/OS	CLASSPATH= *MQ_INSTALLATION_PATH*/mqm/V9R3M0/java/lib/com.ibm.mq.jar: *MQ_INSTALLATION_PATH*/mqm/V9R3M0/java/samples/wmqjava: *MQ_INSTALLATION_PATH*/mqm/V9R3M0/java/samples/pcf

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Derlemek için `-Xlint` seçeneğini kullanırsanız, `com.ibm.mq.ese.jar` ' in var olmadığını bildiren bir ileti görebilirsiniz. Uyarıyı yoksayabilirsiniz. Bu dosya yalnızca Advanced Message Securitykuruluysa bulunur.

IBM MQ classes for JMS ile sağlanan komut dosyaları aşağıdaki ortam değişkenlerini kullanır:

MQ_JAVA_DATA_PATH
Bu ortam değişkeni, günlük ve izleme çıkışına ilişkin dizini belirtir.

MQ_JAVA_KURULUŞ_YOLU
Bu ortam değişkeni, IBM MQ classes for Java kuruluş dizinlerindegösterildiği gibi IBM MQ classes for Java ' in kurulu olduğu dizini belirtir.

MQ_JAVA_LIB_YOLU
Bu ortam değişkeni, Her platform için IBM MQ classes for Java kitaplıklarının yeriiçinde gösterildiği gibi, IBM MQ classes for Java kitaplıklarının saklandığı dizini belirtir. IVTRun gibi IBM MQ classes for Javaile verilen bazı komut dosyaları bu ortam değişkenini kullanır.

Windows Windows' da, kuruluş sırasında tüm ortam değişkenleri otomatik olarak ayarlanır.

Linux AIX AIX and Linuxişletim sistemlerinde, ortam değişkenlerini ayarlamak için `setjmsenv` (32 bit JVM kullanıyorsanız) ya da `setjmsenv64` (64 bit JVM kullanıyorsanız) komut dosyasını kullanabilirsiniz. Bu komut dosyaları *MQ_INSTALLATION_PATH*/java/bin dizininde bulunur.

IBM i IBM isistemlerinde **QIBM_MULTI_THREADED** ortam değişkeni Yolarak ayarlanmalıdır. Birden çok iş parçacıklı uygulamaları, tek iş parçacıklı uygulamaları çalıştırdığınız şekilde çalıştırabilirsiniz. Daha fazla bilgi için bkz. IBM MQ ürününü Java ve JMS ile ayarlama.

IBM MQ classes for Java bir Java 7 Java Runtime Environment (JRE) gerektirir. IBM MQile kurulan uygun bir JRE ' nin yeriyle ilgili bilgi için bkz. “IBM MQ classes for Java için kuruluş dizinleri” sayfa 342.

IBM MQ classes for Java Kitaplıklar
IBM MQ classes for Java kitaplıklarının konumu platforma göre değişir. Bir uygulamayı başlattığınızda bu konumu belirtin.

Java Native Interface (JNI) kitaplıklarının yerini belirtmek için, aşağıdaki biçimde bir **java** komutu kullanarak uygulamanızı başlatın:

```
java -Djava.library.path= library_path application_name
```

Burada *kitaplık_yolu* , JNI kitaplıklarını içeren IBM MQ classes for Javayolunu gösterir. Çizelge 53 sayfa 345 içinde her platform için IBM MQ classes for Java kitaplıklarının yeri gösterilmektedir. Bu çizelgede *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Çizelge 53. Her platform için IBM MQ classes for Java kitaplıklarının konumu	
Hizmet olarak sunulan	**IBM MQ classes for Java kitaplıklarını içeren dizin**
AIX	*MQ_INSTALLATION_PATH*/java/lib (32 bit kitaplık) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitlik kitaplıklar)
Linux (x86 platformu)	*MQ_INSTALLATION_PATH*/java/lib
Linux (POWER, x86-64 ve zSeries s390x platformları)	*MQ_INSTALLATION_PATH*/java/lib (32 bit kitaplık) *MQ_INSTALLATION_PATH*/java/lib64 (64 bitlik kitaplıklar)
Windows	*MQ_INSTALLATION_PATH*\Java\lib (32 bitlik kitaplıklar) *MQ_INSTALLATION_PATH*\Java\lib64 (64 bitlik kitaplıklar)
z/OS	*MQ_INSTALLATION_PATH*/mqm/V8R0M0/java/lib (32 bit ve 64 bit kitaplıklar)

Not:

1. Linux AIX AIX ya da Linux (Power platformu) üzerinde 32 bitlik kitaplıkları ya da 64 bitlik kitaplıkları kullanın. 64 bitlik kitaplıkları, uygulamanızı 64 bitlik bir Java sanal makinesinde (JVM) çalıştırıyorsanız kullanın. Tersi durumda, 32 bitlik kitaplıkları kullanın.
2. Windows Windowssistemlerinde, **java** komutunda konumlarını belirtmek yerine IBM MQ classes for Java kitaplıklarının yerini belirtmek için PATH ortam değişkenini kullanabilirsiniz.
3. IBM i IBM MQ classes for Java komutunu IBM iüzerinde bağ tanımlama kipinde kullanmak için, QMQMJAVA kitaplığının kitaplık listenizde olduğundan emin olun.
4. z/OS z/OSişletim sistemlerinde, 32 bit ya da 64 bit Java sanal makinesi (JVM) kullanabilirsiniz. Hangi kitaplıkların kullanılacağını belirtmeniz gerekmez; IBM MQ classes for Java hangi JNI kitaplıklarının yükleneceğini kendisi belirleyebilir.

İlgili kavramlar

kullanmaIBM MQ classes for Java

IBM MQ classes for Javaürününü kurduktan sonra, kuruluşunuzu kendi uygulamalarınızı çalıştıracak şekilde yapılandırabilirsiniz.

IBM MQ classes for Java ile OSGi desteği

OSGi, uygulamaların kod paketleri olarak konuşlandırılmasını destekleyen bir çerçeve sağlar. IBM MQ classes for Javaürününün bir parçası olarak üç OSGi paketi sağlanır.

OSGi, kod paketleri biçiminde gelen uygulamaların devreye alınmasını destekleyen genel amaçlı, güvenli ve yönetilen bir Java çerçevesi sağlar. OSGi uyumlu aygıtlar, kod paketlerini yükleyip kurabilir ve artık gerekli olmadıklarında bunları kaldırabilir. Çerçeve, kod paketlerinin dinamik ve ölçeklenebilir bir şekilde kurulumunu ve güncellenmesini yönetir.

IBM MQ classes for Java aşağıdaki OSGi paketlerini içerir.

com.ibm.mq.osgi.java_*version_number*.jar
Uygulamaların IBM MQ classes for Java' yi kullanmasına izin veren JAR dosyaları.

com.ibm.mq.jakarta.osgi.allclient_*version_number*.jar

V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging 3.0için bu JAR dosyası, uygulamaların IBM MQ classes for JMS ve IBM MQ classes for Javadosyalarını kullanmasını sağlar ve PCF iletilerini işlemek için kodu da içerir.

com.ibm.mq.osgi.allclient_*version_number*.jar

JMS 2.0 JMS 2.0için bu JAR dosyası, uygulamaların hem IBM MQ classes for JMS hem de IBM MQ classes for Javakullanmasına olanak sağlar ve PCF iletilerini işlemek için kodu da içerir.

com.ibm.mq.jakarta.osgi.allclientprereqs_*version_number*.jar

V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging 3.0için, bu JAR dosyası `com.ibm.mq.jakarta.osgi.allclient_version_number.jar`ile ilgili önkoşulları sağlar.

com.ibm.mq.osgi.allclientprereqs_*version_number*.jar

JMS 2.0 JMS 2.0için, bu JAR dosyası `com.ibm.mq.osgi.allclient_version_number.jar`ile ilgili önkoşulları sağlar.

Burada *version_number* , kurulu IBM MQ sürüm numarasıdır.

Paketler, IBM MQ kurulumunuzun `java/lib/OSGi` alt dizinine ya da Windowsüzerindeki `java\lib\OSGi` klasörüne kurulur.

IBM MQ 8.0olanağından, yeni uygulamalar için `com.ibm.mq.osgi.allclient_8.0.0.0.jar`ve `com.ibm.mq.osgi.allclientprereqs_8.0.0.0.jar` paketlerini kullanın. Bu kod paketlerinin kullanılması, aynı OSGi çerçevesi içinde hem IBM MQ classes for JMS hem de IBM MQ classes for Java çalıştırılamamasına ilişkin kısıtlamayı kaldırır. Ancak diğer tüm kısıtlamalar hala geçerlidir. IBM MQ 8.0öncesi IBM MQ sürümleri için, IBM MQ classes for JMS ya da IBM MQ classes for Java kullanımı kısıtlaması geçerlidir.

Diğer dokuz paket, IBM MQ kurulumunuzun `java/lib/OSGi` alt dizinine ya da Windowsüzerindeki `java\lib\OSGi` klasörüne de kurulur. Bu paketler IBM MQ classes for JMS' in bir parçasıdır ve IBM MQ classes for Java paketi yüklü olan bir OSGi çalıştırma ortamına yüklenmemelidir. IBM MQ classes for Java OSGi kod paketi, IBM MQ classes for JMS kod paketlerinin de yüklendiği bir OSGi yürütme ortamına yüklenirse, IBM MQ classes for Java kod paketini ya da IBM MQ classes for JMS kod paketlerini kullanan uygulamalar çalıştırıldığında aşağıdaki örnekte gösterildiği gibi hatalar oluşur:

```
java.lang.ClassCastException: com.ibm.mq.MQException incompatible with com.ibm.mq.MQException
```

IBM MQ classes for Java için OSGi paketi OSGi Yayın Düzeyi 4 belirtimine yazıldı; OSGi Yayın Düzeyi 3 ortamında çalışmıyor.

OSGi yürütme ortamının gereken DLL dosyalarını ya da paylaşılan kitaplıkları bulabilmesi için sistem yolunuzu ya da kitaplık yolunu doğru olarak ayarlamanız gerekir.

IBM MQ classes for Javaiçin OSGi kod paketini kullanırsanız, OSGi gibi çok sınıflı bir yükleyici ortamında sınıfların yüklenmesinde oluşan bir sorun nedeniyle Java içine yazılan kanal çıkış sınıfları desteklenmez. Bir kullanıcı paketi IBM MQ classes for Java paketini bilebilir, ancak IBM MQ classes for Java paketi herhangi bir kullanıcı paketini bilmez. Sonuç olarak, IBM MQ classes for Java kod paketinde kullanılan sınıf yükleyici, kullanıcı kod paketinde bulunan bir kanal çıkış sınıfını yükleyemez.

OSGi hakkında daha fazla bilgi için OSGi ittifakı web sitesine bakın.

z/OS *IBM MQ classes for Java ürününün z/OS üzerine kurulması*

z/OSüzerinde, yürütme sırasında kullanılan STEPLIB, IBM MQ SCSQAUTH ve SCSQANLE kitaplıklarını içermelidir.

z/OS UNIX System Services' den, aşağıdaki örnekte gösterildiği gibi `.profile` ürününüzdeki bir satırı kullanarak bu kitaplıkları ekleyebilirsiniz; `thlqual` yerine, IBM MQürününü kurarken seçtiğiniz üst düzey veri kümesi niteleyicisini kullanabilirsiniz:

```
export STEPLIB=thlqual.SCSQAUTH:thlqual.SCSQANLE:$STEPLIB
```

Diğer ortamlarda genellikle, STEPLIB birleşiminde SCSQAUTH 'yi içerecek şekilde başlatma JCL' sini düzenlemeniz gerekir:

```
 STEPLIB DD DSN=thlqual.SCSQAUTH,DISP=SHR
         DD DSN=thlqual.SCSQANLE,DISP=SHR
```

IBM MQ classes for Java yapılandırma dosyası
IBM MQ classes for Java yapılandırma dosyası, IBM MQ classes for Java' yi yapılandırmak için kullanılan özellikleri belirtir.

IBM MQ classes for Java yapılandırma dosyasının biçimi, standart bir Java özellikler dosyasının biçimidir.

Örnek bir yapılandırma dosyası (`mqjava.config`), IBM MQ classes for Java kuruluş dizininin `bin` alt dizininde bulunur. Bu dosya, desteklenen tüm özellikleri ve varsayılan değerlerini belgeler.

Not: IBM MQ kuruluşu ilerideki bir düzeltme paketine yükseltildiğinde örnek yapılanış kütüğünün üzerine yazılır. Bu nedenle, örnek yapılandırma dosyasının bir kopyasını uygulamalarınızla kullanmak üzere oluşturmanız önerilir.

Bir IBM MQ classes for Java yapılandırma dosyasının adını ve konumunu seçebilirsiniz. Uygulamanızı başlattığınızda, aşağıdaki biçimi kullanan bir **java** komutu kullanın:

```
java -Dcom.ibm.msg.client.config.location=config_file_url application_name
```

Komutta *config_file_url* , IBM MQ classes for Java yapılandırma dosyasının adını ve konumunu belirten tek tip bir kaynak konum belirleyicidir (URL). Şu tiplerin URL ' leri desteklenir: `http`, `file`, `ftp`ve `jar`.

Aşağıdaki örnekte bir **java** komutu gösterilmektedir:

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/mqjava.config MyAppClass
```

Bu komut, IBM MQ classes for Java yapılanış kütüğünü yerel Windows sisteminde `D:\mydir\mqjava.config` kütüğü olarak tanıtır.

Bir uygulama başlatıldığında, IBM MQ classes for Java yapılandırma dosyasının içeriğini okur ve belirtilen özellikleri bir iç özellik deposunda saklar. **java** komutu bir yapılandırma dosyasını belirtmezse ya da yapılandırma dosyası bulunamazsa, IBM MQ classes for Java tüm özellikler için varsayılan değerleri kullanır. Gerekirse, **java** komutunda bir sistem özelliği olarak belirterek yapılandırma dosyasındaki herhangi bir özelliği geçersiz kılabilirsiniz.

IBM MQ classes for Java yapılanış dosyası, bir uygulama ile bir kuyruk yöneticisi ya da aracı arasında desteklenen iletişimle birlikte kullanılabilir.

IBM MQ MQI client yapılandırma dosyasında belirtilen özelliklerin geçersiz kılınması

IBM MQ MQI client yapılanış kütüğü, IBM MQ classes for Java' yi yapılandırmak için kullanılan özellikleri de belirtebilir. Ancak, IBM MQ MQI client yapılanış dosyasında belirtilen özellikler yalnızca, istemci kipinde bir uygulama kuyruk yöneticisine bağlandığında geçerlidir.

Gerekirse, bir IBM MQ classes for Java yapılandırma dosyasında özellik olarak belirterek IBM MQ MQI client yapılandırma dosyasındaki herhangi bir özniteliği geçersiz kılabilirsiniz. IBM MQ MQI client yapılandırma dosyasındaki bir özniteliği geçersiz kılmak için, IBM MQ classes for Java yapılandırma dosyasında aşağıdaki biçime sahip bir giriş kullanın:

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Girdideki değişkenler aşağıdaki anlamlara sahiptir:

kıta
Özniteliği içeren IBM MQ MQI client yapılandırma dosyasındaki dörtlünün adı.

propName
IBM MQ MQI client yapılandırma dosyasında belirtilen özniteliğin adı.

propValue
IBM MQ MQI client yapılandırma dosyasında belirtilen özniteliğin değerini geçersiz kılan özelliğin değeri.

Alternatif olarak, özelliği **java** komutunda sistem özelliği olarak belirterek IBM MQ MQI client yapılandırma dosyasındaki bir özniteliği geçersiz kılabilirsiniz. Özelliği bir sistem özelliği olarak belirtmek için önceki biçimi kullanın.

Bir IBM MQ MQI client yapılandırma dosyasında yalnızca aşağıdaki öznitelikler IBM MQ classes for Javaile ilgilidir. Diğer öznitelikleri belirtirseniz ya da geçersiz kılarsanız, bunun bir etkisi olmaz. Özellikle, İstemci yapılandırma dosyasının KANAL kısmı içindeki `ChannelDefinitionFile` ve `ChannelDefinitionDirectory` ' in kullanılmadığına dikkat edin. CCDT ' nin IBM MQ classes for Javaile nasıl kullanılacağına ilişkin ayrıntılar için bkz. “IBM MQ classes for Java ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 362 .

Çizelge 54. İstemci yapılanış dosyasının hangi kısmı hangi özniteliği içeriyor?

Kıta	**Öznitelik**
İstemci yapılanış kütüğünün KANAL kısmı	Put1DefaultAlwaysSync
İstemci yapılanış kütüğünün KANAL kısmı	PasswordProtection
ClientExitİstemci yapılanış kütüğünün yol kısmı	ExitsDefaultYolu
ClientExitİstemci yapılanış kütüğünün yol kısmı	ExitsDefaultPath64
ClientExitİstemci yapılanış kütüğünün yol kısmı	JavaExitsSınıf Yolu
İstemci yapılanış kütüğünün JMQI kısmı	useMQCSPauthentication
İstemci yapılanış kütüğünün MessageBuffer kısmı	MaximumSize
İstemci yapılanış kütüğünün MessageBuffer kısmı	PurgeTime
İstemci yapılanış kütüğünün MessageBuffer kısmı	UpdatePercentage
İstemci yapılanış kütüğünün TCP kısmı	ClntRcvBuffSize
İstemci yapılanış kütüğünün TCP kısmı	ClntSndBuffSize
İstemci yapılanış kütüğünün TCP kısmı	Connect_Timeout
İstemci yapılanış kütüğünün TCP kısmı	KeepAlive

IBM MQ MQI client yapılandırmasına ilişkin daha fazla bilgi için bkz. IBM MQ MQI client yapılandırma dosyası, `mqclient.ini`.

İlgili görevler
Java uygulamaları için IBM MQ sınıflarının izlenmesi

Java izlemesini yapılandırmak için Java Standard Environment Trace (Standart Ortam İzleme) olanağının kullanılması
IBM MQ classes for Java izleme olanağını yapılandırmak için Java Standard Environment Trace Settings kısmına bakın.

com.ibm.msg.client.commonservices.trace.outputName = *traceOutputAd*
`traceOutputName` , izleme çıkışının gönderildiği dizin ve dosya adıdır.

Varsayılan olarak, izleme bilgileri uygulamanın yürürlükteki çalışma dizinindeki bir izleme dosyasına yazılır. İzleme kütüğünün adı, uygulamanın çalıştığı ortama bağlıdır:

- IBM MQ 9.1.5 ve IBM MQ 9.1.0 Fix Pack 5' den:

  - Uygulama IBM MQ classes for Java dosyasını yeniden yüklenebilir JAR dosyasından `com.ibm.mq.allclient.jar`yüklediyse, izleme `mqjavaclient_%PID%.cl%u.trc`adlı bir dosyaya yazılır.
  - Uygulama IBM MQ classes for Java dosyasını `com.ibm.mq.jar`JAR dosyasından yüklüyorsa, izleme `mqjava_%PID%.cl%u.trc`adlı bir dosyaya yazılır.
- IBM MQ 9.0.0 Fix Pack 2' dan:
  - Uygulama IBM MQ classes for Java dosyasını yeniden yüklenebilir JAR dosyasından `com.ibm.mq.allclient.jar`yüklediyse, izleme `mqjavaclient_%PID%.trc`adlı bir dosyaya yazılır.
  - Uygulama IBM MQ classes for Java dosyasını `com.ibm.mq.jar`JAR dosyasından yüklüyorsa, izleme `mqjava_%PID%.trc`adlı bir dosyaya yazılır.
- For IBM MQ classes for Java for IBM MQ 9.0.0 Fix Pack 1 or earlier, trace is written to a file called `mqjms_%PID%.trc`.

Burada *%PID%* , izlenmekte olan uygulamanın işlem tanıtıcısıdır ve *%u* , farklı Java sınıf yükleyicileri altında izleme çalıştıran iş parçacıkları arasında ayrım yapmak için benzersiz bir sayıdır.

Bir işlem tanıtıcısı yoksa, rasgele bir sayı oluşturulur ve başına `f`harfi eklenir. İşlem tanıtıcısını belirttiğiniz bir dosya adına eklemek için `%PID%`dizgisini kullanın.

Alternatif bir dizin belirtirseniz, dizin varolmalıdır ve bu dizin için yazma izniniz olmalıdır. Yazma izniniz yoksa, izleme çıkışı `System.err`' e yazılır.

com.ibm.msg.client.commonservices.trace.include = *includeList*
`includeList` , takip edilen paketlerin ve sınıfların ya da `ALL` ya da `NONE`özel değerlerinin bir listesidir.

Paket ya da sınıf adlarını noktalı virgülle (`;`) ayırın. *`includeList`* varsayılan olarak `ALL`değerine ayarlanır ve IBM MQ classes for Javaiçindeki tüm paketleri ve sınıfları izler.

Not: Bir paket ekleyebilir, ancak daha sonra bu paketin alt paketlerini dışlayabilirsiniz. Örneğin, `a.b` ve dışlama paketini `a.b.x`eklerseniz, izleme `a.b.x` ya da `a.b.x.1`değil, `a.b.y` ve `a.b.z`içindeki her şeyi içerir.

com.ibm.msg.client.commonservices.trace.exclude = *excludeList*
`excludeList` , izlenmeyen paketlerin ve sınıfların ya da özel değerlerin `ALL` ya da `NONE`listesidir.

Paket ya da sınıf adlarını noktalı virgülle (`;`) ayırın. *`excludeList`* varsayılan olarak `NONE`değerine ayarlanır ve bu nedenle, IBM MQ classes for JMS içindeki hiçbir paket ve sınıfı izlenmez.

Not: Bir paketi dışlayabilir, ancak daha sonra bu paketin alt paketlerini ekleyebilirsiniz. Örneğin, `a.b` paketini dışlar ve `a.b.x`paketini dahil ederseniz, izleme `a.b.y` ya da `a.b.z`değil, `a.b.x` ve `a.b.x.1`içindeki her şeyi içerir.

Hem içerilen hem de dışlanan olarak, aynı düzeyde belirtilen herhangi bir paket ya da sınıf içerilir.

com.ibm.msg.client.commonservices.trace.maxBytes = *maxArrayByte*
`maxArrayBytes` , herhangi bir bayt dizisinden izlenecek bayt sayısı üst sınırıdır.

`maxArrayBytes` pozitif bir tamsayıya ayarlanırsa, izleme dosyasına yazılan bayt dizisindeki bayt sayısını sınırlar. *`maxArrayBytes`* yazdıktan sonra bayt dizisini keser. *`maxArrayBytes`* ayarı, sonuçtaki izleme kütüğünün büyüklüğünü azaltır ve izlemenin uygulamanın başarımı üzerindeki etkisini azaltır.

Bu "zelli§e iliŸkin `0` de§eri, izleme kt § ne bayt dizilerinin i‡eriklerinin g" nderilmediğini g " rntler.

Varsayılan değer, izleme dosyasına gönderilen bayt dizisindeki bayt sayısı sınırını kaldıran `-1`değeridir.

com.ibm.msg.client.commonservices.trace.limit = *maxTraceByte*
`maxTraceBytes` , bir izleme çıkış dosyasına yazılan byte sayısı üst sınırıdır.

`maxTraceBytes` , *`traceCycles`*ile çalışır. Yazılan izleme baytı sayısı sınıra yakınsa, dosya kapatılır ve yeni bir izleme çıkış dosyası başlatılır.

0 değeri, izleme çıkış kütüğünün uzunluğunun sıfır olduğu anlamına gelir. Varsayılan değer -1olup bu, bir izleme çıkış dosyasına yazılacak veri miktarının sınırsız olduğu anlamına gelir.

com.ibm.msg.client.commonservices.trace.count = *traceCycles*
traceCycles , geçiş için geçilecek izleme çıkış dosyalarının sayısıdır.

Yürürlükteki izleme çıkışı dosyası *maxTraceBytes*ile belirtilen sınıra ulaşırsa, dosya kapatılır. Sonraki izleme çıkışı, sıralı olarak sonraki izleme çıkışı kütüğüne yazılır. Her izleme çıkış dosyası, dosya adının sonuna eklenen sayısal bir sonekle ayırt edilir. Yürürlükteki ya da en son izleme çıkışı kütüğü mqjms.trc.0, bir sonraki en son izleme çıkışı kütüğü mqjms.trc.1. Eski izleme dosyaları, sınıra kadar aynı numaralandırma kalıbını izler.

traceCycles varsayılan değeri 1 'dir. *traceCycles* değeri 1 ise, yürürlükteki izleme çıkış dosyası büyüklük üst sınırına ulaştığında dosya kapatılır ve silinir. Aynı adı taşıyan yeni bir izleme çıkış dosyası başlatıldı. Bu nedenle, bir kerede tek bir izleme çıkış dosyası vardır.

com.ibm.msg.client.commonservices.trace.parameter = *traceParameters*
traceParameters , yöntem değiştirgelerinin ve dönüş değerlerinin izleme kapsamına alınıp alınmayacağını denetler.

traceParameters , varsayılan olarak TRUEdeğerine ayarlanır. *traceParameters* FALSEolarak ayarlanırsa, yalnızca yöntem imzaları izlenir.

com.ibm.msg.client.commonservices.trace.startup = *başlatma*
Kaynakların ayrılması sırasında IBM MQ classes for Java ' un kullanıma hazırlama aşaması vardır. Ana izleme olanağı, kaynak ayırma aşamasında kullanıma hazırlandı.

startup TRUEolarak ayarlanırsa, başlatma izlemesi kullanılır. İzleme bilgileri hemen üretilir ve izleme olanağının kendisi de içinde olmak üzere tüm bileşenlerin kurulumunu içerir. Yapılandırma sorunlarını tanılamak için başlatma izleme bilgileri kullanılabilir. Başlatma izleme bilgileri her zaman System.err' e yazılır.

startup , varsayılan olarak FALSEdeğerine ayarlanır.

Kullanıma hazırlama tamamlanmadan önce *startup* işaretlenir. Bu nedenle, yalnızca komut satırında özelliği Java sistem özelliği olarak belirtin. Bunu IBM MQ classes for Java yapılandırma dosyasında belirtmeyin.

com.ibm.msg.client.commonservices.trace.compress = *compressedTrace*
İzleme çıkışını sıkıştırmak için *compressedTrace* değerini TRUE olarak ayarlayın.

compressedTrace varsayılan değeri FALSEdeğeridir.

compressedTrace TRUEolarak ayarlanırsa, izleme çıkışı sıkıştırılır. Varsayılan izleme çıkış dosyası adı .trzuzantısına sahiptir. Sıkıştırma FALSEolarak ayarlanırsa, varsayılan değer, dosyanın sıkıştırılmadığını belirtmek için .trc uzantısına sahiptir. Ancak, *traceOutputName* dosyasında izleme çıkışına ilişkin dosya adı belirtildiyse, bu ad kullanılır; dosyaya sonek uygulanmaz.

Sıkıştırılmış izleme çıkışı sıkıştırılmamış değerinden küçük. Daha az G/Ç olduğundan, sıkıştırılmamış izlemeden daha hızlı yazılabilir. Sıkıştırılmış izleme, IBM MQ classes for Java performansı üzerinde sıkıştırılmamış izlemeden daha az etkiye sahiptir.

maxTraceBytes ve *traceCycles* ayarlanırsa, birden çok düz dosya yerine birden çok sıkıştırılmış izleme dosyası yaratılır.

IBM MQ classes for Java denetimsiz bir şekilde sona ererse, sıkıştırılmış bir izleme dosyası geçerli olmayabilir. Bu nedenle, izleme sıkıştırması yalnızca IBM MQ classes for Java denetimli bir şekilde kapandığında kullanılmalıdır. Yalnızca incelenmekte olan sorunlar JVM ' nin beklenmedik bir şekilde durmasına neden olmazsa izleme sıkıştırmasını kullanın. System.Halt() ' in kapanmasına ya da olağandışı, denetimsiz JVM sonlandırmasına neden olabilecek sorunları tanılarken izleme sıkıştırmasını kullanmayın.

com.ibm.msg.client.commonservices.trace.level = *traceLevel*
traceLevel , izleme için bir süzgeç düzeyi belirtir. Tanımlanan izleme düzeyleri aşağıdaki gibidir:

- TRACE_NONE: 0

- TRACE_EXCEPTION: 1
- TRACE_WARNING: 3
- TRACE_INFO: 6
- TRACE_ENTRYEXIT: 8
- TRACE_DATA: 9
- TRACE_ALL: Integer.MAX_VALUE

Her izleme düzeyi tüm alt düzeyleri içerir. Örneğin, izleme düzeyi TRACE_INFOolarak ayarlanırsa, tanımlı düzeyi TRACE_EXCEPTION, TRACE_WARNINGya da TRACE_INFO olan herhangi bir izleme noktası izlemeye yazılır. Diğer tüm izleme noktaları dışlanır.

com.ibm.msg.client.commonservices.trace.standalone = *standaloneTrace*

standaloneTrace , IBM MQ classes for Java istemci izleme hizmetinin bir WebSphere Application Server ortamında kullanılıp kullanılmayacağını denetler.

standaloneTrace TRUEolarak ayarlanırsa, izleme yapılanışını saptamak için IBM MQ classes for Java istemcisi izleme özellikleri kullanılır.

standaloneTrace FALSEolarak ayarlanırsa ve IBM MQ classes for Java istemcisi bir WebSphere Application Server kapsayıcısında çalışıyorsa, WebSphere Application Server izleme hizmeti kullanılır. Oluşturulan izleme bilgileri, uygulama sunucusunun izleme ayarlarına bağlıdır.

standaloneTrace varsayılan değeri FALSEdeğeridir.

IBM MQ classes for Java ve yazılım yönetimi araçları
Apache Maven gibi yazılım yönetimi araçları IBM MQ classes for Javaile kullanılabilir.

Birçok büyük geliştirme kuruluşu, üçüncü kişi kitaplıklarının havuzlarını merkezi olarak yönetmek için bu araçları kullanır.

IBM MQ classes for Java , bir dizi JAR dosyasından oluşur. Java dil uygulamalarını bu API ' yi kullanarak geliştirirken, uygulamanın geliştirilmekte olduğu makinede bir IBM MQ Server, IBM MQ Client ya da IBM MQ Client SupportPac kuruluşu gerekir.

Bir yazılım yönetimi aracı kullanmak ve IBM MQ classes for Java dosyasını oluşturan JAR dosyalarını merkezi olarak yönetilen bir havuza eklemek istiyorsanız, aşağıdaki noktalara dikkat edilmelidir:

- Bir havuz ya da kapsayıcı yalnızca kuruluşunuz içindeki geliştiriciler tarafından kullanılabilir kılınmalıdır. Kuruluş dışında dağıtıma izin verilmez.
- Havuzun, tek bir IBM MQ yayın düzeyindeki ya da düzeltme paketindeki JAR dosyalarının eksiksiz ve tutarlı bir kümesini içermesi gerekir.
- Havuzu, IBM Destek tarafından sağlanan herhangi bir bakımla güncellemekten siz sorumlu olursunuz.

IBM MQ 8.0içinden, com.ibm.mq.allclient.jar JAR dosyasının havuza kurulması gerekir.

IBM MQ 9.0' den Bouncy Castle güvenlik sağlayıcısı ve CMS destek JAR dosyaları gereklidir. Daha fazla bilgi için bkz. “IBM MQ classes for Java yeniden dizinlenebilir JAR dosyaları” sayfa 340 ve IBM dışı JRE ' ler için destek.

IBM MQ classes for Java uygulamaları için kuruluş sonrası kuruluş

IBM MQ classes for Javaürününü kurduktan sonra, kuruluşunuzu kendi uygulamalarınızı çalıştıracak şekilde yapılandırabilirsiniz.

En son bilgiler için ya da ortamınızla ilgili daha özel bilgiler için IBM MQ ürün readme (benioku) dosyasına bakmayı unutmayın. Ürün benioku dosyasının en son sürümü IBM MQ, WebSphere MQve MQSeries ürün benioku bilgileri web sayfasında bulunur.

Bir IBM MQ classes for Java uygulamasını bağ tanımları kipinde çalıştırmayı denemeden önce, IBM MQ uygulamasını Yapılandırmakonusunda açıklandığı gibi yapılandırdığınızdan emin olun.

Kuyruk yöneticinizin IBM MQ classes for Java ' den gelen istemci bağlantılarını kabul edecek şekilde yapılandırılması
Kuyruk yöneticinizi istemcilerden gelen bağlantı isteklerini kabul edecek şekilde yapılandırmak için, bir sunucu bağlantı kanalı tanımlayın ve kullanılmasına izin verin ve bir dinleyici programı başlatın.

Ayrıntılar için bkz. “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023.

IBM MQ classes for Java uygulamalarının Java security manager altında çalıştırılması
IBM MQ classes for Java , Java security manager etkinken çalışabilir. Java security manager etkin durumdayken uygulamaları başarıyla çalıştırmak için, Java Virtual Machine (JVM) ürününüzü uygun bir ilke tanımlama dosyasıyla yapılandırmanız gerekir.

Uygun bir ilke tanımlama dosyası yaratmanın en basit yolu, Java runtime environment (JRE) ile sağlanan ilke dosyasını değiştirmektir. Çoğu sistemde bu dosya, JRE dizininizle göreli olarak `path lib/security/java.policy`dizininde saklanır. Tercih ettiğiniz düzenleyiciyi kullanarak ya da JRE ile verilen policytool programını kullanarak ilke dosyalarını düzenleyebilirsiniz.

Aşağıdaki işlemleri yapabilmesi için `com.ibm.mq.jmqi.jar` dosyası için yetki vermeniz gerekir:

- Yuva yarat (istemci kipinde)
- Yerel kitaplığı yükle (bağ tanımlama kipinde)
- Ortamdan çeşitli özellikleri okuma

`os.name` sistem özelliği, Java security manageraltında çalışırken IBM MQ classes for Java tarafından kullanılabilir olmalıdır.

Java uygulamanız Java security manageruygulamasını kullanıyorsa, uygulama tarafından kullanılan `java.security.policy` dosyasına aşağıdaki izni eklemeniz gerekir; tersi durumda, özel durumlar uygulamaya yayınlanır:

```
permission java.lang.RuntimePermission "modifyThread";
```

Bu RuntimePermission , istemcinin, kuyruk yöneticilerine TCP/IP bağlantıları üzerinden çok yollu etkileşimlerin atanmasının ve kapanmasının bir parçası olarak gerekli.

Örnek ilke dosyası girişi

Burada, IBM MQ classes for Java ' in varsayılan güvenlik yöneticisi altında başarıyla çalışmasına izin veren bir ilke dosyası girişi örneği verilmiştir. Bu örnekteki *MQ_INSTALLATION_PATH* dizgisini, IBM MQ classes for Java ' un sisteminizde kurulu olduğu yerle değiştirin.

```
grant codeBase "file: MQ_INSTALLATION_PATH/java/lib/*" {
//We need access to these properties, mainly for tracing
permission java.util.PropertyPermission "user.name","read";
permission java.util.PropertyPermission "os.name","read";
permission java.util.PropertyPermission "user.dir","read";
permission java.util.PropertyPermission "line.separator","read";
permission java.util.PropertyPermission "path.separator","read";
permission java.util.PropertyPermission "file.separator","read";
permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.log.*","read";
permission java.util.PropertyPermission "com.ibm.msg.client.commonservices.trace.*","read";
permission java.util.PropertyPermission "Diagnostics.Java.Errors.Destination.Filename","read";
permission java.util.PropertyPermission "com.ibm.mq.commonservices","read";
permission java.util.PropertyPermission "com.ibm.mq.cfg.*","read";

//Tracing - we need the ability to control java.util.logging
permission java.util.logging.LoggingPermission "control";
// And access to create the trace file and read the log file - assumed to be in the current
directory
permission java.io.FilePermission "*","read,write";

// Required to allow a trace file to be written to the filesystem.
// Replace 'TRACE_FILE_DIRECTORY' with the directory name where trace is to be written to
permission java.io.FilePermission "TRACE_FILE_DIRECTORY","read,write";
permission java.io.FilePermission "TRACE_FILE_DIRECTORY/*","read,write";

// We'd like to set up an mBean to control trace
```

```
permission javax.management.MBeanServerPermission "createMBeanServer";
permission javax.management.MBeanPermission "*","*";

// We need to be able to read manifests etc from the jar files in the installation directory
permission java.io.FilePermission "MQ_INSTALLATION_PATH/java/lib/-","read";

//Required if mqclient.ini/mqs.ini configuration files are used
permission java.io.FilePermission "MQ_DATA_DIRECTORY/mqclient.ini","read";
permission java.io.FilePermission "MQ_DATA_DIRECTORY/mqs.ini","read";

//For the client transport type.
permission java.net.SocketPermission "*","connect,resolve";

//For the bindings transport type.
permission java.lang.RuntimePermission "loadLibrary.*";

//For applications that use CCDT tables (access to the CCDT AMQCLCHL.TAB)
permission java.io.FilePermission "MQ_DATA_DIRECTORY/qmgrs/QM_NAME/@ipcc/AMQCLCHL.TAB","read";

//For applications that use User Exits
permission java.io.FilePermission "MQ_DATA_DIRECTORY/exits/*","read";
permission java.io.FilePermission "MQ_DATA_DIRECTORY/exits64/*","read";
permission java.lang.RuntimePermission "createClassLoader";

//Required for the z/OS platform
permission java.util.PropertyPermission "com.ibm.vm.bitmode","read";

// Used by the internal ConnectionFactory implementation
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";

// Used for controlled class loading
permission java.lang.RuntimePermission "setContextClassLoader";

// Used to default the Application name in Client mode connections
permission java.util.PropertyPermission "sun.java.command","read";

// Used by the IBM JSSE classes
permission java.util.PropertyPermission "com.ibm.crypto.provider.AESNITrace","read";

//Required to determine if an IBM Java Runtime is running in FIPS mode,
//and to modify the property values status as required.
permission java.util.PropertyPermission "com.ibm.jsse2.usefipsprovider","read,write";
permission java.util.PropertyPermission "com.ibm.jsse2.JSSEFIPS","read,write";
//Required if an IBM FIPS provider is to be used for SSL communication.
permission java.security.SecurityPermission "insertProvider.IBMJCEFIPS";

// Required for non-IBM Java Runtimes that establish secure client
// transport mode connections using mutual TLS authentication
permission java.util.PropertyPermission "javax.net.ssl.keyStore","read";
permission java.util.PropertyPermission "javax.net.ssl.keyStorePassword","read";

// Required for Java applications that use the Java Security Manager
permission java.lang.RuntimePermission "modifyThread";
};
```

Bu ilke dosyası örneği, IBM MQ classes for Java ' in güvenlik yöneticisi altında düzgün çalışmasını sağlar, ancak uygulamalarınız çalışmadan önce kendi kodunuzun düzgün çalışmasını sağlamanız gerekebilir.

IBM MQ classes for Java ile verilen örnek kod özellikle güvenlik yöneticisiyle kullanılmak üzere etkinleştirilmedi; ancak, IVT sınamaları bu ilke dosyasıyla ve varsayılan güvenlik yöneticisiyle çalışır.

Önemli:

IBM MQ classes for Java izleme olanağı, ek sistem özelliklerini sorgulama ve diğer dosya sistemi işlemlerini gerçekleştirirken ek izinler gerektirir.

İzlemenin etkinleştirildiği bir güvenlik yöneticisi altında çalışmaya uygun bir şablon güvenlik ilkesi dosyası, IBM MQ kuruluşunun `samples/wmqjava` dizininde `example.security.policy`olarak sağlanır.

Varsayılan kuruluş için `example.security.policy` kütüğünün yeri şöyledir:

Windows
İçinde `C:\Program Files\IBM\MQ\Tools\wmqjava\samples\example.security.policy`

Linux
İçinde `/opt/mqm/samp/wmqjava/samples/example.security.policy`

Solaris
İçinde `/opt/mqm/samp/wmqjava/samples/example.security.policy`

AIX
İçinde `/usr/mqm/samp/wmqjava/samples/example.security.policy`

IBM MQ classes for Java uygulamalarının CICS Transaction Server altında çalıştırılması
Bir IBM MQ classes for Java uygulaması, CICS Transaction Server altında bir işlem olarak çalıştırılabilir.

Bir IBM MQ classes for Java uygulamasını CICS Transaction Server for z/OSaltında hareket olarak çalıştırmak için aşağıdaki adımları gerçekleştirin:

1. Sağlanan CEDA hareketini kullanarak uygulamayı ve hareketi CICS olarak tanımlayın.
2. IBM MQ CICS bağdaştırıcısının CICS sisteminizde kurulu olduğundan emin olun. z/OS (Ayrıntılar için bkz. IBM MQ ' yi CICS ile kullanma .)
3. CICS içinde belirtilen JVM ortamının uygun CLASSPATH ve LIBPATH girişlerini içerdiğini doğrulayın.
4. Normal işlemlerinizi kullanarak işlemi başlatın.

CICS Java hareketlerinin çalıştırılmasıyla ilgili daha fazla bilgi için CICS sistem belgelerinize bakın.

IBM MQ classes for Java kuruluşunun doğrulanması

IBM MQ classes for Javaile birlikte bir kuruluş doğrulama programı (MQIVP) sağlanır. IBM MQ classes for Java' in tüm bağlantı kiplerini sınamak için bu programı kullanabilirsiniz.

Program, hangi bağlantı kipini doğrulamak istediğinizi belirlemek için çeşitli seçenekler ve diğer veriler girmenizi ister. Kuruluşunuzu doğrulamak için aşağıdaki yordamı kullanın:

1. Programı istemci kipinde çalıştıracaksanız, kuyruk yöneticinizi “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023konusunda açıklandığı gibi yapılandırın. Kullanılacak kuyruk SYSTEM.DEFAULT.LOCAL.QUEUE
2. Programı istemci kipinde çalıştıracaksanız, bkz. “kullanmaIBM MQ classes for Java” sayfa 334.

   Programı çalıştıracağınız sistemde bu yordamın geri kalan adımlarını gerçekleştirin.
3. CLASSPATH ortam değişkeninizi “IBM MQ classes for Java ile ilgili ortam değişkenleri” sayfa 343içindeki yönergelere göre güncellediğinizden emin olun.
4. Dizini *MQ_INSTALLATION_PATH*`/mqm/samp/wmqjava/samples`olarak değiştirin; burada *MQ_INSTALLATION_PATH* , IBM MQ kurulumunuzun yoludur. Daha sonra, komut isteminde şunu girin:

   ```
   java -Djava.library.path= library_path MQIVP
   ```

   Burada *kitaplık_yolu* , IBM MQ classes for Java kitaplıklarının yoludur (bkz. “IBM MQ classes for Java Kitaplıklar” sayfa 344).

   (1) işaretli komut isteminde:

   - TCP/IP bağlantısı kullanmak için bir IBM MQ sunucusu anasistem adı girin.
   - Yerel bağlantıyı (bağ tanımlama kipi) kullanmak için alanı boş bırakın (ad girmeyin).

   Program aşağıdakileri yapmayı dener:

   - 1. Kuyruk yöneticisine bağlanma
   - 2. SYSTEM.DEFAULT.LOCAL.QUEUE, kuyruğa bir ileti koyun, kuyruktan bir ileti alın ve kuyruğu kapatın
   - 3. Kuyruk yöneticisiyle bağlantıyı kes
   - 4. İşlemler başarılı olursa ileti döndür

Burada, görebileceğiniz bilgi istemlerinin ve yanıtların bir örneği yer almaktadır. Gerçek bilgi istemleri ve yanıtlarınız IBM MQ ağınıza bağlıdır.

```
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 ...
```

Not:

1. z/OS z/OSişletim sistemlerinde, (1)bilgi isteminde alanı boş bırakın.
2. Sunucu bağlantısını seçerseniz, (2)işaretli bilgi istemlerini görmezsiniz.
3. IBM i IBM iüzerinde, yalnızca QShell 'den `java MQIVP` komutunu yayınlayabilirsiniz. Diğer bir seçenek olarak, uygulamayı `RUNJVA CLASS(MQIVP)` Denetim dili (CL) komutunu kullanarak çalıştırabilirsiniz.

IBM MQ classes for Java örnek uygulamalarının kullanılması

IBM MQ classes for Java örnek uygulamaları, IBM MQ classes for Java API ' nin ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluş ve ileti sistemi sunucunuzun kurulumunu doğrulamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

Bu görev hakkında

Kendi uygulamalarınızı yaratmak için yardıma gereksinim duyarsanız, örnek uygulamaları başlangıç noktası olarak kullanabilirsiniz. Her uygulama için hem kaynak hem de derlenmiş bir sürüm sağlanır. Örnek kaynak kodunu gözden geçirin ve uygulamanız için gereken her bir nesneyi (MQQueueManager, MQConstants, MQMessage, MQPutMessageSeçenekleri ve MQDestination) yaratmak ve uygulamanızın nasıl çalışmasını istediğinizi belirtmek için gereken belirli özellikleri ayarlamak için gereken anahtar adımlarını tanımlayın. Daha fazla bilgi için bkz “IBM MQ classes for Java uygulamaları yazılıyor” sayfa 358. Örnekler, IBM MQ Javaürününün sonraki yayınlarında değiştirilebilir.

Çizelge 55 sayfa 355 , IBM MQ classes for Java örnek uygulamalarının her bir platformda nereye kurulduğunu gösterir:

Çizelge 55. IBM MQ classes for Java örnek uygulamaları için kuruluş dizinleri

Hizmet olarak sunulan	**Dizin**
AIX AIX Linux Linux	*MQ_INSTALLATION_PATH*`/samp/wmqjava/samples`
Windows Windows	*MQ_INSTALLATION_PATH*`\tools\wmqjava\samples`
IBM i IBM i	`/qibm/proddata/mqm/java/samples/wmqjava/samples`
z/OS z/OS	*MQ_INSTALLATION_PATH*`/java/samples/wmqjava`

Çizelge 56 sayfa 355 , IBM MQ classes for Javaile verilen örnek uygulama kümelerini gösterir.

Çizelge 56. IBM MQ classes for Java örnek uygulamalar

Örneğin adı	**Açıklama**
IMSBridgeSample.java	IBM MQ classes for Javaile IMS Köprüsü 'nün kullanılmasını gösteren basit bir program.

Çizelge 56. IBM MQ classes for Java örnek uygulamalar (devamı var)	
Örneğin adı	**Açıklama**
MQIVP.java	IBM MQ Java kuruluş doğrulama programı.
MQMessagePropertiesSample.java	İleti Özellikleri API 'sinin kullanımını gösterir.
MQPubSubApiSample.java	Yayınlama/abone olma API 'sini kullanarak gösterir.
MQSample.java	Bir kuyruktan ileti koymayı ve almayı gösteren basit bir program.
MQSampleMessageManager.java	IBM MQ Temel Java örneklerde ileti işlemeye ilişkin yardımcı program sınıfı.
mqjcivp.properties	Bu kaynak paketi, IBM MQ classes for Java kuruluş doğrulama programı (`MQIVP.java`) tarafından kullanılan iletileri içerir.

IBM MQ classes for Java , örnek uygulamaları çalıştırmak için kullanılabilecek `runjms` adlı bir komut dosyası sağlar. Bu komut dosyası, IBM MQ classes for Java örnek uygulamalarını çalıştırmanıza izin vermek için IBM MQ ortamını ayarlar.

Çizelge 57 sayfa 356 içinde her altyapıda komut dosyasının yeri gösterilir:

Çizelge 57. `runjms` komut dosyasının konumu	
Hizmet olarak sunulan	**Dizin**
AIX AIX Linux Linux	*MQ_INSTALLATION_PATH*`/java/bin/runjms`
Windows Windows	*MQ_INSTALLATION_PATH*`\java\bin\runjms.bat`

IBM i IBM i	`/qibm/proddata/mqm/java/bin/runjms`
	veya
	`/qibm/proddata/mqm/java/bin/runjms64`

z/OS z/OS	*MQ_INSTALLATION_PATH*`java/bin/runjms`

Örnek bir uygulamayı çağırmak üzere `runjms` komut dosyasını kullanmak için aşağıdaki adımları izleyin:

Yordam

1. Bir komut istemi açın ve çalıştırmak istediğiniz örnek uygulamayı içeren dizine gidin.
2. Aşağıdaki komutu girin:

   ```
   Path to the runjms script/runjms sample_application_name
   ```

   Örnek uygulama, gereksinim duyduğu parametrelerin bir listesini görüntüler.
3. Örneği bu parametrelerle çalıştırmak için aşağıdaki komutu girin:

   ```
   Path to the runjms script/runjms sample_application_name parameters
   ```

Örnek

Linux Örneğin, Linuxüzerinde MQIVP örneğini çalıştırmak için aşağıdaki komutları girin:

```
cd /opt/mqm/samp/wmqjava/samples
/opt/mqm/java/bin/runjms MQIVP
```

İlgili kavramlar

“IBM MQ classes for JMS için kurulu olan” sayfa 84
IBM MQ classes for JMSkurulurken bir dizi dosya ve dizin oluşturulur. Windows' ta, ortam değişkenleri otomatik olarak ayarlanarak kuruluş sırasında bazı yapılandırma gerçekleştirilir. Diğer platformlarda ve belirli Windows ortamlarında IBM MQ classes for JMS uygulamalarını çalıştırmadan önce ortam değişkenlerini ayarlamanız gerekir.

IBM MQ classes for Java sorunlarının çözülmesi

Başlangıçta kuruluş doğrulama programını çalıştırın. İzleme olanağını da kullanmanız gerekebilir.

Bir uygulama başarıyla tamamlanmazsa, kuruluş doğrulama programını çalıştırın ve tanılama iletilerinde verilen önerileri izleyin. Kuruluş doğrulama programı “IBM MQ classes for Java kuruluşunun doğrulanması” sayfa 354içinde açıklanmıştır.

Sorunlar devam ederse ve IBM hizmet ekibiyle iletişim kurmanız gerekirse, izleme olanağını açmanız istenebilir. Bunu aşağıdaki örnekte gösterildiği gibi yapın.

`MQIVP` programını izlemek için:

- Bir `com.ibm.mq.commonservices` özellikler dosyası oluşturun (bkz. com.ibm.mq.commonservices' ın kullanılması.
- Aşağıdaki komutu girin:

```
java -Dcom.ibm.mq.commonservices=commonservices_properties_file java
-Djava.library.path= library_path MQIVP -trace
```

Burada:

- *commonservices_properties_file* , `com.ibm.mq.commonservices` özellikler dosyasının yolu (dosya adı dahil).
- *kitaplık_yolu* , IBM MQ classes for Java kitaplıklarının yoludur (bkz. “IBM MQ classes for Java Kitaplıklar” sayfa 344).

İzlemeyi kullanma hakkında daha fazla bilgi için bkz. IBM MQ classes for Java uygulamaları izleme.

z/OS MQ Adv. VUE Java üzerinde çalışan toplu iş uygulamalarına istemci bağlanırlığı z/OS

Belirli koşullar altında, z/OS üzerindeki bir IBM MQ classes for Java uygulaması, z/OS üzerindeki bir kuyruk yöneticisine istemci bağlantısı kullanarak bağlanabilir. İstemci bağlantısının kullanılması IBM MQ topolojilerini basitleştirebilir.

Bir istemci bağlantısı kullanarak, IBM MQ classes for Java uygulaması toplu iş ortamında çalışıyorsa ve aşağıdaki koşullardan biri geçerliyse, uygulama uzak bir z/OS kuyruk yöneticisine bağlanabilir:

- LTS V 9.3.4 IBM MQ classes for Java kodu IBM MQ 9.3.4 ya da sonraki bir sürümde ya da Long Term Support APAR PH56722 uygulanmış olarak bulunur. Kuyruk yöneticisi desteklenen herhangi bir sürümde olabilir.
- Bağlı olduğu kuyruk yöneticisi IBM MQ Advanced for z/OS Value Unit Edition yetkisiyle çalışıyor ve bu nedenle **ADVCAP** parametresi ENABLED olarak ayarlanmış. Kuyruk yöneticisi desteklenen herhangi bir sürümde olabilir.

IBM MQ Advanced for z/OS Value Unit Edition hakkında daha fazla bilgi için bkz. IBM MQ ürün tanıtıcıları ve dışa aktarma bilgileri.

QMGRPRODile ilgili daha fazla bilgi için bkz. DISPLAY QMGR **ADVCAP** ve START QMGR .

z/OS üzerindeki bir IBM MQ classes for Java uygulaması, z/OS

z/OS üzerindeki bir IBM MQ classes for Java uygulaması istemci kipini kullanarak bağlanmayı denerse ve buna izin verilmezse, MQRC_ENVIRONMENT_ERROR döndürülür.

Advanced Message Security (AMS) desteği

IBM MQ classes for Java istemci uygulamaları, bu konuda daha önce açıklanan koşullara bağlı olarak, uzak z/OS kuyruk yöneticilerine bağlanırken AMS kullanabilir.

AMS ' u bu şekilde kullanmak için, istemci uygulamalarının `keystore.conf`içinde `jceracfks` anahtar deposu tipini kullanması gerekir; burada:

- Özellik adı öneki `jceracfks` ve bu ad öneki büyük ve küçük harfe duyarlı değildir.
- Anahtar deposu bir RACF anahtarlığı.
- Parolalar gerekli değildir ve yoksayılır. Bunun nedeni, RACF anahtarlarının parola kullanmaması olabilir.
- Sağlayıcıyı belirtirseniz, sağlayıcı `IBMJCE`olmalıdır.

`jceracfks` komutunu AMSile kullanırken anahtar deposu şu biçimde olmalıdır: `safkeyring://`*`user`*`/`*`keyring`*; burada:

- `safkeyring` bir hazır bilgi ve bu ad büyük ve küçük harfe duyarlı değildir,,,
- *`user`* , anahtarlık sahibinin RACF kullanıcı kimliğidir.
- *`keyring`* , RACF anahtarının adıdır ve anahtarlık adı büyük ve küçük harfe duyarlıdır

Aşağıdaki örnekte, kullanıcı için standart AMS anahtarlık `JOHNDOE`kullanılmıştır:

```
jceracfks.keystore=safkeyring://JOHNDOE/drq.ams.keyring
```

İlgili kavramlar

“JMS/Jakarta Messaging üzerinde çalışan toplu iş uygulamalarına istemci bağlanırlığı z/OS” sayfa 120
Belirli koşullar altında, z/OS üzerindeki bir IBM MQ classes for JMS/Jakarta Messaging uygulaması, z/OS üzerindeki bir kuyruk yöneticisine istemci bağlantısı kullanarak bağlanabilir. İstemci bağlantısının kullanılması IBM MQ topolojilerini basitleştirebilir.

IBM MQ classes for Java uygulamaları yazılıyor

Bu konu grubu, Java uygulamalarının IBM MQ sistemleriyle etkileşimde bulunmasına yardımcı olacak bilgiler sağlar.

IBM MQ kuyruklarına erişmek üzere IBM MQ classes for Java ' u kullanmak için, IBM MQ kuyruklarına ileti koyan ve kuyruklarından ileti alan çağrıları içeren Java uygulamaları yazarsınız. Tek tek sınıfların ayrıntıları için bkz. IBM MQ classes for Java.

Not: Otomatik istemci yeniden bağlantısı IBM MQ classes for Javatarafından desteklenmez.

IBM MQ classes for Java arabirimi

Yordamsal IBM MQ uygulama programlama arabirimi, nesneler üzerinde işlem yapan fiilleri kullanır. Java programlama arabirimi, yöntemleri çağırarak işlem yapacağınız nesneleri kullanır.

Yordamsal IBM MQ uygulama programlama arabirimi, aşağıdaki gibi fiillerin etrafında oluşturulmuştur:

```
MQBACK, MQBEGIN, MQCLOSE, MQCONN, MQDISC,
MQGET, MQINQ, MQOPEN, MQPUT, MQSET, MQSUB
```

Bu fiillerin tümü, parametre olarak, üzerinde çalışacakları IBM MQ nesnesi için bir tanıtıcı alır. Programınız, bu nesneler üzerinde yöntemleri çağırarak işlem yapacağınız bir IBM MQ nesneleri kümesinden oluşur.

Yordamsal arabirimi kullandığınızda, *Hconn* ' un kuyruk yöneticisi için bir tanıtıcı olduğu MQDISC (Hconn, CompCode, Reason) çağrısını kullanarak bir kuyruk yöneticisiyle bağlantı kesersiniz.

Java arabiriminde, kuyruk yöneticisi MQQueueManagersınıfındaki bir nesneyle gösterilir. Bu sınıftaki disconnect () yöntemini çağırarak kuyruk yöneticisiyle bağlantıyı kesersiniz.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.disconnect();
```

IBM MQ classes for Java bağlantı kipleri

IBM MQ classes for Java için programınız, kullanmak istediğiniz bağlantı kiplerine bazı bağımlılıklar içeriyor.

İstemci bağlantıları kullanıyorsanız, IBM MQ MQI client ile ilgili bir dizi farklılık vardır, ancak bu farklılıklar kavramsal olarak benzerdir. Bağ tanımlama kipini kullanıyorsanız, fastpath bağ tanımlarını kullanabilir ve MQBEGIN komutunu yürütebilirsiniz. MQEnvironment sınıfında değişkenleri ayarlayarak hangi kipin kullanılacağını belirtirsiniz.

IBM MQ classes for Java istemci bağlantıları

IBM MQ classes for Java istemci olarak kullanıldığında, IBM MQ MQI clientgibi olur, ancak çeşitli farklılıklar vardır.

IBM MQ classes for Java için istemci olarak kullanmak üzere programlama yapıyorsanız, aşağıdaki farklılıkları dikkate alın:

- Yalnızca TCP/IP ' yi destekler.
- Başlangıçta herhangi bir IBM MQ ortam değişkenlerini okumaz.
- Bir kanal tanımında ve ortam değişkenlerinde saklanacak bilgiler, Ortam adı verilen bir sınıfta saklanabilir. Diğer bir seçenek olarak, bağlantı kurulduğunda bu bilgiler parametre olarak iletilebilir.
- Hata ve kural dışı durum koşulları, `MQException` sınıfında belirtilen bir günlüğe yazılır. Varsayılan hata hedefi Java konsoludur.
- Bir IBM MQ istemci yapılandırma dosyasında yalnızca aşağıdaki öznitelikler IBM MQ classes for Javaile ilgilidir. Diğer öznitelikleri belirtirseniz, bunlar etkisiz olur.

Kıta	**Öznitelik**
ClientExitİstemci yapılanış kütüğünün yol kısmı	ExitsDefaultYolu
ClientExitİstemci yapılanış kütüğünün yol kısmı	ExitsDefaultPath64
ClientExitİstemci yapılanış kütüğünün yol kısmı	JavaExitsClasspath
İstemci yapılanış kütüğünün MessageBuffer kısmı	MaximumSize
İstemci yapılanış kütüğünün MessageBuffer kısmı	PurgeTime
İstemci yapılanış kütüğünün MessageBuffer kısmı	UpdatePercentage
İstemci yapılanış kütüğünün TCP kısmı	ClntRcvBuffSize
İstemci yapılanış kütüğünün TCP kısmı	ClntSndBuffSize
İstemci yapılanış kütüğünün TCP kısmı	Connect_Timeout
İstemci yapılanış kütüğünün TCP kısmı	KeepAlive

- Karakter verilerinin dönüştürülmesini gerektiren bir kuyruk yöneticisine bağlanıyorsanız, V7 Java istemcisi artık kuyruk yöneticisi bunu yapamazsa dönüştürmeyi gerçekleştirebilir. İstemci JVM, istemcinin CCSID 'si ile kuyruk yöneticisinin CCSID 'si arasında dönüştürmeyi desteklemelidir.

- Otomatik istemci yeniden bağlantısı IBM MQ classes for Javatarafından desteklenmez.

İstemci kipinde kullanıldığında, *IBM MQ classes for Java* MQBEGIN çağrılmasını desteklemez.

IBM MQ classes for Java bağ tanımları kipi
IBM MQ classes for Java bağ tanımlama kipi, istemci kipinden üç ana şekilde farklıdır.

Bağ tanımlama kipinde kullanıldığında IBM MQ classes for Java , bir ağ üzerinden iletişim kurmak yerine, var olan kuyruk yöneticisi API 'sine doğrudan çağırmak için Java Native Interface (JNI) olanağını kullanır.

Varsayılan olarak, IBM MQ classes for Java in bağ tanımlama kipini kullanan uygulamalar, *ConnectOption*, MQCNO_STANDARD_BAĞ tanımlarını kullanarak bir kuyruk yöneticisine bağlanır.

IBM MQ classes for Java aşağıdaki *ConnectOptions*ürününü destekler:

- MQCNO_FASTPATH_BINDING
- MQCNO_STANDARD_BINDING
- MQCNO_SHARED_BINDING
- MQCNO_ISOLATED_BINDING

*ConnectOptions*ile ilgili daha fazla bilgi için bkz. "MQCONNX çağrısı kullanılarak bir kuyruk yöneticisiyle bağlantı kurulması" sayfa 708.

Bağ tanımları kipi, IBM MQ for IBM i ve IBM MQ for z/OSdışındaki tüm altyapılarda kuyruk yöneticisi tarafından koordine edilen genel iş birimlerini başlatmak için MQBEGIN çağrılmasını destekler.

`MQEnvironment` sınıfı tarafından sağlanan parametrelerin çoğu bağ tanımlama kipiyle ilgili değildir ve yoksayılır.

Hangi IBM MQ classes for Java bağlantısının kullanılacağını tanımlama
Kullanılacak bağlantı tipi, MQEnvironment sınıfındaki değişkenlerin ayarlanmasıyla belirlenir.

İki değişken kullanılır:

MQEnvironment.properties
Bağlantı tipi, `CMQC.TRANSPORT_PROPERTY`anahtar adıyla ilişkili değere göre belirlenir. Olası değerler şunlardır:

CMQC.TRANSPORT_MQSERIES_BINDINGS
Bağ tanımları kipinde bağlan

CMQC.TRANSPORT_MQSERIES_CLIENT
İstemci kipinde bağlan

CMQC.TRANSPORT_MQSERIES
Bağlantı kipi, *hostname* özelliğinin değerine göre belirlenir

MQEnvironment.hostname
Bu değişkenin değerini aşağıdaki gibi ayarlayın:

- İstemci bağlantıları için, bu değişkenin değerini, bağlanmak istediğiniz IBM MQ sunucusunun anasistem adına ayarlayın.
- Bağ tanımlama kipi için bu değişkeni ayarlamayın ya da boş değere ayarlayın

Kuyruk yöneticilerindeki işlemler
Bu konular derlemi, IBM MQ classes for Javakullanılarak bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini açıklar.

IBM MQ classes for Java için IBM MQ ortamını ayarlama
Bir uygulamanın istemci kipinde bir kuyruk yöneticisine bağlanması için, uygulamanın kanal adını, anasistem adını ve kapı numarasını belirtmesi gerekir.

Not: Bu konudaki bilgiler, uygulamanızın istemci kipinde bir kuyruk yöneticisine bağlanması durumunda anlamlıdır. Bağ tanımlama kipinde bağlanıyorsa ilgili değildir. Bkz: "IBM MQ classes for JMS için bağlantı kipleri" sayfa 105

Kanal adını, anasistem adını ve kapı numarasını şu iki yoldan biriyle belirtebilirsiniz: MQEnvironment sınıfındaki alanlar olarak ya da MQQueueManager nesnesinin özellikleri olarak.

MQEnvironment sınıfında alanlar ayarlarsanız, bu alanlar bir özellikler HASH çizelgesi tarafından geçersiz kılınmaları dışında, tüm uygulamanız için geçerlidir. MQEnvironment 'ta kanal adını ve anasistem adını belirtmek için aşağıdaki kodu kullanın:

```
MQEnvironment.hostname = "host.domain.com";
MQEnvironment.channel = "java.client.channel";
```

Bu, **MQSERVER** ortam değişkeninin ayarlanmasıyla eşdeğerdir:

```
"java.client.channel/TCP/host.domain.com".
```

Varsayılan olarak, Java istemcileri 1414kapısında bir IBM MQ dinleyicisine bağlanmayı dener. Farklı bir kapı belirtmek için aşağıdaki kodu kullanın:

```
MQEnvironment.port = nnnn;
```

Burada nnnn gerekli kapı numarasıdır

Özellikleri yaratma sırasında bir kuyruk yöneticisi nesnesine geçirirseniz, bu özellikler yalnızca o kuyruk yöneticisi için geçerlidir. **hostname**, **channel**ve isteğe bağlı olarak **port**anahtarlarıyla ve uygun değerlerle bir Hashtable nesnesinde girdiler oluşturun. Varsayılan kapıyı (1414) kullanmak için **port** girdisini atlayabilirsiniz. Özellikler HASH çizelgesini kabul eden bir oluşturucu kullanarak MQQueueManager nesnesini yaratın.

Bir uygulama adı ayarlayarak kuyruk yöneticisiyle bağlantı tanımlanması

Bir uygulama, kuyruk yöneticisiyle bağlantısını tanıtan bir ad ayarlayabilir. Bu uygulama adı **DISPLAY CONN MQSC/PCF** komutuyla gösterilir (burada alan **APPLTAG** olarak adlandırılır) ya da IBM MQ Gezgin **Uygulama Bağlantıları** ekranında (alanın **App name** olarak adlandırıldığı yerde).

Uygulama adları 28 karakterle sınırlıdır, bu nedenle daha uzun adlar kesilir. Bir uygulama adı belirtilmezse, varsayılan değer sağlanır. Varsayılan ad çağıran (ana) sınıfa dayalıdır, ancak bu bilgi yoksa `IBM MQ Client for Java` metni kullanılır.

Çağıran sınıfın adı kullanılırsa, gerekirse, baştaki paket adları kaldırılarak sığacak şekilde ayarlanır. Örneğin, çağıran sınıf `com.example.MainAppise`, tam ad kullanılır, ancak çağıran sınıf `com.example.dictionaryAndThesaurus.multilingual.mainAppise`, kullanılabilir uzunluğa uyan en uzun sınıf adı ve en sağdaki paket adı birleşimi olduğundan `multilingual.mainApp` adı kullanılır.

Sınıf adının kendisi 28 karakterden uzunsa, sığacak şekilde kesilir. Örneğin, `com.example.mainApplicationForSecondTestCase` , `mainApplicationForSecondTest`olur.

MQEnvironment sınıfında bir uygulama adı ayarlamak için, adı aşağıdaki kodu kullanarak MQEnvironment.properties HASH çizelgesine **MQConstants.APPNAME_PROPERTY**anahtarıyla ekleyin:

```
MQEnvironment.properties.put(MQConstants.APPNAME_PROPERTY, "my_application_name");
```

MQQueueManager oluşturucusuna geçirilen özellikler HASH çizelgesinde bir uygulama adı ayarlamak için, adı özellikler HASH çizelgesine **MQConstants.APPNAME_PROPERTY**anahtarıyla ekleyin.

IBM MQ istemcisi yapılandırma dosyasında belirtilen özelliklerin geçersiz kılınması

IBM MQ istemcisi yapılandırma dosyası, IBM MQ classes for Java' yi yapılandırmak için kullanılan özellikleri de belirtebilir. Ancak, IBM MQ MQI client yapılanış dosyasında belirtilen özellikler yalnızca, bir uygulama istemci kipinde bir kuyruk yöneticisine bağlandığında geçerlidir.

Gerekirse, IBM MQ yapılandırma dosyasındaki herhangi bir özniteliği aşağıdaki yollardan biriyle geçersiz kılabilirsiniz. Seçenekler öncelik sırasına göre gösterilir.

- Yapılandırma özelliği için bir Java sistem özelliği ayarlayın.
- MQEnvironment.properties eşleminde özelliği ayarlayın.
- Java5 ve sonraki yayın düzeylerinde bir sistem ortam değişkeni ayarlayın.

Bir IBM MQ istemci yapılandırma dosyasında yalnızca aşağıdaki öznitelikler IBM MQ classes for Javaile ilgilidir. Diğer öznitelikleri belirtirseniz ya da geçersiz kılarsanız, bunun bir etkisi olmaz.

Kıta	**Öznitelik**
ClientExitİstemci yapılanış kütüğünün yol kısmı	ExitsDefaultYolu
ClientExitİstemci yapılanış kütüğünün yol kısmı	ExitsDefaultPath64
ClientExitİstemci yapılanış kütüğünün yol kısmı	JavaExitsClasspath
İstemci yapılanış kütüğünün MessageBuffer kısmı	MaximumSize
İstemci yapılanış kütüğünün MessageBuffer kısmı	PurgeTime
İstemci yapılanış kütüğünün MessageBuffer kısmı	UpdatePercentage
İstemci yapılanış kütüğünün TCP kısmı	ClntRcvBufSize
İstemci yapılanış kütüğünün TCP kısmı	ClntSndBufSize
İstemci yapılanış kütüğünün TCP kısmı	Connect_Timeout
İstemci yapılanış kütüğünün TCP kısmı	KeepAlive

IBM MQ classes for Java içinde bir kuyruk yöneticisine bağlanma
MQQueueManager sınıfının yeni bir eşgörünümünü yaratarak bir kuyruk yöneticisine bağlanın. Disconnect () yöntemini çağırarak bir kuyruk yöneticisiyle bağlantıyı kesin.

Artık MQQueueManager sınıfının yeni bir eşgörünümünü yaratarak bir kuyruk yöneticisine bağlanmaya hazırsınız:

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Bir kuyruk yöneticisiyle bağlantıyı kesmek için, kuyruk yöneticisinde disconnect () yöntemini çağırın:

```
queueManager.disconnect();
```

Bağlantı kesme yöntemini çağırırsanız, o kuyruk yöneticisinden eriştiğiniz tüm açık kuyruklar ve işlemler kapatılır. Ancak, bu kaynakları kullanmayı bitirdiğinizde açık bir şekilde kapatmak iyi bir programlama uygulamasıdır. Bunu yapmak için, ilgili nesneler üzerinde close () yöntemini kullanın.

Bir kuyruk yöneticisindeki commit () ve backout () yöntemleri, yordamsal arabirimle kullanılan MQCMIT ve MQBACK çağrılarına eşdeğerdir.

IBM MQ classes for Java ile istemci kanal tanımlama çizelgesinin kullanılması
IBM MQ classes for Java istemci uygulaması, bir istemci kanal tanımlama çizelgesinde (CCDT) saklanan istemci bağlantısı kanal tanımlarını kullanabilir.

Bir IBM MQ classes for Java istemci uygulaması, `MQEnvironment` sınıfında belirli alanları ve ortam özelliklerini ayarlayarak ya da bir özellikler HASH çizelgesinde `MQQueueManager` ' e geçirerek istemci bağlantı kanalı tanımlaması yaratmanın bir alternatifi olarak, istemci kanal tanımlama çizelgesinde saklanan istemci bağlantısı kanal tanımlarını kullanabilir. Bu tanımlamalar, IBM MQ Script (MQSC) komutları ya da IBM MQ Programlanabilir Komut Biçimi (PCF) komutları ya da IBM MQ Explorer kullanılarak yaratılır.

Uygulama bir MQQueueManager nesnesi yarattığında, IBM MQ classes for Java istemcisi kanal tanımlama çizelgesinde uygun bir istemci bağlantı kanalı tanımlaması arar ve bir MQI kanalını başlatmak için kanal tanımlamasını kullanır. İstemci kanal tanımlama çizelgelerine ve bunların nasıl oluşturulacağına ilişkin ek bilgi için İstemci kanal tanımlama çizelgesibaşlıklı konuya bakın.

Bir istemci kanal tanımlama çizelgesini kullanmak için, uygulamanın önce bir URL nesnesi yaratması gerekir. URL nesnesi, istemci kanal tanımlama çizelgesini içeren dosyanın adını ve yerini tanımlayan ve dosyaya nasıl erişilebileceğini belirten bir tek tip kaynak konum belirleyici (URL) içerir.

Örneğin, ccdt1.tab dosyası bir istemci kanal tanımlama çizelgesi içeriyorsa ve uygulamanın çalıştığı sistemde saklanıyorsa, uygulama aşağıdaki şekilde bir URL nesnesi yaratabilir:

```
java.net.URL chanTab1 = new URL("file:///home/admdata/ccdt1.tab");
```

Başka bir örnek olarak, ccdt2.tab dosyasının bir istemci kanal tanımlama çizelgesi içerdiğini ve uygulamanın çalıştığından farklı bir sistemde saklandığını varsayın. Dosyaya FTP iletişim kuralı kullanılarak erişilebiliyorsa, uygulama aşağıdaki şekilde bir URL nesnesi oluşturabilir:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
```

Uygulama bir URL nesnesi yarattıktan sonra, URL nesnesini parametre olarak alan oluşturucuları kullanarak bir MQQueueManager nesnesi yaratabilir. Örnek:

```
MQQueueManager mars = new MQQueueManager("MARS", chanTab2);
```

Bu deyim, IBM MQ classes for Java istemcisinin URL nesnesi chanTab2ile tanıtılan istemci kanal tanımlama çizelgesine erişmesine, çizelgede uygun bir istemci bağlantısı kanal tanımlaması aramasına ve kanal tanımlamasını kullanarak MARS adlı kuyruk yöneticisine bir MQI kanalı başlatmasına neden olur.

Bir uygulama bir istemci kanal tanımlama çizelgesi kullanıyorsa, geçerli olan aşağıdaki noktaları göz önünde bulundurun:

- Uygulama, bir URL nesnesini parametre olarak alan bir oluşturucu kullanarak bir MQQueueManager nesnesi yarattığında, MQEnvironment sınıfında bir alan ya da ortam özelliği olarak hiçbir kanal adı ayarlanmamalıdır. Bir kanal adı ayarlanırsa, IBM MQ classes for Java istemcisi bir MQExceptionatar. Kanal adını belirten alan ya da ortam özelliği, değeri boş değer, boş dizgi ya da tüm boş karakterleri içeren bir dizgi dışında bir değerse ayarlanır.
- MQQueueManager oluşturucusundaki **queueManagerName** parametresi aşağıdaki değerlerden birine sahip olabilir:
  - Kuyruk yöneticisinin adı
  - Bir yıldız imi (*) ve ardından bir kuyruk yöneticisi grubu adı
  - Yıldız işareti (*)
  - Boş değerli, boş bir dizgi ya da tüm boş karakterleri içeren bir dizgi

  Bunlar, Message Queue Interface (MQI) kullanan bir istemci uygulaması tarafından verilen bir MQCONN çağrısındaki **QMgrName** değiştirgesi için kullanılabilecek değerlerle aynıdır. Bu değerlerin anlamı hakkında daha fazla bilgi için bkz.“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692.

  Uygulamanız bağlantı havuzlama özelliğini kullanıyorsa, bkz.“IBM MQ classes for Java içinde varsayılan bağlantı havuzunun denetlenmesi” sayfa 382.
- IBM MQ classes for Java istemcisi, istemci kanal tanımlama çizelgesinde uygun bir istemci bağlantı kanalı tanımlaması bulduğunda, MQI kanalını başlatmak için yalnızca bu kanal tanımlamasından alınan bilgileri kullanır. Uygulamanın MQEnvironment sınıfında ayarlamış olabileceği kanalla ilgili alanlar ya da ortam özellikleri yoksayılır.

  Özellikle, TLS (Transport Layer Security; İletim Katmanı Güvenliği) kullanıyorsanız aşağıdaki noktalara dikkat edin:

– Bir MQI kanalı TLS ' yi yalnızca istemci kanal tanımlaması çizelgesinden çıkarılan kanal tanımlaması IBM MQ classes for Java istemcisi tarafından desteklenen bir CipherSpec adını belirtiyorsa kullanır.
– Bir istemci kanal tanımlama çizelgesi, sertifika iptal listelerini (CRL ' ler) bulunduran LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) sunucularının yeriyle ilgili bilgiler de içerir. IBM MQ classes for Java istemcisi, CRL ' leri bulunduran LDAP sunucularına erişmek için yalnızca bu bilgileri kullanır.
– Bir istemci kanal tanımlama çizelgesi, OCSP yanıtlayıcısının yerini de içerebilir. IBM MQ classes for Java , bir istemci kanal tanımlama çizelgesi kütüğünde OCSP bilgilerini kullanamıyor. Ancak, OCSP ' yi Çevrimiçi Sertifika Protokolünü Kullanma bölümünde açıklandığı gibi yapılandırabilirsiniz.

TLS ' yi bir istemci kanal tanımlama çizelgesiyle kullanma hakkında daha fazla bilgi için MQI kanalının TLS kullandığını belirtmebaşlıklı konuya bakın.

Kanal çıkışlarını kullanıyorsanız aşağıdaki noktalara da dikkat edin:

– Bir MQI kanalı, diğer yöntemlerle belirtilen kanal çıkışlarını ve verilerini tercih etmek için istemci kanal tanımlama çizelgesinden alınan kanal tanımlamasıyla belirtilen kanal çıkışlarını ve ilişkili kullanıcı verilerini kullanır.
– Bir istemci kanal tanımlama çizelgesinden çıkarılan bir kanal tanımı, Java, C ya da C + + ile yazılan kanal çıkışlarını belirtebilir. Java içinde bir kanal çıkışının nasıl yazılacağına ilişkin ek bilgi için bkz. “IBM MQ classes for Java içinde kanal çıkışı oluşturma” sayfa 376. Kanal çıkışının diğer dillerde yazılması hakkında daha fazla bilgi için bkz. “IBM MQ classes for Java ile Java içinde yazılmamış kanal çıkışlarının kullanılması” sayfa 380.

IBM MQ classes for Java istemci bağlantıları için kapı aralığı belirtilmesi
Bir uygulamanın iki yoldan biriyle bağlayabileceği bir kapı ya da kapı aralığı belirtebilirsiniz.

IBM MQ classes for Java uygulaması istemci kipinde bir IBM MQ kuyruk yöneticisine bağlanmaya çalıştığında, güvenlik duvarı yalnızca belirtilen kapılardan ya da kapı aralığından kaynaklanan bağlantılara izin verebilir. Bu durumda, uygulamanın bağlanabileceği bir kapı ya da kapı aralığı belirtebilirsiniz. Kapı (lar) ı aşağıdaki şekillerde belirtebilirsiniz:

- MQEnvironment sınıfında localAddressAyarı alanını ayarlayabilirsiniz. Örnek:

```
MQEnvironment.localAddressSetting = "192.0.2.0(2000,3000)";
```

- CMQC.LOCAL_ADDRESS_PROPERTY. Örnek:

```
(MQEnvironment.properties).put(CMQC.LOCAL_ADDRESS_PROPERTY,
                               "192.0.2.0(2000,3000)");
```

- MQQueueManager nesnesini oluşturabildiğinizde, "192.0.2.0(2000,3000)" değeriyle LOCAL_ADDRESS_PROPERTY içeren bir hashtable özelliğini iletebilirsiniz.

Bu örneklerin her birinde, uygulama daha sonra bir kuyruk yöneticisine bağlandığında, uygulama 192.0.2.0(2000) ile 192.0.2.0(3000) aralığında yerel bir IP adresine ve kapı numarasına bağlanır.

Birden çok ağ arabirimi olan bir sistemde, localAddressSetting alanını ya da CMQC.LOCAL_ADDRESS_PROPERTY.

Kapı aralığını kısıtlarsanız bağlantı hataları oluşabilir. Bir hata oluşursa, MQRC_Q_MGR_NOT_KULLANILABILIR IBM MQ neden kodunu ve aşağıdaki iletiyi içeren bir MQException yayınlanır:

```
Socket connection attempt refused due to LOCAL_ADDRESS_PROPERTY restrictions
```

Belirtilen aralıktaki tüm kapılar kullanımdaysa ya da belirtilen IP adresi, anasistem adı ya da kapı numarası geçerli değilse (örneğin, negatif bir kapı numarası) bir hata oluşabilir.

IBM MQ classes for Java içindeki kuyruklara, konulara ve süreçlere erişilmesi

Kuyruklara, konulara ve süreçlere erişmek için MQQueueManager sınıfının yöntemlerini kullanın. MQOD (nesne tanımlayıcı yapısı) bu yöntemlerin değiştirgelerine daraltıldı.

Kuyruklar

Bir kuyruğu açmak için MQQueueManager sınıfının accessQueue yöntemini kullanabilirsiniz. Örneğin, queueManageradlı bir kuyruk yöneticisinde aşağıdaki kodu kullanın:

```
MQQueue queue = queueManager.accessQueue("qName",CMQC.MQOO_OUTPUT);
```

accessQueue yöntemi, MQQueue sınıfının yeni bir nesnesini döndürür.

Kuyruğu kullanmayı tamamladığınızda, aşağıdaki örnekte olduğu gibi, kuyruğu kapatmak için close () yöntemini kullanın:

```
queue.close();
```

MQQueue oluşturucusunu kullanarak da kuyruk yaratabilirsiniz. Değiştirgeler, bir kuyruk yöneticisi değiştirgesinin eklenmesiyle accessQueue yöntemiyle aynıdır. Örneğin:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            CMQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserID");
```

Kuyruklar yaratırken bir dizi seçenek belirleyebilirsiniz. Bunların ayrıntıları için bkz. Class.com.ibm.mq.MQQueue. Bu şekilde bir kuyruk nesnesi oluşturulması, kendi MQQueue alt sınıflarınızı yazmanızı sağlar.

Konular

Benzer şekilde, MQQueueManager sınıfının accessTopic yöntemini kullanarak bir konuyu açabilirsiniz. Örneğin, queueManageradlı bir kuyruk yöneticisinde abone ve yayınlayıcı yaratmak için aşağıdaki kodu kullanın:

```
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);
```

Konuyu kullanmayı bitirdiğinizde, konuyu kapatmak için close () yöntemini kullanın.

MQTopic oluşturucusunu kullanarak da konu yaratabilirsiniz. Değiştirgeler, bir kuyruk yöneticisi değiştirgesi eklenmiş olarak, accessTopic yöntemiyle aynıdır. Örneğin:

```
MQTopic subscriber = new
      MQTopic(queueManager,"TOPICSTRING","TOPICNAME",
      CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

Konu oluştururken bir dizi seçenek belirtebilirsiniz. Bunların ayrıntıları için bkz. Class com.ibm.mq.MQTopic. Bu şekilde bir konu nesnesi oluşturmak, kendi MQTopic alt sınıflarınızı yazmanızı sağlar.

Yayın ya da abonelik için bir konu açılmalıdır. MQQueueManager sınıfının sekiz accessTopic yöntemi vardır ve Konu sınıfının sekiz oluşturucusu vardır. Her bir durumda, bunlardan dördünün bir **destination** parametresi ve dördünün bir **subscriptionName** parametresi (her ikisinin de bulunduğu iki parametre dahil) vardır. Bunlar yalnızca abonelikler için konuyu açmak için kullanılabilir. Geri kalan iki yöntemin bir **openAs** parametresi vardır ve konu, **openAs** parametresinin değerine bağlı olarak yayın ya da abonelik için açılabilir.

Sürekli abone olarak bir konu yaratmak için, MQQueueManager sınıfının accessTopic yöntemini ya da abonelik adını kabul eden bir MQTopic oluşturucusunu kullanın ve her iki durumda da CMQC.MQSO_DURABLE seçeneği.

Süreçler

Bir sürece erişmek için MQQueueManager' ın accessProcess yöntemini kullanın. Örneğin, queueManageradlı bir kuyruk yöneticisinde, MQProcess nesnesi yaratmak için aşağıdaki kodu kullanın:

```
MQProcess process =
queueManager.accessProcess("PROCESSNAME",
CMQC.MQOO_FAIL_IF_QUIESCING);
```

Bir sürece erişmek için MQQueueManager' ın accessProcess yöntemini kullanın.

accessProcess yöntemi, MQProcess sınıfının yeni bir nesnesini döndürür.

Süreç nesnesini kullanmayı tamamladığınızda, aşağıdaki örnekte olduğu gibi, nesneyi kapatmak için close () yöntemini kullanın:

```
process.close();
```

MQProcess oluşturucusunu kullanarak da süreç yaratabilirsiniz. Değiştirgeler, bir kuyruk yöneticisi değiştirgesi eklenmiş olarak, accessProcess yöntemiyle tam olarak aynıdır. Örneğin:

```
MQProcess process =
      new MQProcess(queueManager,"PROCESSNAME",
      CMQC.MQOO_FAIL_IF_QUIESCING);
```

Bu şekilde bir süreç nesnesi oluşturulması, kendi MQProcess alt sınıflarınızı yazmanızı sağlar.

IBM MQ classes for Java içindeki iletileri işleme

İletiler MQMessage sınıfıyla gösterilir. İletileri, MQQueue ve MQTopic 'in alt sınıflarına sahip MQDestination sınıfının yöntemlerini kullanarak yerleştirdiniz ve aldınız.

MQDestination sınıfının Put () yöntemini kullanarak iletileri kuyruklara ya da konulara koyun. MQDestination sınıfının get () yöntemini kullanarak kuyruklardan ya da konulardan ileti alıyorsunuz. MQPUT ve MQGET ' in byte dizilerini koyduğu ve aldığı yordamsal arabirimden farklı olarak, Java programlama dili MQMessage sınıfının eşgörünümlerini yerleştirir ve alır. MQMessage sınıfı, gerçek ileti verilerini içeren veri arabelleğini, o iletiyi tanımlayan tüm MQMD (ileti tanımlayıcı) parametreleri ve ileti özellikleriyle birlikte içerir.

Yeni bir ileti oluşturmak için MQMessage sınıfının yeni bir eşgörünümünü oluşturun ve verileri ileti arabelleğine koymak için writeXXX yöntemlerini kullanın.

Yeni ileti eşgörünümü yaratıldığında, MQMD için başlangıç değerleri ve dil bildirimleriiçinde tanımlandığı şekilde, tüm MQMD parametreleri otomatik olarak varsayılan değerlerine ayarlanır. MQDestination 'ın Put () yöntemi, parametre olarak MQPutMessageOptions sınıfının bir eşgörünümünü de alır. Bu sınıf MQPMO yapısını temsil eder. Aşağıdaki örnek bir ileti yaratır ve bir kuyruğa koyar:

```
// 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);
```

MQDestination 'ın get () yöntemi, kuyruktan yeni alınan iletiyi gösteren yeni bir MQMessage eşgörünümü döndürür. Değiştirge olarak MQGetMessageSeçenekleri sınıfının bir eşgörünümünü de alır. Bu sınıf MQGMO yapısını temsil eder.

İleti büyüklüğü üst sınırını belirtmenize gerek yoktur; get () yöntemi, iç arabelleğinin büyüklüğünü gelen iletiye sığacak şekilde otomatik olarak ayarlar. Döndürülen iletideki verilere erişmek için MQMessage sınıfının readXXX yöntemlerini kullanın.

Aşağıdaki örnek, bir kuyruktan ileti nasıl alacağınızı göstermektedir:

```
// 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);
```

Okuma ve yazma yöntemlerinin kullandığı sayı biçimini *kodlama* üye değişkenini ayarlayarak değiştirebilirsiniz.

characterSet üye değişkenini ayarlayarak karakter kümesini, dizgileri okumak ve yazmak için kullanılacak şekilde değiştirebilirsiniz.

Ek bilgi için MQMessage class başlıklı konuya bakın.

Not: MQMessage writeUTF() yöntemi, içerdiği Unicode byte 'ların yanı sıra dizginin uzunluğunu da otomatik olarak kodlar. İletiniz başka bir Java programı tarafından okunduğunda (readUTF() kullanılarak), dizgi bilgisi göndermenin en basit yolu budur.

IBM MQ classes for Java ürününde kalıcı olmayan iletilerin performansını iyileştirme
İletilere göz atarken ya da istemci uygulamasından kalıcı olmayan iletileri kullanırken başarımı artırmak için *önden okuma*seçeneğini kullanabilirsiniz. MQGET ya da zamanuyumsuz tüketim kullanan istemci uygulamaları, iletilere göz atarken ya da kalıcı olmayan iletileri kullanırken performans geliştirmelerinden yararlanacaktır.

Önden okuma olanağıyla ilgili genel bilgi için ilgili konuya bakın.

IBM MQ classes for Javaiçinde CMQC.MQSO_READ_AHEAD ve CMQC.MQSO_NO_READ_AHEAD özellikleri, ileti tüketicilerinin ve kuyruk tarayıcılarının o nesnede önden okuma kullanmalarına izin verilip verilmediğini saptamak için kullanılır.

IBM MQ classes for Java komutunu kullanarak iletileri zamanuyumsuz olarak koyma
Bir iletiyi zamanuyumsuz olarak koymak için MQPMO_ASYNC_RESPONSE değerini ayarlayın.

İletileri, MQDestination sınıfının Put () yöntemini kullanarak kuyruklara ya da konulara koyarsınız. Bir iletiyi zamanuyumsuz olarak, yani kuyruk yöneticisinden yanıt beklemeden işlemin tamamlanmasını sağlamak için, MQPutMessageSeçenekleri 'nin seçenekler alanında MQPMO_ASYNC_RESPONSE değerini ayarlayabilirsiniz. Zamanuyumsuz girişlerin başarılı ya da başarısız olduğunu saptamak için MQQueueManager.getAsyncDurum çağrısını kullanın.

IBM MQ classes for Java içinde yayınla/abone ol

IBM MQ classes for Javaiçinde konu MQTopic sınıfı tarafından temsil edilir ve MQTopic.put() yöntemlerini kullanarak konuyu yayınlarsınız.

IBM MQ yayınlama/abone olma hakkında genel bilgi için bkz. İleti alışverişi yayınlama/abone olma.

IBM MQ ileti üstbilgilerinin IBM MQ classes for Java ile işlenmesi

Farklı ileti üstbilgisi tiplerini gösteren Java sınıfları sağlanır. İki yardımcı sınıf da sağlanır.

MQHeader arabirimi

Üstbilgi nesneleri, üstbilgi alanlarına erişmek ve ileti içeriğini okumak ve yazmak için genel amaçlı yöntemler sağlayan MQHeader arabirimi tarafından tanımlanır. Her üstbilgi tipinin, MQHeader arabirimini gerçekleştiren ve tek tek alanlar için alıcı ve ayarlayıcı yöntemleri ekleyen kendi sınıfı vardır. Örneğin, MQRFH2 üstbilgi tipi MQRFH2 sınıfıyla gösterilir; MQDLH üstbilgi tipi MQDLH sınıfıyla gösterilir, vb. Üstbilgi sınıfları, gerekli veri dönüştürme işlemlerini otomatik olarak gerçekleştirir ve belirlenen herhangi bir sayısal kodlama ya da karakter takımındaki (CCSID) verileri okuyabilir ya da yazabilir.

Önemli: MQRFH2 üstbilgi sınıfları iletiyi rasgele erişim dosyası olarak görür; bu, imlecin iletinin başında konumlandırılması gerektiği anlamına gelir. MQRFH, MQRFH2, MQCIH, MQDEAD, MQIIH ya da MQXMIT gibi bir iç ileti üstbilgisi sınıfını kullanmadan önce, iletiyi sınıfa geçirmeden önce iletinin imleç konumunu doğru konuma güncellediğinizden emin olun.

Yardımcı sınıflar

İki yardımcı sınıf (MQHeaderIterator ve MQHeaderList), iletilerdeki üstbilgi içeriğinin okunmasına ve kodunun çözülmesine (ayrıştırılmasına) yardımcı olacaktır:

- MQHeaderIterator sınıfı java.util.Iteratorgibi çalışır. İletide daha fazla üstbilgi olduğu sürece, next () yöntemi true değerini döndürür ve nextHeader() ya da next () yöntemi sonraki üstbilgi nesnesini döndürür.
- MQHeaderList , java.util.Listgibi çalışır. MQHeaderIteratorgibi, üstbilgi içeriğini ayrıştırır, ancak belirli üstbilgileri aramanızı, yeni üstbilgiler eklemenizi, var olan üstbilgileri kaldırmanızı, üstbilgi alanlarını güncellemenizi ve üstbilgi içeriğini bir iletiye geri yazmanızı sağlar. Diğer bir seçenek olarak, boş bir MQHeaderListyaratabilir, sonra bunu üstbilgi eşgörünümleriyle doldurabilir ve bir iletiye bir kez ya da defalarca yazabilirsiniz.

MQHeaderIterator ve MQHeaderList sınıfları, belirli ileti tipleri ve biçimleriyle hangi IBM MQ üstbilgi sınıflarının ilişkilendirildiğini öğrenmek için MQHeaderRegistry içindeki bilgileri kullanır. MQHeaderRegistry , tüm yürürlükteki IBM MQ biçimlerinin ve üstbilgi tiplerinin ve bunların uygulama sınıflarının bilgisiyle yapılandırılır ve kendi üstbilgi tiplerinizi de kaydedebilirsiniz.

Aşağıdaki yaygın olarak kullanılan IBM MQ üstbilgileri için destek sağlanır

- MQRFH-Kurallar ve biçimlendirme üstbilgisi
- MQRFH2 - IBM Integration Bus' e ait bir Message Broker 'a ileti iletmek için kullanılan MQRFH gibi. İleti özelliklerini içermek için de kullanılır
- MQCIH- CICS Köprüsü
- MQDLH-Harf üstbilgisi
- MQIIH- IMS bilgi üstbilgisi
- MQRMH-başvuru iletisi üstbilgisi
- MQSAPH- SAP üstbilgisi
- MQWIH-İş bilgileri üstbilgisi
- MQXQH-İletim Kuyruğu üstbilgisi
- MQDH-Dağıtım üstbilgisi
- MQEPH-Kapsüllenmiş PCF üstbilgisi

Kendi üstbilgilerinizi gösteren sınıfları da tanımlayabilirsiniz.

Bir RFH2 üstbilgisi almak üzere MQHeaderIterator kullanmak için, GetMessageSeçenekleri 'nde MQGMO_PROPERTIES_FORCE_MQRFH2 ayarını yapın ya da PROPCTL kuyruk özelliğini FORCE olarak ayarlayın.

IBM MQ classes for Java komutunu kullanarak bir iletideki tüm üstbilgileri yazdırma
Bu örnekte, MQHeaderIterator yönetim ortamı, kuyruktan alınan bir MQMessage içindeki üstbilgileri ayrıştırır. nextHeader() yönteminin döndürdüğü MQHeader nesneleri, toString yöntemi çağrıldığında yapılarını ve içeriklerini görüntüler.

```
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);
}
```

IBM MQ classes for Java komutunu kullanarak bir iletideki üstbilgilerin üzerine atlama
Bu örnekte, MQHeaderIterator ' un skipHeaders() yöntemi, ileti okuma imlecini son üstbilginin hemen ardından konumlandırıyor.

```
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 ();
```

Neden kodunun IBM MQ classes for Java kullanılarak gönderilmeyen bir iletide bulunması
Bu örnekte, okuma yöntemi iletiden okuyarak MQDLH nesnesine veri yerleştirir. Okuma işleminden sonra, ileti okuma imleci MQDLH üstbilgi içeriğinden hemen sonra konumlandırılır.

Kuyruk yöneticisinin teslim edilmeyen ileti kuyruğundaki iletilerin başına bir teslim edilmeyen harf üstbilgisi (MQDLH) eklenir. Bu iletilerin nasıl işleneceğine karar vermek için (örneğin, iletilerin yeniden denenip denenmeyeceğini ya da atılıp atılmayacağını saptamak için), bir kullanılmayan harf işleme uygulamasının MQDLH ' de bulunan neden koduna bakması gerekir.

```
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 ());
```

Tüm üstbilgi sınıfları, kendilerini doğrudan iletiden tek bir adımda kullanıma hazırlamak için uygun bir oluşturucu da sağlar. Bu örnekteki kod aşağıdaki gibi basitleştirilebilir:

```
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 ());
```

IBM MQ classes for Java komutunu kullanarak bir gönderilmeyen ileti üstbilgisini okuma ve kaldırma
Bu örnekte, MQDLH, üstbilgiyi bir kullanılmayan harf iletisinden kaldırmak için kullanılır.

Bir gitmeyen harf işleme uygulaması, neden kodlarının geçici bir hata göstermesi durumunda genellikle reddedilen iletileri yeniden gönderir. İletiyi yeniden sunmadan önce, MQDLH üstbilgisini kaldırmalıdır.

Bu örnek aşağıdaki adımları gerçekleştirir (örnek koddaki açıklamalara bakın):

1. MQHeaderList iletinin tamamını okur ve iletide karşılaşılan her üstbilgi, listedeki bir öğe olur.
2. Gönderilmeyen iletiler ilk üstbilgi olarak bir MQDLH içerir; bu nedenle, üstbilgi listesinin ilk öğesinde bulunabilir. MQHeaderList oluşturulduğunda iletiden MQDLH ' ye veri yerleştirildiği için okuma yönteminin çağrılması gerekmez.
3. Neden kodu, MQDLH sınıfı tarafından sağlanan getReason() yöntemi kullanılarak çıkarılır.
4. Neden kodu incelendi ve iletinin yeniden gönderilmesinin uygun olduğunu gösteriyor. MQDLH, MQHeaderList remove () yöntemi kullanılarak kaldırılır.
5. MQHeaderList , kalan içeriğini yeni bir ileti nesnesine yazar. Yeni ileti artık MQDLH dışında özgün iletideki her şeyi içeriyor ve bir kuyruğa yazılabilir. Oluşturucuya ve yazma yöntemine ilişkin **true** bağımsız değişkeni, ileti gövdesinin MQHeaderListiçinde tutulabileceğini ve yeniden yazılacağını gösterir.
6. Yeni iletinin ileti tanımlayıcısındaki biçim alanı artık daha önce MQDLH biçim alanında bulunan değeri içeriyor. İleti verileri, ileti tanımlayıcısında belirlenen sayısal kodlama ve CCSID ile eşleşir.

```
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.
```

IBM MQ classes for Java komutunu kullanarak bir iletinin içeriğini yazdırma
Bu örnek, üstbilgileri de içinde olmak üzere bir iletinin içeriğini yazdırmak için MQHeaderList öğesini kullanır.

Çıkış, iletinin gövdesinin yanı sıra tüm üstbilgi içeriğinin bir görünümünü de içerir. MQHeaderList sınıfı tek seferde tüm üstbilgilerin kodunu çözerken, MQHeaderIterator bu üstbilgileri uygulama denetimi altında bir kerede birer birer çözer. WebSphere MQ uygulamaları yazarken basit bir hata ayıklama aracı sağlamak için bu tekniği kullanabilirsiniz.

```
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));
```

Bu örnek, MQMD sınıfını kullanarak ileti tanımlayıcı alanlarını da yazdırır. com.ibm.mq.headers.MQMD sınıfının copyFrom() yöntemi, üstbilgi nesnesini ileti gövdesini okumak yerine MQMessage ileti tanımlayıcısı alanlarından doldurur.

```
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));
```

IBM MQ classes for Java komutunu kullanarak iletide belirli bir üstbilgi tipini bulma
Bu örnek, varsa, bir iletide MQRFH2 üstbilgisini bulmak için MQHeaderList ' in indexOf(String) yöntemini kullanır.

```
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);
        ...
}
```

IBM MQ classes for Java kullanarak MQRFH2 üstbilgisini çözümleme
Bu örnek, MQRFH2 sınıfını kullanarak, adlandırılmış bir klasördeki bilinen bir alan değerine nasıl erişileceğini gösterir.

MQRFH2 sınıfı, yalnızca yapının sabit bölümündeki alanlara değil, NameValueVeri alanı içinde taşınan XML kodlu klasör içeriğine de erişmenin çeşitli yollarını sağlar. Bu örnek, adlandırılmış bir klasördeki bilinen bir alan değerine nasıl erişileceğini gösterir; bu örnekte, bir MQ JMS iletisindeki yanıt kuyruğu adını gösteren jms klasöründeki Rto alanı gösterilir.

```
MQRFH2 rfh = ...

String value = rfh.getStringFieldValue ("jms", "Rto");
```

Bir MQRFH2 dosyasının içeriğini keşfetmek için (doğrudan belirli alanları istemenin tersine), alanları ve diğer klasörleri içerebilecek bir klasörün yapısını gösteren MQRFH2.Elementlistesini döndürmek için getFolders yöntemini kullanabilirsiniz. Bir alan ya da klasör boş değere ayarlandığında, alan ya da klasör MQRFH2' den kaldırılır. NameValueVeri klasörü içeriğini bu şekilde değiştirdiğinizde, StrucLength alanı otomatik olarak güncellenir.

IBM MQ classes for Java kullanan MQMessage nesneleri dışındaki byte akışlarının okunması ve yazılması
Bu örnekler, veri kaynağı bir MQMessage nesnesi olmadığında IBM MQ üstbilgi içeriğini ayrıştırmak ve işlemek için üstbilgi sınıflarını kullanır.

Veri kaynağı bir MQMessage nesnesi dışında bir şey olsa da IBM MQ üstbilgi içeriğini ayrıştırmak ve işlemek için üstbilgi sınıflarını kullanabilirsiniz. Her üstbilgi sınıfı tarafından uygulanan MQHeader arabirimi `int read (java.io.DataInput message, int encoding, int characterSet)` ve `int write (java.io.DataOutput message, int encoding, int characterSet)` yöntemlerini sağlar. com.ibm.mq.MQMessage sınıfı, java.io.DataInput ve java.io.DataOutput arabirimlerini uygular. Bu, ileti tanımlayıcısında belirtilen kodlamayı ve CCSID ' yi geçersiz kılarak MQMessage içeriğini okumak ve yazmak için iki MQHeader yöntemini kullanabileceğiniz anlamına gelir. Bu, farklı kodlamalı bir üstbilgi zinciri içeren iletiler için kullanışlıdır.

DataInput ve DataOutput nesnelerini, örneğin, dosya ya da yuva akışları ya da JMS iletilerinde taşınan bayt dizileri gibi diğer veri akışlarından da alabilirsiniz. java.io.DataInputStream sınıfları DataInput ve java.io.DataOutputStream sınıfları DataOutputsınıfını uygular. Bu örnek, bir bayt dizisinden IBM MQ üstbilgi içeriğini okur:

```
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);
```

MQHeaderIterator ile başlayan satır değiştirilebilir

```
MQDLH dlh = new MQDLH (in, CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
// or any other header type
```

Bu örnek, DataOutputAkımını kullanarak bir bayt dizisine yazar:

```
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 ();
```

Bu şekilde akışlarla çalışırken, kodlama ve characterSet bağımsız değişkenleri için doğru değerleri kullanmaya dikkat edin. Üstbilgileri okurken, bayt içeriğinin ilk yazıldığı kodlamayı ve CCSID ' yi belirleyin. Üstbilgileri yazarken, üretmek istediğiniz kodlamayı ve CCSID ' yi belirleyin. Veri dönüştürme, üstbilgi sınıfları tarafından otomatik olarak gerçekleştirilir.

IBM MQ classes for Java kullanarak yeni üstbilgi tipleri için sınıf yaratılması
IBM MQ classes for Javaile birlikte sağlanmayan üstbilgi tipleri için Java sınıfları oluşturabilirsiniz.

IBM MQ classes for Javaile verilen herhangi bir üstbilgi sınıfıyla aynı şekilde kullanabileceğiniz yeni bir üstbilgi tipini gösteren bir Java sınıfı eklemek için, MQHeader arabirimini gerçekleştiren bir sınıf yaratırsınız. En basit yaklaşım, com.ibm.mq.headers.impl.Header sınıfını genişletmektir. Bu örnek, MQTM üstbilgi yapısını gösteren tam işlevsel bir sınıf üretir. Her alan için tek tek alıcı ve ayarlayıcı yöntemleri eklemenize gerek yoktur, ancak üstbilgi sınıfı kullanıcıları için kullanışlıdır. Alan adı için dizgi alan soysal getValue ve setValue yöntemleri, üstbilgi tipinde tanımlanan tüm alanlar için çalışır. Devralınan okuma, yazma ve boyut yöntemleri, yeni üstbilgi tipinin örneklerinin okunmasını ve yazılmasını sağlar ve üstbilgi boyutunu alan tanımına dayalı olarak doğru şekilde hesaplar. Tip tanımlaması yalnızca bir kez yaratılır, ancak bu üstbilgi sınıfının birçok eşgörünümü yaratılır. Yeni üstbilgi tanımlamasını MQHeaderIterator ya da MQHeaderList sınıflarını kullanarak kod çözme için MQHeaderRegistryöğesini kullanarak kaydettirin. Ancak, MQTM üstbilgi sınıfının gerçekte bu pakette belirtildiğini ve varsayılan kayıtta kayıtlı olduğunu unutmayın.

```
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.
}
```

IBM MQ classes for Java ile PCF iletilerinin işlenmesi

Java sınıfları, PCF yapılandırılmış iletileri yaratmak ve ayrıştırmak, PCF isteklerinin gönderilmesini ve PCF yanıtlarının toplanmasını kolaylaştırmak için sağlanır.

PCFMessage ve MQCFGR sınıfları, PCF değiştirge yapılarının dizilerini gösterir. Bunlar, PCF parametrelerinin eklenmesi ve alınması için uygun yöntemler sağlar.

PCF değiştirge yapıları MQCFH, MQCFIN, MQCFIN64, MQCFST, MQCFBS, MQCFIL, MQCFIL64 MQCFSL ve MQCFGR sınıflarıyla gösterilir. Bu arabirimler temel işletim arabirimlerini paylaşır:

- İleti içeriğini okuma ve yazma yöntemleri: read (), write () ve size ()
- Parametreleri işleme yöntemleri: getValue (), setValue (), getParameter () ve diğerleri
- PCF içeriğini bir MQMessage içinde ayrıtan sıralı değer listesi (enumerator) yöntemi.nextParameter ()

PCF süzgeç parametresi, bir süzgeç işlevi sağlamak için sorgunun komutlarında kullanılır. Şu sınıflarda kapsüllenmiş:

- MQCFIF-tamsayı süzgeci
- MQCFSF-dizgi süzgeci
- MQCFBF-byte süzgeci

Bir Kuyruk Yöneticisi, komut sunucusu kuyruğu ve ilişkili bir yanıt kuyruğu bağlantısını yönetmek için iki aracı sınıfı (PCFAgent ve PCFMessageAgent) sağlanır. PCFMessageAgent , PCFAgent 'ı genişletir ve normalde bunun tercih edilmesinde kullanılmalıdır. PCFMessageAgent sınıfı, alınan MQMessages ' i dönüştürür ve bunları çağırana PCFMessage dizisi olarak geri iletir. PCFAgent, kullanmadan önce ayrıştırmanız gereken bir MQMessages dizisi döndürür.

IBM MQ classes for Java içindeki ileti özelliklerini işleme

İleti tanıtıcılarını işlemek için yapılan işlev çağrılarının IBM MQ classes for Javaiçinde eşdeğeri yoktur. İleti tanıtıcısı özelliklerini ayarlamak, döndürmek ya da silmek için MQMessage sınıfının yöntemlerini kullanın.

İleti özellikleriyle ilgili genel bilgi için bkz. “Özellik adları” sayfa 26.

IBM MQ classes for Java içinde iletilere erişim, MQMessage sınıfı aracılığıyla olur. Bu nedenle, Java ortamında ileti tanıtıcıları sağlanmaz ve IBM MQ işlevinin MQCRTMH, MQDLTMH, MQMHBUF ve MQBUFMH işlev çağrılarına eşdeğeri yoktur.

Yordamsal arabirimde ileti tanıtıcısı özelliklerini ayarlamak için MQSETMP çağrısını kullanın. IBM MQ classes for Javaiçinde, MQMessage sınıfının uygun yöntemini kullanın:

- setBooleanÖzelliği
- setByteÖzelliği
- setBytesÖzelliği
- setShortÖzelliği
- setIntÖzelliği
- setInt2Property
- setInt4Property
- setInt8Property
- setLongÖzelliği
- setFloatÖzelliği
- setDoubleÖzelliği
- setStringÖzelliği

- setObjectÖzelliği

Bunlara bazen toplu olarak *set*property* yöntemleri denir.

Yordamsal arabirimde ileti tanıtıcısı özelliklerinin değerini döndürmek için MQINQMP çağrısını kullanın. IBM MQ classes for Javaiçinde, MQMessage sınıfının uygun yöntemini kullanın:

- getBooleanÖzelliği
- getByteÖzelliği
- getBytesÖzelliği
- getShortÖzelliği
- getIntÖzelliği
- getInt2Property
- getInt4Property
- getInt8Property
- getLongÖzelliği
- getFloatÖzelliği
- getDoubleÖzelliği
- getStringÖzelliği
- getObjectÖzelliği

Bunlara bazen toplu olarak *get*property* yöntemleri denir.

Yordamsal arabirimdeki ileti tanıtıcısı özelliklerinin değerini silmek için MQDLTMP çağrısını kullanın. IBM MQ classes for Javaiçinde MQMessage sınıfının deleteProperty yöntemini kullanın.

IBM MQ classes for Java içindeki hataların işlenmesi

Java `try` ve `catch` bloklarını kullanarak IBM MQ classes for Java ' den kaynaklanan hataları işleyin.

Java arabirimindeki yöntemler bir tamamlanma kodu ve neden kodu döndürmez. Bunun yerine, bir IBM MQ çağrısından kaynaklanan tamamlanma kodu ve neden kodu sıfır değilse bir kural dışı durum oluşur. Bu, IBM MQ' e her çağrıdan sonra dönüş kodlarını denetlemeniz gerekmemesi için program mantığını basitleştirir. Programınızda hangi noktalarda başarısızlık olasılığıyla başa çıkmak istediğinize karar verebilirsiniz. Bu noktalarda, kodunuzu aşağıdaki örnekte olduğu gibi `try` ve `catch` bloklarıyla çevreleyebilirsiniz:

```
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() );
}
```

z/OS için Java kural dışı durumlarında bildirilen IBM MQ çağrı neden kodları API tamamlama ve neden kodlarıiçinde belgelenmiştir.

Bir IBM MQ classes for Java uygulaması çalışırken yayınlanan kural dışı durumlar da günlüğe yazılır. Ancak bir uygulama, belirli bir neden koduyla ilişkili kural dışı durumların günlüğe kaydedilmesini önlemek için MQException.logExclude() yöntemini çağırabilir. Bunu, belirli bir neden koduyla ilişkili birçok kural dışı durumun yayınlanmasını beklediğiniz ve günlüğün bu kural dışı durumlarla doldurulmasını istemediğiniz durumlarda yapmak isteyebilirsiniz. Örneğin, uygulamanız bir döngü etrafında her yinelendiğinde bir kuyruktan ileti almayı denerse ve bu girişimlerin çoğu için kuyrukta uygun bir ileti olmasını beklemezseniz, MQRC_NO_MSG_AVAILABLE neden koduyla ilişkili kural dışı durumların günlüğe kaydedilmesini önlemek

isteyebilirsiniz. Bir uygulama daha önce belirli bir neden koduyla ilişkili kural dışı durumların günlüğe kaydedilmesini önlediyse, MQException.logInclude() yöntemi çağrılarak bu kural dışı durumların yeniden günlüğe kaydedilmesine izin verebilir.

Bazen neden kodu, hatayla ilişkili tüm ayrıntıları iletmez. Yayınlanan her kural dışı durum için, bir uygulamanın bağlantılı kural dışı durumu denetlemesi gerekir. Bağlantılı kural dışı durumun kendisi başka bir bağlantılı kural dışı duruma sahip olabilir ve bağlantılı kural dışı durumlar, özgün temel soruna geri dönen bir zincir oluşturur. Bağlantılı bir kural dışı durum, java.lang.Throwable sınıfının zincirleme kural dışı durum düzeneği kullanılarak gerçekleştirilir ve bir uygulama Throwable.getCause() yöntemini çağırarak bağlantılı bir kural dışı durum alır. MQException 'ın eşgörünümü olan bir kural dışı durumdan, MQException.getCause(), com.ibm.mq.jmqi.JmqiException'ın temel eşgörünümünü alır ve bu kural dışı durumdan getCause , hataya neden olan temel java.lang.Exception ' ı alır.

IBM MQ classes for Java içinde öznitelik değerlerini alma ve ayarlama

Birçok ortak öznitelik için getXXX() ve setXXX() yöntemleri sağlanır. Diğerlerine soysal inquire () ve set () yöntemleri kullanılarak erişilebilir.

Sık kullanılan özniteliklerin çoğu için MQManagedObject, MQDestination, MQQueue, MQTopic, MQProcess ve MQQueueManager sınıfları getXXX() ve setXXX() yöntemlerini içerir. Bu yöntemler, öznitelik değerlerini almanızı ve ayarlamanızı sağlar. MQDestination, MQQueue ve MQTopic için, yöntemlerin yalnızca nesneyi açtığınızda uygun sorguyu belirttiğinizde ve işaretleri ayarladığınızda işe yaradığını unutmayın.

Daha az yaygın olan öznitelikler için, MQQueueManager, MQDestination, MQQueue, MQTopic ve MQProcess sınıflarının tümü, MQManagedObjectadlı bir sınıftan edinilir. Bu sınıf, inquire () ve set () arabirimlerini tanımlar.

Yeni işlecini kullanarak yeni bir kuyruk yöneticisi nesnesi yarattığınızda, nesne sorgu için otomatik olarak açılır. Bir süreç nesnesine erişmek için accessProcess() yöntemini kullandığınızda, o nesne sorma için otomatik olarak açılır. Bir kuyruk nesnesine erişmek için accessQueue() yöntemini kullandığınızda, bu nesne sorma ya da ayarlama işlemleri için otomatik olarak açılmaz. Bunun nedeni, bu seçeneklerin otomatik olarak eklenmesinin bazı uzak kuyruk tiplerinde sorunlara neden olmasıdır. Bir kuyrukta inquire, set, getXXXve setXXX yöntemlerini kullanmak için, accessQueue() yönteminin openOptions parametresinde uygun sorgulamayı belirtmeniz ve işaretleri ayarlamanız gerekir. Aynısı hedef ve konu nesneleri için de geçerlidir.

Sorma ve ayarlama yöntemleri üç parametre alır:

- seçiciler dizisi
- intAttrs dizisi
- charAttrs dizisi

Java içindeki bir dizinin uzunluğu her zaman bilindiğinden, MQINQ ' da bulunan SelectorCount, IntAttrCount ve CharAttrUzunluk parametrelerine gerek yoktur. Aşağıdaki örnek, bir kuyrukta nasıl sorgu yapılacağını göstermektedir:

```
// 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));
```

Java içinde çok parçacıklı programlar

Java yürütme ortamı, doğal olarak çoklu iş parçacıklıdır. IBM MQ classes for Java , bir kuyruk yöneticisi nesnesinin birden çok iş parçacığı tarafından paylaşılmasını sağlar, ancak hedef kuyruk yöneticisine tüm erişimin eşitlenmesini sağlar.

Çoklu iş parçacıklı programların Javaiçinde önlenmesi zordur. Bir kuyruk yöneticisine bağlanan ve başlatma sırasında bir kuyruk açan basit bir program düşünün. Program ekranda tek bir düğme görüntüler. Bir kullanıcı bu düğmeyi tıklattığında, program kuyruktan bir ileti getirir.

Java yürütme ortamı, doğal olarak çoklu iş parçacıklıdır. Bu nedenle, uygulamanın kullanıma hazırlanması bir iş parçacığında gerçekleşir ve düğmeye yanıt olarak yürütülen kod ayrı bir iş parçacığında (kullanıcı arabirimi iş parçacığı) yürütülür.

C tabanlı IBM MQ MQI clientile, birden çok iş parçacığı tarafından tanıtıcıların paylaşılmasıyla ilgili sınırlamalar olduğundan, bu bir soruna neden olur. IBM MQ classes for Java , bir kuyruk yöneticisi nesnesinin (ve ilişkili kuyruğunun, konu ve işlem nesnelerinin) birden çok iş parçacığı tarafından paylaşılmasına izin vererek bu kısıtlamayı gevşetir.

IBM MQ classes for Java somutlaması, belirli bir bağlantı (MQQueueManager nesne eşgörünümü) için, hedef IBM MQ kuyruk yöneticisine tüm erişimin uyumlulaştırılmasını sağlar. Bir kuyruk yöneticisine çağrı yapmak isteyen bir iş parçacığı, o bağlantı için devam eden diğer tüm çağrılar tamamlanıncaya kadar engellenir. Programınızdaki birden çok iş parçacığından aynı kuyruk yöneticisine eşzamanlı erişim gerekiyorsa, eşzamanlı erişim gerektiren her iş parçacığı için yeni bir MQQueueManager nesnesi yaratın. (Bu, her iş parçacığı için ayrı bir MQCONN çağrısı yayınlanmasına eşdeğerdir.)

Not: `com.ibm.mq.MQGetMessageOptions` sınıfının somut örnekleri, eşzamanlı olarak ileti isteyen iş parçacıkları arasında paylaşılmamalıdır. Bu sınıfın eşgörünümleri, ilgili MQGET isteği sırasında verilerle güncellenir; bu, nesnenin aynı eşgörünümünde birden çok iş parçacığı eşzamanlı olarak çalışıyorsa beklenmeyen sonuçlara neden olabilir.

IBM MQ classes for Java içinde kanal çıkışlarının kullanılması

IBM MQ classes for Javakullanılarak bir uygulamada kanal çıkışlarının nasıl kullanılacağına ilişkin genel bakış.

Aşağıdaki konularda, Javaiçinde bir kanal çıkışının nasıl yazılacağı, nasıl atanacağı ve verilere nasıl aktarılacağı açıklanmaktadır. Daha sonra C ile yazılan kanal çıkışlarının nasıl kullanılacağını ve bir kanal çıkışları dizisinin nasıl kullanılacağını açıklar.

Uygulamanızın kanal çıkış sınıfını yüklemek için doğru güvenlik iznine sahip olması gerekir.

IBM MQ classes for Java içinde kanal çıkışı oluşturma

Uygun bir arabirim uygulayan bir Java sınıfı tanımlayarak kendi kanal çıkışlarınızı sağlayabilirsiniz.

Bir çıkış gerçekleştirmek için, uygun arabirimi uygulayan yeni bir Java sınıfı tanımlarsınız. com.ibm.mq.exits paketinde üç çıkış arabirimi tanımlanır:

- WMQSendExit
- WMQReceiveExit
- WMQSecurityExit

Not: Kanal çıkışları yalnızca istemci bağlantıları için desteklenir; bağ tanımlama bağlantıları için desteklenmez. Örneğin, C dilinde yazılmış bir istemci uygulaması kullanıyorsanız, Java kanal çıkışı IBM MQ classes for Javakullanamazsınız.

Bir bağlantı için tanımlanan TLS şifrelemesi *bundan sonra* gönderme ve güvenlik çıkışları çağrıldı. Benzer şekilde, *alma ve güvenlik çıkışları çağrılmadan* önce şifre çözme gerçekleştirilir.

Aşağıdaki örnek, üç arabirimi de gerçekleştiren bir sınıfı tanımlar:

```
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
  }
}
```

Her çıkışa bir MQCXP nesnesi ve bir MQCD nesnesi geçirilir. Bu nesneler, yordamsal arabirimde tanımlı MQCXP ve MQCD yapılarını gösterir.

Yazdığınız her çıkış sınıfının bir oluşturucusu olmalıdır. Bu, varsayılan oluşturucu ya da dizgi bağımsız değişkeni alan bir oluşturucu olabilir. Bir dizgi gerekiyorsa, kullanıcı verileri yaratıldığında çıkış sınıfına geçirilir. Çıkış sınıfı hem varsayılan bir oluşturucu hem de tek bir bağımsız değişken oluşturucu içeriyorsa, tek bağımsız değişken oluşturucusunun önceliği vardır.

Gönderme ve güvenlik çıkışları için, çıkış kodunuzun sunucuya göndermek istediğiniz verileri döndürmesi gerekir. Alma çıkışı için, çıkış kodunuz IBM MQ ' un yorumlamasını istediğiniz değiştirilmiş verileri döndürmelidir.

Olası en basit çıkış gövdesi:

```
{ return agentBuffer; }
```

Kanal çıkışından kuyruk yöneticisini kapatmayın.

Var olan kanal çıkış sınıflarının kullanılması

In versions of IBM MQ earlier than 7.0, you would implement these exits using the interfaces MQSendExit, MQReceiveExit, and MQSecurityExit, as in the following example. Bu yöntem geçerli olmaya devam eder, ancak yeni yöntem daha iyi işlevsellik ve performans için tercih edilir.

```
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
  }
}
```

IBM MQ classes for Java içinde bir kanal çıkışı atama
IBM MQ classes for Javakomutunu kullanarak bir kanal çıkışı atayabilirsiniz.

IBM MQ classes for Javaiçindeki IBM MQ kanalının doğrudan eşdeğeri yoktur. Kanal çıkışları bir MQQueueManager' a atanır. Örneğin, WMQSecurityExit arabirimini gerçekleştiren bir sınıf tanımladıktan sonra, bir uygulama güvenlik çıkışını şu dört yoldan biriyle kullanabilir:

- MQQueueManager nesnesi yaratmadan önce MQEnvironment.channelSecurityExit alanına sınıfın bir eşgörünümünü atayarak
- MQEnvironment.channelSecurityExit alanını, MQQueueManager nesnesi yaratmadan önce güvenlik çıkışı sınıfını gösteren bir dizgiye ayarlayarak
- CMQC.SECURITY_EXIT_PROPERTY anahtarıyla MQQueueManager ' a geçirilen özellikler içinde bir anahtar/değer çifti yaratarak
- İstemci kanal tanımlama çizelgesinin (CCDT) kullanılması

MQEnvironment.channelSecurityExit alanını bir dizgiye ayarlayarak, hashtable özelliklerinde bir anahtar/değer çifti yaratarak ya da CCDT kullanarak atanan herhangi bir çıkış, varsayılan bir oluşturucuyla yazılmalıdır. Bir sınıfın somut örneği olarak atanan bir çıkış, uygulamaya bağlı olarak varsayılan bir oluşturucuya ihtiyaç duymaz.

Bir uygulama, benzer bir gönderme ya da alma çıkışını kullanabilir. Örneğin, aşağıdaki kod parçası, daha önce MQEnvironment kullanılarak tanımlanan MyMQExitssınıfında uygulanan güvenlik, gönderme ve alma çıkışlarının nasıl kullanılacağını gösterir:

```
MyMQExits myexits = new MyMQExits();
MQEnvironment.channelSecurityExit = myexits;
MQEnvironment.channelSendExit = myexits;
MQEnvironment.channelReceiveExit = myexits;
:
MQQueueManager jupiter = new MQQueueManager("JUPITER");
```

Bir kanal çıkışını atamak için birden çok yöntem kullanılırsa, öncelik sırası aşağıdaki gibidir:

1. Bir CCDT 'nin URL adresi MQQueueManager' a geçirilirse, CCDT içeriği kullanılacak kanal çıkışlarını belirler ve MQEnvironment ya da özellikler çizelgesindeki çıkış tanımlamaları yoksayılır.
2. CCDT URL geçirilmezse, MQEnvironment 'tan çıkış tanımlamaları ve hashtable birleştirilir
   - MQEnvironment ve Hashtable içinde aynı çıkış tipi tanımlandıysa, Hashtable 'daki tanımlama kullanılır.
   - Eşdeğer eski ve yeni çıkış tipleri belirtilirse (örneğin, yalnızca IBM WebSphere MQ 7.0öncesi sürümlerde kullanılan çıkış tipi için kullanılabilen sendExit alanı ve herhangi bir gönderme çıkışı için kullanılabilen channelSendÇıkış alanı), eski çıkış yerine yeni çıkış (channelSendExit) kullanılır.

Bir kanal çıkışını dize olarak bildirdiyseniz, kanal çıkış programını bulmak için IBM MQ ' i etkinleştirmeniz gerekir. Bunu, uygulamanın çalıştığı ortama ve kanal çıkış programlarının nasıl paketlendiğine bağlı olarak çeşitli şekillerde yapabilirsiniz.

- Uygulama sunucusunda çalışan bir uygulama için, dosyaları Çizelge 58 sayfa 379 içinde gösterilen dizinde ya da **exitClasspath**tarafından başvurulan JAR dosyalarında paketlenmiş olarak saklamanız gerekir.
- Uygulama sunucusunda çalışmayan bir uygulama için aşağıdaki kurallar geçerlidir:
  - Kanal çıkış sınıflarınız ayrı JAR dosyalarında paketlenmişse, bu JAR dosyaları **exitClasspath**içinde yer almalıdır.
  - Kanal çıkış sınıflarınız JAR dosyalarında paketlenmediyse, sınıf dosyaları Çizelge 58 sayfa 379 dizininde gösterilen dizinde ya da JVM sistem sınıf yolundaki herhangi bir dizinde ya da **exitClasspath**dizininde saklanabilir.

exitClasspath özelliği dört şekilde belirtilebilir. Öncelik sırasına göre, bu yollar aşağıdaki gibidir:

1. com.ibm.mq.exitClasspath sistem özelliği (`-D` seçeneği kullanılarak komut satırında tanımlanır)
2. `mqclient.ini` dosyasının exitPath kısmı
3. CMQC.EXIT_CLASSPATH_PROPERTY
4. MQEnvironment değişkeni **exitClasspath**

Birden çok yolu java.io.File.pathSeparator karakterini kullanarak ayırın.

Çizelge 58. Kanal çıkış programlarına ilişkin dizin

Hizmet olarak sunulan	**Dizin**
AIX AIX Linux Linux	`/var/mqm/exits` (32 bit kanal çıkış programları) `/var/mqm/exits64` (64 bit kanal çıkış programları)
Windows	*kuruluş_veri_dizini*\çıkışları

Not: *install_data_dir* , kuruluş sırasında IBM MQ veri dosyaları için seçtiğiniz dizindir. Varsayılan dizin `C:\ProgramData\IBM\MQ`dizinidir.

IBM MQ classes for Java içinde kanal çıkışlarına veri aktarma
Kanal çıkışlarına veri aktarabilir ve kanal çıkışlarından uygulamanıza veri döndürebilirsiniz.

agentBuffer parametresi

Gönderme çıkışı için, *agentBuffer* parametresi gönderilmek üzere olan verileri içerir. Bir alma çıkışı ya da güvenlik çıkışı için, *agentBuffer* parametresi yeni alınan verileri içerir. agentBuffer.limit () ifadesi dizinin uzunluğunu gösterdiğinden, uzunluk değiştirgesine gerek yoktur.

Gönderme ve güvenlik çıkışları için, çıkış kodunuzun sunucuya göndermek istediğiniz verileri döndürmesi gerekir. Alma çıkışı için, çıkış kodunuz IBM MQ ' un yorumlamasını istediğiniz değiştirilmiş verileri döndürmelidir.

Olası en basit çıkış gövdesi:

```
{ return agentBuffer; }
```

Kanal çıkışları, arka dizisi olan bir arabellekle çağrılır. En iyi başarımı elde etmek için, çıkışın bir arka diziye sahip bir arabellek döndürmesi gerekir.

Kullanıcı verileri

Bir uygulama channelSecurityExit, channelSendExit ya da channelReceiveExit ayarını tanımlayarak bir kuyruk yöneticisine bağlanıyorsa, channelSecurityExitUserData, channelSendExitUserData ya da channelReceiveExitUserVeri alanlarını kullanarak, çağrıldığında 32 baytlık kullanıcı verileri uygun kanal çıkış sınıfına geçirilebilir. Bu kullanıcı verileri kanal çıkış sınıfı tarafından kullanılabilir, ancak çıkış her çağrılışında yenilenir. Bu nedenle, kanal çıkışında kullanıcı verilerinde yapılan değişiklikler kaybedilir. Kanal çıkışındaki verilerde kalıcı değişiklikler yapmak istiyorsanız, MQCXP exitUserAlanını kullanın. Bu alandaki veriler, çıkışın çağrılması arasında tutulur.

Uygulama securityExit, sendExitya da receiveExitdeğerini ayarlarsa, bu kanal çıkış sınıflarına kullanıcı verileri geçirilemez.

Bir uygulama kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, istemci bağlantı kanalı tanımında belirtilen tüm kullanıcı verileri çağrıldığında kanal çıkış sınıflarına geçirilir. İstemci kanal tanımlama çizelgesinin kullanılmasıyla ilgili daha fazla bilgi için bkz. “IBM MQ classes for Java ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 362.

IBM MQ classes for Java ile Java içinde yazılmamış kanal çıkışlarının kullanılması
Bir Java uygulamasından C dilinde yazılmış kanal çıkış programlarını kullanma.

IBM MQiçinde, C dilinde yazılmış bir kanal çıkış programının adını, channelSecurityExit, channelSendExit ya da channelReceiveMQEnvironment nesnesindeki ya da Hashtable özelliklerindeki çıkış alanlarına geçirilen bir String olarak belirtebilirsiniz. Ancak, başka bir dilde yazılmış bir uygulamada Java ile yazılmış bir kanal çıkışını kullanamazsınız.

Çıkış programı adını `library(function)` biçiminde belirtin ve çıkış programının konumunun Çıkış yolukonusunda açıklandığı gibi belirtildiğinden emin olun.

C dilinde bir kanal çıkışının nasıl yazılacağına ilişkin bilgi için bkz. "İleti sistemi kanalları için kanal çıkış programları" sayfa 920.

IBM MQ classes for Java içinde kanal gönderme ya da alma çıkışlarının sırasını kullanma
Bir IBM MQ classes for Java uygulaması, art arda çalıştırılan bir kanal gönderme ya da alma çıkışları sırasını kullanabilir.

Gönderme çıkışları sırasını kullanmak için uygulama, gönderme çıkışlarını içeren bir liste ya da dizgi yaratabilir. Liste kullanılırsa, Liste 'nin her öğesi aşağıdakilerden biri olabilir:

- WMQSendExit arabirimini gerçekleştiren kullanıcı tanımlı bir sınıfın eşgörünümü
- MQSendExit arabirimini gerçekleştiren kullanıcı tanımlı bir sınıfın eşgörünümü (Java içinde yazılmış bir gönderme çıkışı için)
- MQExternalSendçıkış sınıfının eşgörünümü (Java içinde yazılmamış bir gönderme çıkışı için)
- MQSendExitZincir sınıfının eşgörünümü
- Dizgi sınıfının somut örneği

Liste başka bir Liste içeremez.

Uygulama, bir alma çıkışları dizisini benzer bir şekilde kullanabilir.

Bir dizgi kullanılırsa, her biri bir Java sınıfının adı ya da `library(function)` biçimindeki bir C programı olan bir ya da daha çok virgülle ayrılmış çıkış tanımından oluşmalıdır.

Daha sonra uygulama, MQQueueManager nesnesi yaratmadan önce Liste ya da Dizgi nesnesini MQEnvironment.channelSendExit alanına atar.

Çıkışlara aktarılan bilgilerin bağlamı yalnızca çıkışların etki alanı içindedir. Örneğin, bir Java çıkışı ve bir C çıkışı zincirlenmişse, Java çıkışının varlığı C çıkışını etkilemez.

Çıkış zinciri sınıflarının kullanılması

IBM WebSphere MQ 7.0öncesi sürümlerde, çıkış dizilerine izin vermek için iki sınıf sağlanmıştır:

- MQSendExitZinciri, MQSendExit arabirimini gerçekleştirir
- MQReceiveExitZinciri, MQReceiveExit arabirimini gerçekleştiriyor

Bu sınıfların kullanımı geçerli kalır, ancak yeni yöntem tercih edilir. IBM MQ Sınıflar for Java arabirimlerinin kullanılması, uygulamanızın `com.ibm.mq.jar` `com.ibm.mq.exits` paketindeki yeni arabirim kümesi kullanılıyorsa `com.ibm.mq.jar`' a bağımlı olmaya devam ettiği anlamına gelir.

Bir gönderme çıkışları sırasını kullanmak için uygulama, her nesnenin aşağıdakilerden biri olduğu bir nesne listesi oluşturdu:

- MQSendExit arabirimini gerçekleştiren kullanıcı tanımlı bir sınıfın eşgörünümü (Java içinde yazılmış bir gönderme çıkışı için)
- MQExternalSendçıkış sınıfının eşgörünümü (Java içinde yazılmamış bir gönderme çıkışı için)
- MQSendExitZincir sınıfının eşgörünümü

Uygulama, bu nesne listesini oluşturucuda değiştirge olarak geçirerek bir MQSendExitZincir nesnesi yarattı. Daha sonra uygulama, bir MQQueueManager nesnesi yaratmadan önce MQSendExitZincir nesnesini MQEnvironment.sendExit alanına atamıştır.

IBM MQ classes for Java içinde kanal sıkıştırma

Bir kanalda akan verilerin sıkıştırılması, kanalın performansını artırabilir ve ağ trafiğini azaltabilir. IBM MQ classes for Java , IBM MQiçinde yerleşik sıkıştırma işlevini kullanın.

IBM MQile sağlanan işlevi kullanarak, ileti kanallarında ve MQI kanallarında akan verileri sıkıştırabilir ve her iki kanal tipinde de üstbilgi verilerini ve ileti verilerini birbirinden bağımsız olarak sıkıştırabilirsiniz. Varsayılan olarak, kanalda veri sıkıştırılmaz. Kanal sıkıştırmanın IBM MQiçinde nasıl uygulandığı da içinde olmak üzere tam açıklaması için bkz. Veri sıkıştırma (COMPMSG) ve Üstbilgi sıkıştırma (COMPHDR).

IBM MQ classes for Java uygulaması, bir java.util.Collection nesnesi yaratılarak istemci bağlantısında üstbilgi ya da ileti verilerinin sıkıştırılması için kullanılabilecek teknikleri belirtir. Her sıkıştırma tekniği, derlemdeki bir Tamsayı nesnesidir ve uygulamanın derleme nesnesine sıkıştırma tekniklerini ekleme sırası, istemci bağlantısı başlatıldığında sıkıştırma tekniklerinin kuyruk yöneticisiyle kararlaştırıldığı sıradır. Uygulama daha sonra derlemi hdrCompList alanına, üstbilgi verileri için ya da ileti verileri için, MQEnvironment sınıfındaki msgCompList alanına atayabilir. Uygulama hazır olduğunda, bir MQQueueManager nesnesi yaratarak istemci bağlantısını başlatabilir.

Aşağıdaki kod parçaları, açıklanan yaklaşımı göstermektedir. İlk kod parçası, üstbilgi veri sıkıştırmasını nasıl uygulayacağınızı gösterir:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(CMQXC.MQCOMPRESS_SYSTEM));
:
MQEnvironment.hdrCompList = headerComp;
:
MQQueueManager qMgr = new MQQueueManager(QM);
```

İkinci kod parçası, ileti verileri sıkıştırmasını nasıl uygulayacağınızı gösterir:

```
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);
```

İkinci örnekte, sıkıştırma teknikleri istemci bağlantısı başlatıldığında RLE ve ZLIBHIGH sırasıyla kararlaştırılır. Seçilen sıkıştırma tekniği, MQQueueManager nesnesinin geçerlilik süresi boyunca değiştirilemez.

İstemci bağlantısında hem istemci hem de kuyruk yöneticisi tarafından desteklenen üstbilgi ve ileti verilerine ilişkin sıkıştırma teknikleri, bir MQChannelDefinition nesnesinin hdrCompListesinde ve msgCompListe alanlarında bir kanal çıkışına iletilir. Bir istemci bağlantısında üstbilgi ve ileti verilerini sıkıştırmakta kullanılan gerçek teknikler, MQChannelExit nesnesinin CurHdrSıkıştırma ve CurMsgSıkıştırma alanlarında bir kanal çıkışına geçirilir.

Bir istemci bağlantısında sıkıştırma kullanılırsa, herhangi bir kanal alma çıkışları işlendikten sonra herhangi bir kanal gönderme çıkışları işlenmeden ve ayıklanmadan önce veriler sıkıştırılır. Gönderme ve alma çıkışlarına geçirilen veriler bu nedenle sıkıştırılmış durumdadır.

Sıkıştırma tekniklerinin belirtilmesi ve hangi sıkıştırma tekniklerinin kullanılabileceği hakkında daha fazla bilgi için bkz. Class com.ibm.mq.MQEnvironment ve Interface com.ibm.mq.MQC.

IBM MQ classes for Java içinde TCP/IP bağlantısının paylaşılması

Tek bir TCP/IP bağlantısını paylaşmak için birden çok MQI kanalı eşgörünümü yapılabilir.

IBM MQ classes for Javaiçinde, tek bir TCP/IP bağlantısını paylaşabilen etkileşimlerin sayısını denetlemek için MQEnvironment.sharingConversations değişkenini kullanırsınız.

SHARECNV özniteliği, bağlantı paylaşımına yönelik en iyi çalışma yaklaşımıdır. Bu nedenle, IBM MQ classes for Java ile 0 'dan büyük bir SHARECNV değeri kullanıldığında, yeni bir bağlantı isteğinin her zaman önceden oluşturulmuş bir bağlantıyı paylaşacağı garanti edilmez.

IBM MQ classes for Java içinde bağlantı havuzu oluşturma

IBM MQ classes for Java , yedek bağlantıların yeniden kullanılmak üzere havuza yollanmasına izin verir.

IBM MQ classes for Java , IBM MQ kuyruk yöneticilerine birden çok bağlantıyla uğraşan uygulamalar için ek destek sağlar. Bir bağlantı artık gerekli olmadığında, onu yok etmek yerine, havuza gönderebilir ve daha sonra yeniden kullanılabilir. Bu, rasgele kuyruk yöneticilerine dizisel olarak bağlanan uygulamalar ve ara katman yazılımları için önemli bir performans geliştirmesi sağlayabilir.

IBM MQ , varsayılan bir bağlantı havuzu sağlar. Uygulamalar, MQEnvironment sınıfı aracılığıyla belirteçleri kaydettirerek ve kayıttan kaldırarak bu bağlantı havuzunu etkinleştirebilir ya da devre dışı bırakabilir. IBM MQ classes for Java bir MQQueueManager nesnesi oluşturduğunda havuz etkinse, bu varsayılan havuzu arar ve uygun bağlantıları yeniden kullanır. Bir MQQueueManager.disconnect () çağrısı oluştuğunda, temel bağlantı havuza döndürülür.

Diğer bir seçenek olarak, uygulamalar belirli bir kullanım için MQSimpleConnectionManager bağlantı havuzu oluşturabilir. Daha sonra, uygulama bu havuzu bir MQQueueManager nesnesinin oluşturulması sırasında belirtebilir ya da o havuzu varsayılan bağlantı havuzu olarak kullanılmak üzere MQEnvironment 'a geçirebilir.

Bağlantıların çok fazla kaynak kullanmasını önlemek için, MQSimpleConnectionManager nesnesinin işleyebileceği toplam bağlantı sayısını sınırlayabilir ve bağlantı havuzunun boyutunu sınırlayabilirsiniz. Bir JVM içindeki bağlantılar için çakışan talepler varsa, sınırların ayarlanması kullanışlıdır.

Varsayılan olarak, getMaxConnections () yöntemi sıfır değerini döndürür; bu, MQSimpleConnectionManager nesnesinin işleyebileceği bağlantı sayısının sınırı olmadığı anlamına gelir. setMaxConnections () yöntemini kullanarak bir sınır ayarlayabilirsiniz. Bir sınır ayarlarsanız ve sınıra ulaşılırsa, başka bir bağlantı isteği, MQRC_MAX_CONNS_LIMIT_ERIŞILDIĞI bir neden koduyla bir MQException yayınlanmasına neden olabilir.

IBM MQ classes for Java içinde varsayılan bağlantı havuzunun denetlenmesi

Bu örnek, varsayılan bağlantı havuzunun nasıl kullanılacağını gösterir.

MQApp1: örnek uygulamasını göz önünde bulundurun:

```
import com.ibm.mq.*;
public class MQApp1
{
      public static void main(String[] args) throws MQException
      {
         for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
         }
      }
}
```

MQApp1 , komut satırından yerel kuyruk yöneticilerinin bir listesini alır, her birine sırayla bağlanır ve bazı işlemler gerçekleştirir. Ancak, komut satırı aynı kuyruk yöneticisini birçok kez listelerken, yalnızca bir kez bağlanmak ve bu bağlantıyı birçok kez yeniden kullanmak daha verimli olur.

IBM MQ classes for Java , bunu yapmak için kullanabileceğiniz varsayılan bir bağlantı havuzu sağlar. Havuzu etkinleştirmek için MQEnvironment.addConnectionPoolToken() yöntemlerinden birini kullanın. Havuzu geçersiz kılmak için MQEnvironment.removeConnectionPoolToken() kullanın.

Aşağıdaki örnek uygulama (MQApp2), işlevsel olarak MQApp1ile aynıdır, ancak her kuyruk yöneticisine yalnızca bir kez bağlanır.

```
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);

       }
}
```

İlk kalın çizgi, bir MQPoolToken nesnesini MQEnvironment olanağına kaydettirerek varsayılan bağlantı havuzunu etkinleştirir.

MQQueueManager oluşturucusu şimdi bu havuzda uygun bir bağlantı olup olmadığını arar ve yalnızca var olan bir bağlantı bulamazsa kuyruk yöneticisine bağlantı yaratır. qmgr.disconnect() çağrısı, havuz bağlantısını daha sonra yeniden kullanmak üzere döndürür. Bu API çağrıları, örnek uygulama MQApp1ile aynıdır.

Vurgulanan ikinci satır, havuzda saklanan kuyruk yöneticisi bağlantılarını yok eden varsayılan bağlantı havuzunu devre dışı bırakır. Bu önemlidir; tersi durumda, uygulama havuzdaki bir dizi canlı kuyruk yöneticisi bağlantısıyla sona erer. Bu durum, kuyruk yöneticisi günlüklerinde görünecek hatalara neden olabilir.

Bir uygulama bir kuyruk yöneticisine bağlanmak için istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, MQQueueManager oluşturucusu önce uygun bir istemci bağlantı kanalı tanımlaması için çizelgeyi arar. Bir bağlantı bulunursa, oluşturucu kanal için kullanılabilecek bir bağlantı için varsayılan bağlantı havuzunu arar. Oluşturucu havuzda uygun bir bağlantı bulamazsa, istemci kanal tanımlama çizelgesinde sonraki uygun istemci bağlantı kanalı tanımlamasını arar ve daha önce açıklandığı gibi devam eder. Oluşturucu istemci kanal tanımlama çizelgesi aramasını tamamlar ve havuzda uygun bir bağlantı bulamazsa, oluşturucu çizelgeye ilişkin ikinci bir arama başlatır. Bu arama sırasında, oluşturucu her uygun istemci bağlantı kanalı tanımlaması için yeni bir bağlantı yaratmayı dener ve yaratmayı başardığı ilk bağlantıyı kullanır.

Varsayılan bağlantı havuzu en fazla on kullanılmayan bağlantıyı saklar ve kullanılmayan bağlantıları en fazla beş dakika etkin tutar. Uygulama bunu değiştirebilir (ayrıntılar için bkz. “IBM MQ classes for Java içinde farklı bir bağlantı havuzu sağlanması” sayfa 384).

Bir MQPoolTokensağlamak için MQEnvironment kullanmak yerine uygulama kendi oluşturabilir:

```
MQPoolToken token=new MQPoolToken();
MQEnvironment.addConnectionPoolToken(token);
```

Bazı uygulamalar ya da ara katman yazılımı satıcıları, özel bir bağlantı havuzuna bilgi aktarmak için MQPoolToken alt sınıflarını sağlar. Bu bilgiler oluşturulabilir ve addConnectionPoolToken() olanağına bu şekilde geçirilebilirler; böylece bağlantı havuzuna ek bilgiler aktarılabilir.

IBM MQ classes for Java içinde varsayılan bağlantı havuzu ve birden çok bileşen
Bu örnek, kayıtlı MQPoolToken nesnelerinin durağan bir kümesine MQPoolTokens eklenmesini ya da kaldırılmasını gösterir.

MQEnvironment, kayıtlı MQPoolToken nesnelerinin statik bir kümesini içerir. Bu kümeye MQPoolTokens eklemek ya da kaldırmak için aşağıdaki yöntemleri kullanın:

- MQEnvironment.addConnectionPoolToken()
- MQEnvironment.removeConnectionPoolToken()

Bir uygulama, bağımsız olarak var olan ve bir kuyruk yöneticisini kullanarak iş gerçekleştiren birçok bileşenden oluşabilir. Böyle bir uygulamada, her bileşen, geçerlik süresi boyunca MQEnvironment kümesine bir MQPoolToken eklemelidir.

Örneğin, MQApp3 örnek uygulaması on iş parçacığı yaratır ve her birini başlatır. Her iş parçacığı kendi MQPoolToken' ını kaydeder, bir süre bekler ve kuyruk yöneticisine bağlanır. İş parçacığı bağlantıyı kestikten sonra, kendi MQPoolTokenöğesini kaldırır.

MQPoolTokenskümesinde en az bir simge varken varsayılan bağlantı havuzu etkin kalır, bu nedenle bu uygulama süresince etkin kalır. Uygulamanın bir ana nesneyi iş parçacıklarının genel denetiminde tutması gerekmez.

```
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);
      }
}
```

IBM MQ classes for Java içinde farklı bir bağlantı havuzu sağlanması
Bu örnek, farklı bir bağlantı havuzu sağlamak için **com.ibm.mq.MQSimpleConnectionManager** sınıfının nasıl kullanılacağını gösterir.

Bu sınıf, bağlantı havuzlama için temel olanaklar sağlar ve uygulamalar, havuzun davranışını özelleştirmek için bu sınıfı kullanabilir.

Somutlaştırıldıktan sonra, MQQueueManager oluşturucusunda bir MQSimpleConnectionManager belirtilebilir. MQSimpleConnectionManager daha sonra, oluşturulan MQQueueManager' ın altında yatan bağlantıyı yönetir. MQSimpleConnectionManager uygun bir havuza yollanmış bağlantı içeriyorsa, bu bağlantı yeniden kullanılır ve MQQueueManager.disconnect () çağrısından sonra MQSimpleConnectionManager 'a döndürülür.

Aşağıdaki kod parçası bu davranışı gösterir:

```
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);
```

İlk MQQueueManager oluşturucusu sırasında oluşan bağlantı, qmgr.disconnect() çağrısından sonra myConnMan içinde saklanır. Daha sonra, MQQueueManager oluşturucusuna yapılan ikinci çağrı sırasında bağlantı yeniden kullanılır.

İkinci satır MQSimpleConnectionManager 'ı etkinleştirir. Son satır MQSimpleConnectionManager 'ı devre dışı bırakır ve havuzda tutulan bağlantıları yok eder. MQSimpleConnectionManager, varsayılan olarak, bu kısımda daha sonra açıklanan MODE_AUTO ' da bulunur.

MQSimpleConnectionManager, bağlantıları en son kullanılan temelde ayırır ve en son kullanılan temelde bağlantıları yok eder. Varsayılan değer olarak, bağlantı beş dakika boyunca kullanılmamışsa ya da havuzda kullanılmayan ondan fazla bağlantı varsa, bağlantı yok edilir. MQSimpleConnectionManager.setTimeout() yöntemini çağırarak bu değerleri değiştirebilirsiniz.

Ayrıca, MQQueueManager oluşturucusunda bir Bağlantı Yöneticisi sağlanmadığı zaman, varsayılan bağlantı havuzu olarak kullanılmak üzere bir MQSimpleConnectionYöneticisi ayarlayabilirsiniz.

Aşağıdaki uygulama bunu gösterir:

```
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);
      }
}
```

Koyu çizgiler bir MQSimpleConnectionManager nesnesi yaratır ve yapılandırır. Yapılandırma aşağıdakileri yapar:

- Bir saat boyunca kullanılmayan bağlantıları sonlandırır
- myConnMan tarafından yönetilen bağlantı sayısını 75 ile sınırlar
- Havuzdaki kullanılmayan bağlantı sayısını 50 ile sınırlar
- Varsayılan değer olan MODE_AUTO ' yı ayarlar. Bu, havuzun yalnızca varsayılan bağlantı yöneticisi olması ve MQEnvironment tarafından tutulan MQPoolTokens kümesinde en az bir simge olması durumunda etkin olduğu anlamına gelir.

Yeni MQSimpleConnectionManager, varsayılan bağlantı yöneticisi olarak ayarlanır.

Son satırda, uygulama MQApp3.main() öğesini çağırır. Bu, her iş parçacığının IBM MQ bağımsız olarak kullandığı bir dizi iş parçacığını çalıştırır. Bu iş parçacıkları bağlantıları kurarken myConnMan kullanır.

IBM MQ classes for Java kullanılarak JTA/JDBC koordinasyonu

IBM MQ classes for Java , IBM MQ ' in JDBC tip 2 ya da JDBC tip 4 uyumlu sürücü sağlayan bir veritabanı için eşgüdümcü olarak işlev görmesini sağlayan MQQueueManager.begin () yöntemini destekler.

Bu destek tüm platformlarda kullanılamaz. Hangi platformların JDBC koordinasyonunu desteklediğini denetlemek için bkz. IBM MQ için Sistem Gereksinimleri.

XA-JTA desteğini kullanmak için özel JTA anahtar kitaplığını kullanmanız gerekir. Bu kitaplığı kullanma yöntemi, Windows platformunu mu, yoksa diğer platformlardan birini mi kullandığınıza bağlı olarak değişir.

Windows üzerinde JTA/JDBC koordinasyonunun yapılandırılması
XA kitaplığı, `jdbcxxx.dll`biçiminde bir DLL olarak sağlanır.

Sağlanan `jdbcora12.dll` , Windows sunucusu kuruluşu için IBM MQ için Oracle 12Cile uyumluluk sağlar.

Windows sistemlerinde XA kitaplığı tam bir DLL olarak sağlanır. Bu DLL 'nin adı `jdbcxxx.dll` ' dir; burada `xxx` , anahtar kitaplığının derlendiği veritabanını gösterir. Bu kitaplık, IBM MQ classes for Java kuruluşunuzun `java\lib\jdbc` ya da `java\lib64\jdbc` dizininde bulunur. Anahtar yükleme dosyası olarak da açıklanan XA kitaplığını kuyruk yöneticisine bildirmeniz gerekir. IBM MQ Explorerkullanın. XA kaynak yöneticisi altındaki kuyruk yöneticisi özellikleri panosunda anahtar yükleme dosyasının ayrıntılarını belirtin. Yalnızca kitaplığın adını vermelisiniz. Örneğin:

Db2 veritabanı için SwitchFile alanını şu değere ayarlayın: `dbcdb2`

Oracle veritabanı için SwitchFile alanını şu değere ayarlayın: `jdbcora`

Notlar:

1. Oracle 12C , yalnızca Windowsiçin IBM MQ üzerinde IBM MQ classes for Javatarafından desteklenir.
2. Oracle 12C ' nin desteklenen sürümü 12.1.0.1.0 Enterprise Edition ve gelecekteki düzeltme paketleridir.
3. Oracle 64 bit Windows üzerinde 64 bit veritabanları için 32 bit Oracle istemcisi gerekir.
4. IBM MQ classes for Javakomutunu kullanarak IBM MQ , hareket eşgüdümcüsü olarak işlev görür. Ancak JTA tarzı bir işleme katılmak mümkün değildir.

Windows dışındaki platformlarda JTA/JDBC koordinasyonunun yapılandırılması
Nesne dosyaları sağlanır. Sağlanan makefile 'ı kullanarak uygun olanı bağlayın ve yapılanış dosyasını kullanarak kuyruk yöneticisine bildirin.

IBM MQ , her veritabanı yönetim sistemi için iki nesne dosyası sağlar. 32 bitlik bir anahtar kitaplığı oluşturmak için bir nesne dosyasını bağlamanız ve 64 bitlik bir anahtar kitaplığı oluşturmak için diğer nesne dosyasını bağlamanız gerekir. Db2için, her nesne dosyasının adı `jdbcdb2.o`ve Oracle için her nesne dosyasının adı `jdbcora.o`olur.

Her nesne dosyasını IBM MQile sağlanan uygun makefile ile bağlamanız gerekir. Bir anahtar kitaplığı, farklı sistemlerde farklı konumlarda saklanabilecek diğer kitaplıkları gerektirir. Ancak, anahtar kitaplığı `setuid` ortamında çalışan kuyruk yöneticisi tarafından yüklendiği için, anahtar kitaplığı bu kitaplıkları bulmak için kitaplık yolu ortam değişkenini kullanamıyor. Bu nedenle, sağlanan makefile, bir anahtar kitaplığının bu kitaplıkların tam olarak nitelenmiş yol adlarını içermesini sağlar.

Bir anahtar kitaplığı oluşturmak için aşağıdaki biçimde bir **make** komutu girin. 32 bit anahtar kitaplığı oluşturmak için IBM MQ kuruluşunuzun `/java/lib/jdbc` dizinine komutu girin. 64 bit anahtar kitaplığı oluşturmak için komutu `/java/lib64/jdbc` dizinine girin.

```
make DBMS
```

Burada *DBMS* , anahtar kitaplığını yaratmakta olduğunuz veritabanı yönetim sistemidir. Geçerli değerler, Db2 için db2 ve Oracleiçin `oracle` değerleridir.

Not:

- 32 bit uygulamaları çalıştırmak için, kullanmakta olduğunuz her veritabanı yönetim sistemi için hem 32 bit, hem de 64 bit anahtar kitaplığı yaratmanız gerekir. 64 bit uygulamaları çalıştırmak için yalnızca 64 bit anahtar kitaplığı yaratmanız gerekir. Db2için, her anahtar kitaplığının adı `jdbcdb2` ve Oracleiçin her anahtar kitaplığının adı `jdbcora`olur. Makefiles, 32 bit ve 64 bit anahtar kitaplıklarının farklı IBM MQ dizinlerinde saklanmasını sağlar. 32 bit anahtar kitaplığı `/java/lib/jdbc` dizininde saklanır ve 64 bit anahtar kitaplığı `/java/lib64/jdbc` dizininde saklanır.
- Oracle 'i bir sistemde herhangi bir yere kurabileceğiniz için, makefiles uygulaması Oracle ' un kurulu olduğu yeri bulmak için **ORACLE_HOME** ortam değişkenini kullanır.
- IBM MQ varsayılan yerden başka bir yere kurulduysa, makefile 'da **MQ_INSTALLATION_PATH** değerini değiştirin.

Db2, Oracleya da her ikisi için anahtar kitaplıklarını oluşturduktan sonra, bunları kuyruk yöneticinizde bildirmeniz gerekir. Kuyruk yöneticisi yapılanış dosyasında (qm.ini) Db2 ya da Oracle veritabanları için `XAResourceManager` kısmı zaten varsa, her bir bölmede `SwitchFile` girişini aşağıdakilerden biriyle değiştirmeniz gerekir:

Db2 veritabanı için

```
SwitchFile=jdbcdb2
```

Oracle veritabanı için

```
SwitchFile=jdbcora
```

32 bit ya da 64 bit anahtar kitaplığının tam olarak nitelenmiş yol adını belirtmeyin. Yalnızca kitaplığın adını belirleyin.

Kuyruk yöneticisi yapılanış dosyası Db2 ya da Oracle veritabanları için `XAResourceManager` kısmı içermiyorsa ya da ek `XAResourceManager` kısmı eklemek istiyorsanız, `XAResourceManager` kısmı oluşturulmasına ilişkin bilgi için bkz. IBM MQ Yönetimi . Ancak, yeni bir `XAResourceManager` kısmında yer alan her `SwitchFile` girişi, bir Db2 ya da Oracle veritabanı için daha önce açıklandığı gibi olmalıdır. `ThreadOfControl=PROCESS`girişini de eklemelisiniz.

Kuyruk yöneticisi yapılanış dosyasını güncelledikten ve uygun tüm veritabanı ortam değişkenlerinin ayarlandığından emin olduktan sonra kuyruk yöneticisini yeniden başlatabilirsiniz.

JTA/JDBC koordinasyonunu kullanma
API çağrılarınızı sağlanan örnekteki gibi kodlayın.

Bir kullanıcı uygulaması için API çağrılarının temel sırası:

```
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()
```

getJDBCConnection çağrısında `xads` , bağlanılacağı veritabanının ayrıntılarını tanımlayan XADataSource arabiriminin veritabanına özgü bir somutlamasıdır. getJDBCConnection' a geçirilecek uygun bir XADataSource nesnesinin nasıl yaratılacağını belirlemek için veritabanınızın belgelerine bakın.

JDBC çalışmasını gerçekleştirmek için sınıf yolunuzu veritabanına özgü uygun jar dosyalarıyla da güncellemeniz gerekir.

Birden çok veritabanına bağlanmanız gerekiyorsa, hareketi birkaç farklı bağlantıda gerçekleştirmek için getJDBCConnection ' ı birkaç kez çağırmanız gerekir.

getJDBCConnection'ın iki biçimi vardır; bunlar XADataSource.getXAConnection' ın iki biçimini yansıtır:

```
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
```

Bu yöntemler, JTA işlevlerini kullanmayan müşteriler için JVM doğrulayıcısıyla ilgili sorunları önlemek için throws yantümcelerinde Kural Dışı Durum bildiriyor. Yayınlanan gerçek kural dışı durum javax.transaction.xa.XAException ' dır; bu kural dışı durum, jta.jar dosyasının daha önce gerektirmeyen programlar için sınıf yoluna eklenmesini gerektirir.

JTA/JDBC desteğini kullanmak için uygulamanıza aşağıdaki deyimi eklemelisiniz:

```
MQEnvironment.properties.put(CMQC.THREAD_AFFINITY_PROPERTY, new Boolean(true));
```

JTA/JDBC koordinasyonunda bilinen sorunlar ve sınırlamalar
JTA/JDBC desteğinin bazı sorunları ve sınırlamaları, kullanımda olan veritabanı yönetim sistemine bağlıdır; örneğin, test edilen JDBC sürücüleri, bir uygulama çalışırken veritabanı kapatıldığında farklı davranır. Bir uygulamanın kullandığı veritabanıyla bağlantı kesildiyse, uygulamanın kuyruk yöneticisiyle ve veritabanıyla yeni bir bağlantı kurmak için gerçekleştirebileceği adımlar vardır; böylece, gereken işlemsel işi gerçekleştirmek için bu yeni bağlantıları kullanabilir.

JTA/JDBC desteği JDBC sürücülerine çağrı yaptığı için, bu JDBC sürücülerinin uygulanması sistem davranışı üzerinde önemli bir etkiye sahip olabilir. Özellikle, test edilen JDBC sürücüleri, bir uygulama çalışırken veritabanı kapatıldığında farklı davranır.

Önemli: Açık bağlantıları tutan uygulamalar varken, veritabanını her zaman aniden kapatmaktan kaçının.

Not: Bir IBM MQ classes for Java uygulamasının, IBM MQ ' in veritabanı eşgüdümcüsü olarak işlev görmesi için bağ tanımlama kipini kullanarak bağlanması gerekir.

Birden çok XAResourceManager kısmı
Bir kuyruk yöneticisi yapılanış dosyasında (`qm.ini`) birden çok XAResourceManager kısmı kullanılması desteklenmez. İlki dışındaki tüm XAResourceManager kısmı yoksayılır.

Db2

Bazen Db2 bir SQL0805N hatası döndürür. Bu sorun şu CLP komutuyla çözülebilir:

```
DB2 bind @db2cli.lst blocking all grant public
```

Daha fazla bilgi için Db2 belgelerine bakın.

XAResourceManager kısmı, ThreadOfControl=PROCESS kullanacak şekilde yapılandırılmalıdır. Db2 8.1 ve daha yüksek düzeyler için bu, Db2denetim ayarının varsayılan iş parçacığıyla eşleşmez; bu nedenle, XA Open String içinde toc=p belirtilmelidir. JTA/JDBC eşgüdümünü içeren Db2 için örnek bir XAResourceManager kısmı aşağıdaki gibidir:

```
XAResourceManager:
     Name=jdbcdb2
     SwitchFile=jdbcdb2
     XAOpenString=uid=userid,db=dbalias,pwd=password,toc=p
     ThreadOfControl=PROCESS
```

Bu, JTA/JDBC koordinasyonunu kullanan Java uygulamalarının çok iş parçacıklı olmasını önlemez.

Oracle
MQQueueManager.disconnect () bir SQLException oluşturduktan sonra JDBC Connection.close() yönteminin çağrılması. MQQueueManager.disconnect () işleminden önce Connection.close() komutunu arayın ya da Connection.close() çağrıyı atlayın.

Veritabanı bağlantılarıyla ilgili sorunların işlenmesi

Bir IBM MQ classes for Java uygulaması IBM MQtarafından sağlanan JTA/JDBC desteğini kullandığında, genellikle aşağıdaki adımları gerçekleştirir:

1. Hareket yöneticisi olarak işlev görecek kuyruk yöneticisine yönelik bir bağlantıyı gösteren yeni bir MQQueueManager nesnesi yaratır.
2. Harekette yer alacak veritabanına nasıl bağlanılacağına ilişkin ayrıntıları içeren bir XADataSource nesnesi oluşturur.
3. Daha önce yaratılmış olan XADataSource ' u (XADataSource) aktaran MQQueueManager.getJDBCConnection(XADataSource) yöntemini çağırır. Bu, IBM MQ classes for Java ' in veritabanıyla bağlantı kurmasına neden olur.
4. XA hareketini başlatmak için MQQueueManager.begin () yöntemini çağırır.

5. İleti alışverişi ve veritabanı çalışmasını gerçekleştirir.
6. Gerekli işlerin tümü tamamlandığında, MQQueueManager.commit () yöntemini çağırır. Bu işlem XA hareketini tamamlar.
7. Bu noktada yeni bir XA hareketi gerekiyorsa, uygulama 4, 5 ve 6. adımları yineleyebilir.
8. Uygulama tamamlandığında, 3. adımda yaratılan veritabanı bağlantısını kapatmalı ve kuyruk yöneticisiyle bağlantıyı kesmek için MQQueueManager.disconnect () yöntemini çağırmalıdır.

IBM MQ classes for Java , bir uygulama MQQueueManager.getJDBCConnection(XADataSource) öğesini çağırdığında yaratılan tüm veritabanı bağlantılarının iç listesini içerir. Bir kuyruk yöneticisinin XA hareketinin işlenmesi sırasında veritabanıyla iletişim kurması gerekiyorsa, aşağıdaki işlemler gerçekleşir:

1. Kuyruk yöneticisi, veritabanına geçirilmesi gereken XA çağrısının ayrıntılarını aktararak IBM MQ classes for Java' i çağırır.
2. Daha sonra IBM MQ classes for Java , listede uygun bağlantıyı arar ve ardından bu bağlantıyı kullanarak XA çağrısına veri tabanı akışı sağlar.

Bu işlem sırasında herhangi bir noktada veritabanıyla bağlantı kesilirse, uygulama aşağıdaki işlemleri gerçekleştirmelidir:

1. MQQueueManager.backout () yöntemini çağırarak, hareket altında yapılan var olan işleri geri çevirin.
2. Veritabanı bağlantısını kapatın. Bu, IBM MQ classes for Java ' in bozuk veritabanı bağlantısının ayrıntılarını iç listesinden kaldırmasına neden olmalıdır.
3. MQQueueManager.disconnect () yöntemini çağırarak kuyruk yöneticisiyle bağlantıyı kesin.
4. Yeni bir MQQueueManager nesnesi oluşturarak kuyruk yöneticisiyle yeni bir bağlantı kurun.
5. MQQueueManager.getJDBCConnection(XADataSource) yöntemini çağırarak yeni bir veritabanı bağlantısı yaratın.
6. İşlemsel işi yeniden gerçekleştirin.

Bu, uygulamanın kuyruk yöneticisiyle ve veritabanıyla yeni bir bağlantı kurmasını sağlar ve daha sonra, gereken hareket işlerini gerçekleştirmek için bu bağlantıları kullanır.

IBM MQ classes for Java içinde Transport Layer Security (TLS) desteği

IBM MQ classes for Java istemci uygulamaları TLS şifrelemesini destekler. TLS şifrelemesini kullanmak için bir JSSE sağlayıcısı gerekir.

TRANSPORT (CLIENT) kullanan IBM MQ classes for Java istemci uygulamaları TLS şifrelemesini destekler. TLS, iletişim şifrelemesi, kimlik doğrulaması ve ileti bütünlüğü sağlar. Genellikle İnternet üzerindeki ya da bir intranet içindeki herhangi iki eş arasındaki iletişimin güvenliğini sağlamak için kullanılır.

IBM MQ classes for Java , TLS şifrelemesini işlemek için Java Secure Socket Extension (JSSE) kullanır ve bu nedenle bir JSSE sağlayıcısı gerektirir. JSE v1.4 JVM ' lerin yerleşik bir JSSE sağlayıcısı vardır. Sertifikaların nasıl yönetileceği ve saklanacağına ilişkin ayrıntılar, sağlayıcıdan sağlayıcıya farklılık gösterebilir. Bu konuda bilgi için JSSE sağlayıcınızın belgelerine bakın.

Bu bölümde, JSSE sağlayıcınızın doğru olarak kurulduğu ve yapılandırıldığı ve uygun sertifikaların kurulduğu ve JSSE sağlayıcınızın kullanımına sunulduğu varsayılır.

IBM MQ classes for Java istemci uygulamanız bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, bkz. “IBM MQ classes for Java ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 362.

IBM MQ classes for Java içinde TLS ' yi etkinleştirme

TLS ' yi etkinleştirmek için bir CipherSuitebelirtirsiniz. CipherSuitebelirtmenin iki yolu vardır.

TLS yalnızca istemci bağlantıları için desteklenir. TLS 'yi etkinleştirmek için, kuyruk yöneticisiyle iletişim kurarken kullanılacak CipherSuite ' i belirtmeniz gerekir ve bu CipherSuite hedef kanalda ayarlanan CipherSpec ile eşleşmelidir. Ayrıca, CipherSuite adlı ürün grubu JSSE sağlayıcınız tarafından desteklenmelidir. Ancak, CipherSuites CipherSpecs ' den farklıdır ve bu nedenle farklı adlara sahiptir. “IBM MQ classes for Java içinde TLS CipherSpecs ve CipherSuites” sayfa 394 , IBM MQ tarafından desteklenen CipherSpecs ' i JSSE tarafından bilinen eşdeğer CipherSuites ile eşlemek için bir tablo içerir.

TLS 'yi etkinleştirmek için, MQEnvironment 'ın sslCipherSuite statik üye değişkenini kullanarak CipherSuite ' i belirtin. Aşağıdaki örnek, TLS_RSA_WITH_AES_128_CBC_SHA256: CipherSpec ile TLS gerektirecek şekilde ayarlanan SECURE.SVRCONN.CHANNELadlı bir SVRCONN kanalına bağlanır.

```
MQEnvironment.hostname       = "your_hostname";
MQEnvironment.channel        = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.sslCipherSuite = "SSL_RSA_WITH_AES_128_CBC_SHA256";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

Kanalın CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA256değerine sahip olmasına rağmen, Java uygulamasının SSL_RSA_WITH_AES_128_CBC_SHA256 CipherSuite belirtmesi gerekir. CipherSpecs ile CipherSuitesarasındaki eşlemelerin bir listesi için bkz. “IBM MQ classes for Java içinde TLS CipherSpecs ve CipherSuites” sayfa 394 .

Bir uygulama, CMQC.SSL_CIPHER_SUITE_PROPERTYortam özelliğini ayarlayarak bir CipherSuite de belirtebilir.

Diğer bir seçenek olarak, İstemci Kanal Tanımlama Çizelgesi 'ni (CCDT) kullanın. Daha fazla bilgi için bkz. “IBM MQ classes for Java ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 362

IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) tarafından desteklenen bir CipherSuite kullanmak için istemci bağlantısına gereksinim duyarsanız, bir uygulama MQEnvironment sınıfında sslFipsGerekli alanını `true`olarak ayarlayabilir. Alternatif olarak, uygulama CMQC.SSL_FIPS_REQUIRED_PROPERTYortam özelliğini ayarlayabilir. Varsayılan değer, bir istemci bağlantısının IBM MQtarafından desteklenen herhangi bir CipherSuite kullanabileceği anlamına gelen `false`değeridir.

Bir uygulama birden çok istemci bağlantısı kullanıyorsa, uygulama ilk istemci bağlantısını yarattığında kullanılan sslFipsGerekli alanının değeri, uygulama sonraki bir istemci bağlantısı yarattığında kullanılacak değeri belirler. Bu nedenle, uygulama sonraki bir istemci bağlantısı oluşturduğunda, sslFipsGerekli alanının değeri yoksayılır. sslFipsGerekli alanı için farklı bir değer kullanmak istiyorsanız uygulamayı yeniden başlatmanız gerekir.

TLS kullanarak başarıyla bağlanmak için JSSE güvenli deposu, kuyruk yöneticisi tarafından sunulan sertifikanın doğrulanabileceği sertifika yetkilisi kök sertifikalarıyla ayarlanmalıdır. Benzer şekilde, SVRCONN kanalındaki SSLClientAuth MQSSL_CLIENT_AUTH_REQUIRED olarak ayarlandıysa, JSSE anahtar deposu kuyruk yöneticisi tarafından güvenilen bir tanıtıcı sertifika içermelidir.

İlgili başvurular

AIX, Linux, and Windows için Federal Bilgi İşleme Standartları (FIPS)

IBM MQ classes for Java içinde kuyruk yöneticisinin ayırt edici adını kullanma

Kuyruk yöneticisi, ayırt edici adı (DN) içeren bir TLS sertifikasını kullanarak kendisini tanımlar. Bir IBM MQ classes for Java istemci uygulaması, doğru kuyruk yöneticisiyle iletişim kurduğundan emin olmak için bu DN ' yi kullanabilir.

MQEnvironment 'ın sslPeerAd değişkeni kullanılarak bir DN örüntüsü belirtildi. Örneğin, ayar:

```
MQEnvironment.sslPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

Yalnızca kuyruk yöneticisi, QMGR. ile başlayan bir Ortak Adı olan bir sertifika sunarsa bağlantının başarılı olmasını sağlar. ve en az iki Kuruluş Birimi adı; bunlardan ilki IBM ve ikincisi WebSphereolmalıdır.

sslPeerAdı ayarlanırsa, bağlantılar yalnızca geçerli bir kalıba ayarlandıysa ve kuyruk yöneticisi eşleşen bir sertifika sunuyorsa başarılı olur.

Bir uygulama, CMQC.SSL_PEER_NAME_PROPERTYortam özelliğini ayarlayarak kuyruk yöneticisinin ayırt edici adını da belirleyebilir. Ayırt edici adlarla ilgili daha fazla bilgi için bkz. Ayırt edici adlar.

IBM MQ classes for Java içinde sertifika iptal listelerini kullanma

java.security.cert.CertStore sınıfıyla kullanılacak sertifika iptal listelerini belirtin. IBM MQ classes for Java daha sonra sertifikaları belirtilen CRL ile karşılaştırır.

Sertifika iptal listesi (CRL), sertifika yetkilisi ya da yerel kuruluş tarafından iptal edilen bir sertifika kümesidir. CRL ' ler genellikle LDAP sunucularında barındırılır. Java 2 v1.4ile, bağlantı sırasında bir CRL sunucusu belirtilebilir ve bağlantıya izin verilmeden önce kuyruk yöneticisi tarafından sunulan sertifika CRL ile karşılaştırılarak denetlenir. Sertifika iptal listeleri ve IBM MQile ilgili daha fazla bilgi için bkz. Sertifika İptal Listeleri ve Yetki İptal Listeleriyle Çalışma ve CRL 'lere ve ARL' lere IBM MQ classes for Java ve IBM MQ classes for JMSile erişme.

Not: LDAP sunucusunda barındırılan bir CRL ile CertStore ' u başarıyla kullanmak için Java Software Development Kit (SDK) ürününüzü CRL ile uyumlu olduğundan emin olun. Bazı SDK 'lar CRL' nin LDAP v2için bir şema tanımlayan RFC 2587 'ye uymasını gerektirir. LDAP v3 sunucularının çoğu RFC 2256 'yı kullanır.

Kullanılacak CRL ' ler java.security.cert.CertStore sınıfıyla belirtilir. CertStoreeşgörünümlerinin nasıl elde edileceğine ilişkin tüm ayrıntılar için bu sınıfla ilgili belgelere bakın. LDAP sunucusuna dayalı bir CertStore yaratmak için, önce kullanılacak sunucu ve kapı ayarlarıyla kullanıma hazırlanmış bir LDAPCertStoreDeğiştirgeleri somut örneği yaratın. Örneğin:

```
import java.security.cert.*;
CertStoreParameters csp = new LDAPCertStoreParameters("crl_server", 389);
```

CertStoreDeğiştirgeler somut örneği yaratıldıktan sonra, LDAP tipinde bir CertStore yaratmak için CertStore üzerindeki durağan oluşturucuyu kullanın:

```
CertStore cs = CertStore.getInstance("LDAP", csp);
```

Diğer CertStore tipleri (örneğin, Derlem) de desteklenir. Genellikle yedeklilik sağlamak için aynı CRL bilgileriyle ayarlanan birkaç CRL sunucusu vardır. Bu CRL sunucularının her biri için bir CertStore nesnesine sahip olduğunuzda, bunların tümünü uygun bir Derlem içine yerleştirin. Aşağıdaki örnek, ArrayList' ne yerleştirilen CertStore nesnelerini göstermektedir:

```
import java.util.ArrayList;
Collection crls = new ArrayList();
crls.add(cs);
```

Bu derlem, CRL denetimini etkinleştirmek için bağlanmadan önce MQEnvironment durağan değişkenine (sslCert) ayarlanabilir:

```
MQEnvironment.sslCertStores = crls;
```

Bir bağlantı kurulurken kuyruk yöneticisi tarafından sunulan sertifikanın geçerliliği aşağıdaki gibi denetlenir:

1. sslCertmağazaları tarafından tanıtılan derlemdeki ilk CertStore nesnesi, bir CRL sunucusunu tanımlamak için kullanılır.
2. CRL sunucusuyla iletişim kurma girişiminde bulunuldu.
3. Girişim başarılı olursa, sunucuda sertifika için bir eşleşme aranır.
   a. Sertifikanın iptal edilmiş olduğu tespit edilirse, arama işlemi sona ermiştir ve bağlantı isteği başarısız olur; neden kodu MQRC_SSL_CERTIFICATE_REIPTAL edildi.
   b. Sertifika bulunamazsa, arama işlemi sona ermiştir ve bağlantının devam etmesine izin verilir.
4. Sunucuyla iletişim kurma girişimi başarısız olursa, bir CRL sunucusunu tanımlamak için sonraki CertStore nesnesi kullanılır ve işlem 2. adımdan yinelenir.

   Bu, Kaynak Grubundaki son CertStore ise ya da Kaynak Grubu CertStore nesnesi içermediyse, arama işlemi başarısız oldu ve bağlantı isteği MQRC_SSL_CERT_STORE_ERROR neden koduyla başarısız oldu.

Derlem nesnesi, CertStores ' un kullanıldığı sırayı belirler.

CertStores derlemi, CMQC.SSL_CERT_STORE_PROPERTYkullanılarak da ayarlanabilir. Bu özellik, kolaylık sağlamak amacıyla, bir Derlemin üyesi olmadan tek bir CertStore belirtilmesini de sağlar.

sslCertMağazaları boş değere ayarlanırsa, CRL denetimi gerçekleştirilmez. sslCipherSuite ayarlanmazsa bu özellik yoksayılır.

IBM MQ classes for Java içinde gizli anahtarı yeniden müzakere etme
Bir IBM MQ classes for Java istemci uygulaması, istemci bağlantısında şifreleme için kullanılan gizli anahtarın, gönderilen ve alınan toplam bayt sayısı bakımından ne zaman yeniden anlaşıldığını denetleyebilir.

Uygulama bunu aşağıdaki yollardan biriyle yapabilir: Uygulama bu yöntemlerden birden fazlasını kullanıyorsa, olağan öncelik kuralları geçerlidir.

- MQEnvironment sınıfında sslResetSayı alanını ayarlayarak.
- Bir Hashtable nesnesinde MQC.SSL_RESET_COUNT_PROPERTY ortam özelliğini ayarlayarak. Daha sonra uygulama, MQEnvironment sınıfındaki `properties` alanına Hashtable 'ı atar ya da Hashtable 'ı oluşturucusundaki bir MQQueueManager nesnesine geçirir.

sslResetSayı alanı ya da ortam özelliği MQC.SSL_RESET_COUNT_PROPERTY değeri, gizli anahtar yeniden anlaşılmadan önce IBM MQ classes for Java istemci kodu tarafından gönderilen ve alınan toplam bayt sayısını gösterir. Gönderilen bayt sayısı, şifrelemeden önceki sayıdır ve alınan bayt sayısı, şifre çözmeden sonraki sayıdır. Bayt sayısı, IBM MQ classes for Java istemcisi tarafından gönderilen ve alınan denetim bilgilerini de içerir.

Sıfırlama sayısı sıfırsa (varsayılan değer), gizli anahtar hiçbir zaman yeniden anlaşılmaz. CipherSuite belirtilmezse sıfırlama sayısı yoksayılır.

IBM MQ classes for Java içinde özelleştirilmiş bir SSLSocketFactory sağlanması
Uyarlanmış Bir JSSE Yuva Üreticisi kullanıyorsanız, MQEnvironment.sslSocketFactory öğesini uyarlanmış üretici nesnesine ayarlayın. Ayrıntılar farklı JSSE uygulamaları arasında farklılık gösterir.

Farklı JSSE uygulamaları farklı özellikler sağlayabilir. Örneğin, özel bir JSSE uygulaması, belirli bir şifreleme donanımı modelinin yapılandırılmasına izin verebilir. Ayrıca, bazı JSSE sağlayıcıları anahtar depolarının ve güvenli depoların programa göre özelleştirilmesine izin verir ya da anahtar deposundan kimlik sertifikası seçiminin değiştirilmesine izin verir. JSSE ' de tüm bu özelleştirmeler bir üretici sınıfına (javax.net.ssl.SSLSocketFactory) soyutlandırılır.

Uyarlanmış bir SSLSocketFactory somutlaması yaratılmasına ilişkin ayrıntılar için JSSE belgelerinize bakın. Ayrıntılar sağlayıcıdan sağlayıcıya değişir, ancak tipik bir adım sırası şöyle olabilir:

1. SSLContext 'te durağan bir yöntem kullanarak SSLContext nesnesi yarat
2. Bu SSLContext 'i uygun KeyManager ve TrustManager uygulamalarıyla kullanıma hazırlayın (kendi üretici sınıflarından yaratılır)
3. SSLContext 'ten SSLSocketFactory yarat

Bir SSLSocketFactory nesnesine sahip olduğunuzda, MQEnvironment.sslSocketFactory öğesini özelleştirilmiş üretici nesnesine ayarlayın. Örneğin:

```
javax.net.ssl.SSLSocketFactory sf = sslContext.getSocketFactory();
MQEnvironment.sslSocketFactory = sf;
```

IBM MQ classes for Java IBM MQ kuyruk yöneticisine bağlanmak için bu SSLSocketFactory öğesini kullanın. Bu özellik CMQC.SSL_SOCKET_FACTORY_PROPERTYkullanılarak da ayarlanabilir. sslSocketFactory boş değere ayarlanırsa, JVM ' nin varsayılan SSLSocketFactory kullanılır. sslCipherSuite ayarlanmazsa bu özellik yoksayılır.

Özel SSLSocketFactorieskullandığınızda TCP/IP bağlantı paylaşımının etkisini göz önünde bulundurun. Bağlantı paylaşımı olanaklıysa, üretilen yuva sonraki bağlantı isteği bağlamında bir şekilde farklı olsa da, sağlanan SSLSocketFactory için yeni bir yuva istenmez. Örneğin, sonraki bir bağlantıda farklı bir istemci sertifikası sunulacaksa, bağlantı paylaşımına izin verilmemelidir.

IBM MQ classes for Java içindeki JSSE anahtar deposunda ya da güvenli depoda değişiklik yapılması
JSSE anahtar deposunu ya da güvenilir depoyu değiştirirseniz, değişikliklerin yürürlüğe girmesi için belirli işlemler gerçekleştirmeniz gerekir.

JSSE anahtar deposu ya da güvenilir deposunun içeriğini değiştirir ya da anahtar deposu ya da güvenli depo dosyasının konumunu değiştirirseniz, o sırada çalışan IBM MQ classes for Java uygulamaları değişiklikleri otomatik olarak algılamaz. Değişikliklerin yürürlüğe girmesi için aşağıdaki işlemler gerçekleştirilmelidir:

- Uygulamaların tüm bağlantılarını kapatması ve bağlantı havuzlarındaki kullanılmayan bağlantıları yok etmesi gerekir.
- JSSE sağlayıcınız anahtar deposundaki ve güvenilir depodaki bilgileri önbelleğe alırsa, bu bilgilerin yenilenmesi gerekir.

Bu işlemler gerçekleştirildikten sonra uygulamalar bağlantılarını yeniden yaratabilir.

Uygulamalarınızı nasıl tasarladığınıza ve JSSE sağlayıcınız tarafından sağlanan işleve bağlı olarak, uygulamalarınızı durdurmadan ve yeniden başlatmadan bu işlemleri gerçekleştirmek mümkün olabilir. Ancak, uygulamaların durdurulması ve yeniden başlatılması en basit çözüm olabilir.

IBM MQ classes for Java ile TLS kullanılırken hata işleme
TLS kullanarak bir kuyruk yöneticisine bağlanırken IBM MQ classes for Java tarafından bir dizi neden kodu yayınlanabilir.

Bunlar aşağıdaki listede açıklanmıştır:

MQRC_SSL_NOT_ALLOWED
sslCipherSuite özelliği ayarlandı, ancak bağ tanımları bağlantısı kullanıldı. Yalnızca istemci bağlantısı TLS ' yi destekler.

MQRC_JSSE_ERROR
JSSE sağlayıcısı, IBM MQtarafından işlenemeyen bir hata bildirdi. Bu, JSSE ile ilgili bir yapılandırma sorunundan ya da kuyruk yöneticisi tarafından sunulan sertifikanın doğrulanamamasından kaynaklanabilir. JSSE tarafından üretilen kural dışı durum, MQException 'da getCause() yöntemi kullanılarak alınabilir.

MQRC_SSL_INITIALIZION_HATA
TLS yapılandırma seçenekleri belirtilmiş olarak bir MQCONN ya da MQCONNX çağrısı yayınlandı, ancak TLS ortamı kullanıma hazırlanırken bir hata oluştu.

MQRC_SSL_PEER_NAME_MISMATCH
sslPeerAd özelliğinde belirtilen DN kalıbı, kuyruk yöneticisi tarafından sunulan DN ile eşleşmedi.

MQRC_SSL_PEER_NAME_ERROR (QRC_SSL_PEER_NAME_ERROR)
sslPeerAd özelliğinde belirtilen DN kalıbı geçerli değil.

MQRC_UNSUPPORTED_CIPHER_SUITE
sslCipherSuite içinde adı belirtilen CipherSuite JSSE sağlayıcısı tarafından tanınmadı. JSSE sağlayıcısı tarafından desteklenen CipherSuites ' in tam listesi, SSLSocketFactory.getSupportedCipherSuites() yöntemi kullanılarak bir program tarafından edinilebilir. IBM MQ ile iletişim kurmak için kullanılabilecek CipherSuites listesini “IBM MQ classes for Java içinde TLS CipherSpecs ve CipherSuites” sayfa 394içinde bulabilirsiniz.

MQRC_SSL_CERTIFICATE_IPTAL edildi
Kuyruk yöneticisi tarafından sunulan sertifika, sslCertStores özelliğiyle belirtilen bir CRL ' de bulundu. Kuyruk yöneticisini güvenilir sertifikaları kullanacak şekilde güncelleyin.

MQRC_SSL_CERT_STORE_HATA
Kuyruk yöneticisi tarafından sunulan sertifika için, belirtilen CertStores ' ların hiçbirinde arama yapılamadı. MQException.getCause() yöntemi, denenen ilk CertStore aranırken oluşan hatayı döndürür. Nedensel kural dışı durum NoSuchElementException, ClassCastException ya da NullPointerkural dışı durumuysa, sslCertStores özelliğinde belirtilen Derlemin en az bir geçerli CertStore nesnesi içerip içermediğini denetleyin.

IBM MQ classes for Java içinde TLS CipherSpecs ve CipherSuites
IBM MQ classes for Java uygulamalarının bir kuyruk yöneticisiyle bağlantı kurabilme yeteneği, MQI kanalının sunucu ucunda belirtilen CipherSpec 'e ve istemci ucunda belirtilen CipherSuite ' e bağlıdır.

Aşağıdaki tabloda, IBM MQ tarafından desteklenen CipherSpecs ve bunların eşdeğer CipherSuiteslistelenmektedir.

Deprecated Aşağıdaki çizelgede listelenen CipherSpecsöğesinin IBM MQ tarafından kullanımdan kaldırılıp kaldırılmadığını ve kaldırılıp kaldırılmadığını görmek için kullanımdan kaldırılan CipherSpecs konusunu gözden geçirmeniz gerekir.

Önemli: Listelenen CipherSuites , IBM MQile birlikte verilen IBM Java Runtime Environment (JRE) tarafından desteklenmektedir. Listelenen CipherSuites , Oracle Java JRE tarafından desteklenenleri içerir. Uygulamanızı Oracle Java JRE kullanacak şekilde yapılandırma hakkında daha fazla bilgi için bkz. “Uygulamanızın IBM Java ya da Oracle Java CipherSuite eşlemelerini kullanacak şekilde yapılandırılması” sayfa 413.

Çizelge, iletişim için kullanılan protokolü ve CipherSuite ' in FIPS 140-2 standardına uyup uymadığını da gösterir.

Not: AIX, Linux, and Windowsişletim sistemlerinde IBM MQ , IBM Crypto for C (ICC) şifreleme modülü aracılığıyla FIPS 140-2 uyumluluğu sağlar. Bu modüle ilişkin sertifika Geçmiş durumuna taşındı. Müşteriler, IBM Crypto for C (ICC) sertifikasını görüntüleyip NIST tarafından sağlanan tüm önerilere dikkat etmelidir. Yeni bir FIPS 140-3 modülü şu anda devam ediyor ve durumu İşlem listesindeki NIST CMVP modüllerindearanarak görüntülenebilir.

Uygulama, FIPS 140-2 uyumluluğunu zorunlu kılacak şekilde yapılandırılmadıysa, ancak uygulama için FIPS 140-2 uyumluluğu yapılandırıldıysa (yapılandırmaya ilişkin aşağıdaki notlara bakın) yalnızca FIPS 140-2 uyumlu olarak işaretlenmiş CipherSuites yapılandırılabilir; diğer CipherSuites kullanılmaya çalışılması bir hatayla sonuçlanır.

Not: Her JRE birden çok şifreleme güvenliği sağlayıcıya sahip olabilir; bunların her biri aynı CipherSuiteuygulamasına katkıda bulunabilir. Ancak, tüm güvenlik sağlayıcıları FIPS 140-2 sertifikalı değildir. Bir uygulama için FIPS 140-2 uyumluluğu uygulanmazsa, CipherSuite için onaylanmamış bir uygulama kullanılabilir. CipherSuite teorik olarak standardın gerektirdiği minimum güvenlik düzeyini karşılasa bile, onaylanmamış uygulamalar FIPS 140-2 ile uyumlu çalışmayabilir. IBM MQ Java uygulamalarında FIPS 140-2 uygulamasını yapılandırma hakkında daha fazla bilgi için aşağıdaki notlara bakın.

CipherSpecs ve CipherSuitesiçin FIPS 140-2 ve Suite-B uyumluluğu hakkında daha fazla bilgi için bkz. CipherSpecs. ABD Federal Information Processing Standards(Federal Bilgi İşleme Standartları) ile ilgili bilgileri de bilmeniz gerekebilir.

CipherSuites ' in tam kümesini kullanmak ve sertifikalı FIPS 140-2 ve/veya Suite-B uyumluluğuyla çalışmak için uygun bir JRE gereklidir. IBM Java 7 Service Refresh 4 Fix Pack 2 ya da daha yüksek bir IBM JRE düzeyi, Çizelge 59 sayfa 395içinde listelenen TLS 1.2 CipherSuites için uygun desteği sağlar.

TLS 1.3 şifrelerini kullanabilmek için uygulamanızı çalıştıran JRE 'nin TLS 1.3' ü desteklemesi gerekir.

Not: Bazı CipherSuiteskullanmak için JRE 'de' unrestricted ' ilke dosyalarının yapılandırılması gerekir. İlke dosyalarının bir SDK ya da JRE ' de nasıl ayarlandığına ilişkin daha fazla ayrıntı için, kullandığınız sürüme ilişkin *Security Reference for IBM SDK, Java Technology Edition* adlı yayındaki *IBM SDK Policy files* başlıklı konuya bakın.

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	evet

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_AES_128_CBC_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	TLS 1.2	evet

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_AES_128_GCM_SHA256	SSL_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	TLS 1.2	evet

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_AES_256_CBC_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	TLS 1.2	evet

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_AES_256_GCM_SHA384	SSL_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	TLS 1.2	evet
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLS 1.2	hayır

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS 1.2	hayır
ECDHE_RSA_3DES_EDE_CBC_SHA256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	evet

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_AES_128_CBC_SHA256	SSL_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	evet

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_AES_128_GCM_SHA256	SSL_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	evet

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_AES_256_CBC_SHA384	SSL_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	TLS 1.2	evet

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_AES_256_GCM_SHA384	SSL_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	evet
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLS 1.2	hayır

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec “1” sayfa 413	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLS_ECDHE_RSA_WITH_RC4_128_SHA	TLS 1.2	hayır
TLS_RSA_WITH_3DES_EDE_CBC_SHA “2” sayfa 413	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.0	hayır “4” sayfa 413

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	hayır "4" sayfa 413
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	hayır "4" sayfa 413

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_RSA_WITH_AES_128_GCM_SHA256	SSL_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	hayır "4" sayfa 413
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0	hayır "4" sayfa 413

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	hayır "4" sayfa 413
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	hayır "4" sayfa 413

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA	TLS 1.0	hayır
TLS_RSA_WITH_NULL_SHA256	SSL_RSA_WITH_NULL_SHA256	TLS_RSA_WITH_NULL_SHA256	TLS 1.2	hayır
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA	TLS 1.2	hayır

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
ANY_TLS12	*TLS12	*TLS12	TLS 1.2	evet
TLS_AES_128_GCM_SHA256 "3" sayfa 413	TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLS V1.3	hayır
TLS_AES_256_GCM_SHA384 "3" sayfa 413	TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLS V1.3	hayır

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec **"1" sayfa 413**	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_CHACHA20_POLY1305_SHA256 "3" sayfa 413	TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLS V1.3	hayır
TLS_AES_128_CCM_SHA256 "3" sayfa 413	TLS_AES_128_CCM_SHA256	TLS_AES_128_CCM_SHA256	TLS V1.3	hayır

Çizelge 59. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (Şifre Belirtimleri) (devamı var)

CipherSpec “1” sayfa 413	**Eşdeğer CipherSuite (IBM JRE)**	**Eşdeğer CipherSuite (Oracle JRE)**	**Protokol**	**FIPS 140-2 uyumlu**
TLS_AES_128_CCM_8_SHA256 “3” sayfa 413	TLS_AES_128_CCM_8_SHA256	TLS_AES_128_CCM_8_SHA256	TLS V1.3	hayır
Herhangi“3” sayfa 413	*ANY	*ANY	Çoklu	hayır
ANY_TLS13 “3” sayfa 413	*TLS13	*TLS13	TLS V13	hayır
ANY_TLS12_OR_HIGHER “3” sayfa 413	*TLS12ORHIGHER	*TLS12ORHIGHER	TLS 1.2 ve üstü	hayır
ANY_TLS13_OR_HIGHER “3” sayfa 413	*TLS13ORHIGHER	*TLS13ORHIGHER	TLS 1.3 ve üstü	hayır

Notlar:

1. Bu, bir CCDT (ikili ya da JSON) de dahil olmak üzere IBM MQiçindeki bir kanalda yapılandırılan değerdir.
2. Deprecated CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA kullanımdan kaldırılmıştır. Ancak, bağlantı AMQ9288hatasıyla sonlandırılmadan önce 32 GB ' ye kadar veri aktarmak için kullanılabilir. Bu hatayı önlemek için üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckullanırken gizli anahtar sıfırlamasını etkinleştirmeniz gerekir.
3. TLS v1.3 şifrelerini kullanabilmek için uygulamanızı çalıştıran Java runtime environment (JRE) TLS v1.3' ü desteklemelidir.
4. V 9.3.0.17 V 9.3.5.1 IBM MQ 9.3.5 CSU 1 ve IBM MQ 9.3.0 CSU 17işletim sistemlerinde IBM Java 8 JRE, FIPS kipinde çalışırken RSA anahtar değiştokuşu desteğini kaldırır.

IBM MQ classes for Java uygulamasında şifreleme takımlarını ve FIPS uyumluluğunu yapılandırma

- IBM MQ classes for Java kullanan bir uygulama, bir bağlantı için CipherSuite ' i ayarlamak üzere iki yöntemden birini kullanabilir:
  - MQEnvironment sınıfındaki sslCipherSuite alanını CipherSuite adına ayarlayın.
  - MQQueueManager oluşturucusuna CipherSuite adına geçirilen özellikler için CMQC.SSL_CIPHER_SUITE_PROPERTY özelliğini ayarlayın.
- IBM MQ classes for Java kullanan bir uygulama, FIPS 140-2 uyumluluğunu zorlamak için iki yöntemden birini kullanabilir:
  - MQEnvironment sınıfında sslFipsGerekli alanını true olarak ayarlayın.
  - MQQueueManager oluşturucusuna geçirilen özellikler hash çizelgesi için CMQC.SSL_FIPS_REQUIRED_PROPERTYin özelliğini true olarak ayarlayın.

Uygulamanızın IBM Java ya da Oracle Java CipherSuite eşlemelerini kullanacak şekilde yapılandırılması

Not:

V 9.3.3 Continuous Delivery from IBM MQ 9.3.3için, hangi eşlemelerin kullanılacağını denetleyen Java Sistem Özelliği `com.ibm.mq.cfg.useIBMCipherMappings`süründen kaldırılır. IBM MQ 9.3.3' den bir Şifre, CipherSpec ya da CipherSuite adı olarak tanımlanabilir ve IBM MQtarafından doğru şekilde işlenir. IBM MQ classes for Java uygulamanız bir kuyruk yöneticisine güvenli TLS bağlantıları oluşturursa, aşağıdaki üç Jackson JAR dosyası gereklidir:

- `jackson-annotations.jar`
- `jackson-core.jar`
- `jackson-databind.jar`

Önemli: `com.ibm.mq.cfg.useIBMCipherMappings` ile ilgili aşağıdaki bilgiler yalnızca IBM MQ 9.3.3 öncesinde Long Term Support ve Continuous Delivery için geçerlidir.

Uygulamanızın varsayılan IBM Java CipherSuite ile IBM MQ CipherSpec eşlemelerini mi, yoksa Oracle CipherSuite ile IBM MQ CipherSpec eşlemelerini mi kullanacağını yapılandırabilirsiniz. Bu nedenle, uygulamanızın IBM JRE ya da Oracle JRE kullandığından bağımsız olarak TLS CipherSuites kullanabilirsiniz. Java Sistem Özelliği `com.ibm.mq.cfg.useIBMCipherMappings` hangi eşlemelerin kullanılacağını denetler. Özellik aşağıdaki değerlerden biri olabilir:

doğru
IBM Java CipherSuite - IBM MQ CipherSpec eşlemelerini kullanın.

Bu değer varsayılan değerdir.

yanlış
Oracle CipherSuite - IBM MQ CipherSpec eşlemelerini kullanın.

IBM MQ Java ve TLS şifrelerinin kullanılmasıyla ilgili daha fazla bilgi için MQ Java, TLS Şifreleri, Non-IBM JRE 'leri ve APAR' ları IT06775, IV66840, IT09423, IT10837başlıklı MQdev web günlüğü gönderisine bakın.

Birlikte çalışabilirlik sınırlamaları

Bazı CipherSuites , kullanılmakta olan protokole bağlı olarak birden çok IBM MQ CipherSpecile uyumlu olabilir. Ancak yalnızca Tablo 1 'de belirtilen TLS sürümünü kullanan CipherSuite/CipherSpec birleşimi desteklenir. Desteklenmeyen CipherSuites ve CipherSpecs birleşimlerini kullanma girişimi uygun bir kural dışı durumla başarısız olur. Bu CipherSuite/CipherSpec birleşimlerinden herhangi birini kullanan kuruluşlar, desteklenen bir birleşime taşınmalıdır.

Aşağıdaki tabloda, bu sınırlamanın geçerli olduğu CipherSuites gösterilmektedir.

Çizelge 60. CipherSuites ve bunların desteklenen ve desteklenmeyen CipherSpecs

CipherSuite	Desteklenen TLS CipherSpec	Desteklenmeyen SSL CipherSpec
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA “1” sayfa 414	TRIPLE_DES_SHA_US
SSL_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA	DES_SHA_EXPORT (DışA AKTARMA)
SSL_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_RC4_128_SHA256	RC4_SHA_US

Not:

1. Deprecated Bu CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA kullanımdan kaldırılmıştır. Ancak, bağlantı AMQ9288hatasıyla sonlandırılmadan önce 32 GB ' ye kadar veri aktarmak için kullanılabilir. Bu hatayı önlemek için üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckullanırken gizli anahtar sıfırlamasını etkinleştirmeniz gerekir.

IBM MQ classes for Java uygulamalarının çalıştırılması

İstemciyi ya da bağ tanımlama kipini kullanarak bir uygulama (main () yöntemi içeren bir sınıf) yazarsanız, programınızı Java yorumlayıcısını kullanarak çalıştırın.

Şu komutu kullanın:

```
java -Djava.library.path= library_path MyClass
```

Burada *kitaplık_yolu* , IBM MQ classes for Java kitaplıklarının yoludur. Daha fazla bilgi için “IBM MQ classes for Java Kitaplıklar” sayfa 344 başlıklı konuya bakın.

İlgili görevler

IBM MQ classes for Java uygulamalarını izleme

IBM MQ Kaynak Bağdaştırıcısının İzlenmesi

IBM MQ classes for Java ortama bağımlı davranış

IBM MQ classes for Java , farklı IBM MQsürümlerine karşı çalışabilecek uygulamalar oluşturmanızı sağlar. Bu konu derlemi, bu farklı sürümlere bağlı Java sınıflarının davranışını açıklar.

IBM MQ classes for Java , tüm ortamlarda tutarlı işlev ve davranış sağlayan bir sınıf çekirdeği sağlar. Bu çekirdek dışındaki özellikler, uygulamanın bağlı olduğu kuyruk yöneticisinin yeteneğine bağlıdır.

Burada belirtilenler dışında, sergilenen davranış, kuyruk yöneticisine uygun MQI uygulama başvurusunda açıklanmıştır.

IBM MQ classes for Java içindeki temel sınıflar

IBM MQ classes for Java , tüm ortamlarda kullanılabilen bir temel sınıf kümesi içerir.

Aşağıdaki sınıf kümesi çekirdek sınıfları olarak kabul edilir ve yalnızca "IBM MQ classes for Java çekirdek sınıflarına ilişkin kısıtlamalar ve varyasyonlar" sayfa 416içinde listelenen ikincil varyasyonlara sahip tüm ortamlarda kullanılabilir.

- MQOrtamı
- MQException
- MQGetMessageSeçenekleri

  Dışlama:

  - MatchOptions
  - GroupStatus
  - SegmentStatus
  - Bölümleme
- MQManagedObject

  Dışlama:

  - inquire ()
  - set ()
- MQMessage

  Dışlama:

  - groupId
  - messageFlags
  - messageSequenceNumarası
  - offset
  - originalLength
- MQPoolServices
- MQPoolServicesOlayı
- MQPoolServicesEventListener
- MQPoolToken
- MQPutMessageSeçenekleri

  Dışlama:

  - knownDestSayısı
  - unknownDestSayısı
  - invalidDestSayısı
  - recordFields
- MQProcess
- MQQueue
- MQQueueManager

  Dışlama:

  - begin ()
  - accessDistributionList ()
- MQSimpleConnectionYöneticisi
- MQTopic

- MQC

Not:

1. Bazı sabitler çekirdekte bulunmaz (ayrıntılar için bkz. “IBM MQ classes for Java çekirdek sınıflarına ilişkin kısıtlamalar ve varyasyonlar” sayfa 416); bunları tam olarak taşınabilir programlarda kullanmayın.
2. Bazı platformlar tüm bağlantı kiplerini desteklemez. Bu altyapılarda, yalnızca desteklenen kiplerle ilgili temel sınıfları ve seçenekleri kullanabilirsiniz.

IBM MQ classes for Java çekirdek sınıflarına ilişkin kısıtlamalar ve varyasyonlar

Eşdeğer MQI çağrılarının ortam farklılıkları olsa da, temel sınıflar genellikle tüm ortamlarda tutarlı davranır. Davranış, aşağıdaki küçük kısıtlamalar ve çeşitlemeler dışında, bir AIX, Linuxya da Windows kuyruk yöneticisi kullanılıyor gibi olur.

*IBM MQ classes for Java içinde MQGMO_ * değerlerine ilişkin kısıtlamalar*
Bazı MQGMO_ * değerleri tüm kuyruk yöneticileri tarafından desteklenmez.

Aşağıdaki MQGMO_ * değerlerinin kullanılması, bir MQQueue.get() öğesinden MQException yayınlanmasına neden olabilir:

MQGMO_SYNCPOINT_IF_PERSISTENT
MQGMO_MARK_SKIP_BACKOUT
MQGMO_BROWSE_MSG_UNDER_CURSOR
MQGMO_KİLİK
MQGMO_UNLOCK
MQGMO_LOGICAL_ORDER
MQGMO_COMPLETE_MESSAGE
MQGMO_ALL_MSGS_VAR
MQGMO_ALL_SEGMENTS_VAR
MQGMO_UNMARKED_BROWSE_MSG
MQGMO_MARK_BROWSE_HANDLE
MQGMO_MARK_BROWSE_CO_OP
MQGMO_UNMARK_BROWSE_HANDLE
MQGMO_UNMARK_BROWSE_CO_OP

Buna ek olarak, Java' den kullanıldığında MQGMO_SET_SIGNAL desteklenmez.

*IBM MQ classes for Java içinde MQPMRF_ * değerleri için kısıtlamalar*
Bunlar yalnızca dağıtım listesine ileti yerleştirilirken kullanılır ve yalnızca dağıtım listelerini destekleyen kuyruk yöneticileri tarafından desteklenir. Örneğin, z/OS kuyruk yöneticileri dağıtım listelerini desteklemez.

*IBM MQ classes for Java içinde MQPMO_ * değerlerine ilişkin kısıtlamalar*
Bazı MQPMO_ * değerleri tüm kuyruk yöneticileri tarafından desteklenmez

Aşağıdaki MQPMO_ * değerlerinin kullanılması, bir MQQueue.put() ya da MQQueueManager.Put () tarafından MQException yayınlanmasıyla sonuçlanabilir:

MQPMO_LOGICAL_ORDER
MQPMO_NEW_CORREL_ID
MQPMO_NEW_MESSAGE_ID
MQPMO_RESOLVE_LOCAL_Q

*IBM MQ classes for Java içindeki MQCNO_ * değerleri için kısıtlamalar ve varyasyonlar*
Bazı `MQCNO_*` değerleri desteklenmez.

- Otomatik istemci yeniden bağlantısı IBM MQ classes for Javatarafından desteklenmez. Ayarladığınız `MQCNO_RECONNECT_*` değeri ne olursa olsun, bağlantı `MQCNO_RECONNECT_DISABLED`değerini ayarlıyormuş gibi davranmaya devam eder.

- MQCNO_FASTPATH , MQCNO_FASTPATH' u desteklemeyen kuyruk yöneticilerindeki yoksayılır. İstemci bağlantıları tarafından da yoksayılır.

*IBM MQ classes for Java içinde MQRO_ * değerleri için kısıtlamalar*
Aşağıdaki rapor seçenekleri ayarlanabilir.

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

Daha fazla bilgi için bkz. Rapor.

z/OS üzerinde IBM MQ classes for Java ile diğer platformlar arasında çeşitli farklar
IBM MQ for z/OS , bazı bölgelerdeki diğer platformlarda IBM MQ ' den farklı davranır.

BackoutCount
z/OS kuyruk yöneticisi, ileti 255 kereden fazla geriletilmiş olsa da, en çok 255 BackoutCount değerini döndürür.

Varsayılan dinamik kuyruk öneki
Bir z/OS kuyruk yöneticisine bağ tanımlama bağlantısı kullanılarak bağlanıldığında, varsayılan dinamik kuyruk öneki CSQ. * olur. Tersi durumda, varsayılan dinamik kuyruk öneki AMQ. * olur.

MQQueueManager oluşturucusu
Client Connect z/OSüzerinde desteklenmez. İstemci seçenekleriyle bağlantı kurma girişimi MQCC_FAILED ve MQRC_ENVIRONMENT_ERROR ile bir MQException ile sonuçlanır.

MQQueueManager oluşturucusu, MQRC_CHAR_CONVERSION_ERROR (IBM-1047 ve ISO8859-1 kod sayfaları arasında dönüştürmeyi kullanıma hazırlayamazsa) ya da MQRC_UCS2_CONVERSION_ERROR (kuyruk yöneticisinin kod sayfası ile Unicode arasında dönüştürmeyi kullanıma hazırlayamazsa) ile de başarısız olabilir. Uygulamanız bu neden kodlarından biriyle başarısız olursa, Dil Ortamı 'nın Ulusal Dil Kaynakları bileşeninin kurulu olduğundan emin olun ve doğru dönüştürme çizelgelerinin bulunduğundan emin olun.

Unicode için dönüştürme çizelgeleri, z/OS C/C++ isteğe bağlı özelliğinin bir parçası olarak kurulur. UCS-2 dönüşümlerinin etkinleştirilmesiyle ilgili ek bilgi için *z/OS C/C++ Programming Guide*, SC09-4765belgesine bakın.

IBM MQ classes for Java çekirdek sınıflarının dışındaki özellikler
IBM MQ classes for Java , tüm kuyruk yöneticileri tarafından desteklenmeyen API uzantılarını kullanmak için özel olarak tasarlanmış bazı işlevleri içerir. Bu konu derlemi, bunları desteklemeyen bir kuyruk yöneticisi kullanılırken nasıl davrandıklarını açıklar.

MQQueueManager oluşturucu seçeneğindeki çeşitlemeler
MQQueueManager oluşturucuların bazıları isteğe bağlı bir tamsayı bağımsız değişkeni içeriyor. Bu bağımsız değişkenin bazı değerleri tüm platformlarda kabul edilmez.

Bir MQQueueManager oluşturucusu isteğe bağlı bir tamsayı bağımsız değişkeni içerdiğinde, MQI 'ın MQCNO seçenekleri alanıyla eşlenir ve olağan ve hızlı yol bağlantısı arasında geçiş yapmak için kullanılır. Kullanılan seçenekler yalnızca MQCNO_STANDARD_BINDING ya da MQCNO_FASTPATH_BINDING ise, oluşturucunun bu genişletilmiş biçimi tüm ortamlarda kabul edilir. Diğer seçenekler, oluşturucunun MQRC_OPTIONS_ERROR ile başarısız olmasına neden olur. Hızlı yol seçeneği CMQC.MQCNO_FASTPATH_BINDING , yalnızca onu destekleyen bir kuyruk yöneticisine bağ tanımlama bağlantısıyla onurlandırılır. Diğer ortamlarda yoksayılır.

MQQueueManager.begin () yöntemiyle ilgili kısıtlamalar
Bu yöntem yalnızca, bağ tanımlama kipindeki AIX, Linux, and Windows sistemlerinde bir IBM MQ kuyruk yöneticisine karşı kullanılabilir. Tersi durumda, MQRC_ENVIRONMENT_ERROR ile başarısız olur.

Daha fazla ayrıntı için bkz. “IBM MQ classes for Java kullanılarak JTA/JDBC koordinasyonu” sayfa 385 .

MQGetMessageSeçenekleri alanlarındaki varyasyonlar
Bazı kuyruk yöneticileri Sürüm 2 MQGMO yapısını desteklemediği için, bazı alanları varsayılan değerlerine ayarlamanız gerekir.

Sürüm 2 MQGMO yapısını desteklemeyen bir kuyruk yöneticisi kullanırken, aşağıdaki alanları varsayılan değerlerine ayarlanmış olarak bırakın:

GroupStatus
SegmentStatus
Bölümleme

Ayrıca, MatchOptions alanı yalnızca MQMO_MATCH_MSG_ID ve MQMO_MATCH_CORREL_ID ' yi destekler. Bu alanlara desteklenmeyen değerler koyarsanız, sonraki MQDestination.get() MQRC_GMO_ERROR ile başarısız olur. Kuyruk yöneticisi Sürüm 2 MQGMO yapısını desteklemiyorsa, bu alanlar başarılı bir MQDestination.get() işleminden sonra güncellenmez.

IBM MQ classes for Java içindeki dağıtım listelerindeki kısıtlamalar
Her kuyruk yöneticisi bir MQDistributionListaçmanıza izin vermez.

Dağıtım listeleri yaratmak için aşağıdaki sınıflar kullanılır:

MQDistributionList
MQDistributionListÖğesi
MQMessageTracker

MQDistributionLists ve MQDistributionListöğelerini herhangi bir ortamda yaratabilir ve bunlara veri yerleştirebilirsiniz, ancak tüm kuyruk yöneticileri bir MQDistributionListaçmanıza izin vermezler. Özellikle, z/OS kuyruk yöneticileri dağıtım listelerini desteklemez. Böyle bir kuyruk yöneticisi kullanılırken MQDistributionList açma girişimi MQRC_OD_ERROR ile sonuçlanır.

MQPutMessageSeçenekleri alanlarındaki varyasyonlar
Bir kuyruk yöneticisi dağıtım listelerini desteklemiyorsa, bazı MQPMO alanları farklı şekilde işlenir.

MQPMO ' daki dört alan, MQPutMessageSeçenekler sınıfında aşağıdaki üye değişkenleri olarak görsel olarak gerçekleştirilir:

knownDestSayısı
unknownDestSayısı
invalidDestSayısı
recordFields

Bu alanların öncelikli olarak dağıtım listeleriyle kullanılması amaçlanmıştır. Ancak, dağıtım listelerini destekleyen bir kuyruk yöneticisi, bir MQPUT ' den sonra tek bir kuyruğa DestCount alanlarını doldurur. Örneğin, kuyruk bir yerel kuyruğa çözümlenirse, knownDestSayısı 1 olarak ayarlanır ve diğer iki sayı alanı 0 olarak ayarlanır.

Kuyruk yöneticisi dağıtım listelerini desteklemiyorsa, bu değerlerin benzetimi aşağıdaki gibi olur:

- Yer () başarılı olursa, unknownDestSayısı 1 olarak ayarlanır ve diğerleri 0 olarak ayarlanır.
- () başarısız olursa, invalidDestSayısı 1 olarak ayarlanır ve diğerleri 0 olarak ayarlanır.

recordFields değişkeni dağıtım listeleriyle kullanılır. Bir değer, ortamdan bağımsız olarak, istediğiniz zaman recordFields içine yazılabilir. MQPutMessageOptions nesnesi MQDistributionList.Put () yerine sonraki bir MQDestination.put() ya da MQQueueManager.Put () üzerinde kullanılırsa bu yoksayılır.

IBM MQ classes for Java ile MQMD alanlarındaki kısıtlamalar
Bölümlemeyi desteklemeyen bir kuyruk yöneticisi kullanılırken, ileti bölümlemeyle ilgili bazı MQMD alanları varsayılan değerlerinde bırakılmalıdır.

Aşağıdaki MQMD alanları büyük ölçüde ileti bölümlemeyle ilgilidir:

GroupId

MsgSeqNumarası
Göreli Konum
MsgFlags
OriginalLength

Bir uygulama bu MQMD alanlarından herhangi birini varsayılan değerleri dışında bir değere ayarlarsa ve bunları desteklemeyen bir kuyruk yöneticisine bir Put () ya da get () işlemi yaparsa, Put () ya da get () MQRC_MD_ERROR ile bir MQException oluşur. Böyle bir kuyruk yöneticisine sahip başarılı bir koyma değeri () ya da get (), MQMD alanlarını her zaman varsayılan değerlerine ayarlanmış olarak bırakır. İleti gruplamayı ve bölümlemeyi desteklemeyen bir kuyruk yöneticisine karşı çalışan bir Java uygulamasına gruplanmış ya da bölümlenmiş ileti göndermeyin.

Bir Java uygulaması bu alanları desteklemeyen bir kuyruk yöneticisinden ileti alma () girişiminde bulunursa ve alınacak fiziksel ileti, bölümlenmiş iletiler grubunun (yani, MQMD alanları için varsayılan olmayan değerleri olan) bir parçasıysa, hatasız olarak alınır. Ancak, MQMessage içindeki MQMD alanları güncellenmez, MQMessage biçim özelliği MQFMT_MD_EXTENSION olarak ayarlanır ve gerçek ileti verilerinin başına, yeni alanlara ilişkin değerleri içeren bir MQMDE yapısı eklenir.

CICS Transaction Server altında IBM MQ classes for Java ile ilgili kısıtlamalar

CICS Transaction Server for z/OS ortamında, yalnızca ana (ilk) iş parçacığının CICS ya da IBM MQ çağrılarını yapmasına izin verilir.

IBM MQ JMS sınıflarının bir CICS Java uygulamasında kullanım için desteklenmediğini unutmayın.

Bu nedenle, MQQueueManager ya da MQQueue nesnelerini bu ortamdaki iş parçacıkları arasında paylaşmak ya da bir alt iş parçacığında yeni bir MQQueueManager yaratmak mümkün değildir.

z/OS "z/OS üzerinde IBM MQ classes for Java ile diğer platformlar arasında çeşitli farklar" sayfa 417 , bir z/OS kuyruk yöneticisine karşı çalışırken IBM MQ classes for Java için geçerli olan bazı kısıtlamaları ve varyasyonları tanımlar. Buna ek olarak, CICSaltında çalışırken, MQQueueManager üzerindeki hareket denetimi yöntemleri desteklenmez. Uygulamalar MQQueueManager.commit () ya da MQQueueManager.backout () yayınlamak yerine, JCICS görev eşitleme yöntemlerini, Task.commit() ve Task.rollback() yöntemlerini kullanır. Görev sınıfı, com.ibm.cics.server paketindeki JCICS tarafından sağlanır.

IBM MQ kaynak bağdaştırıcısının kullanılması

Kaynak bağdaştırıcısı, bir uygulama sunucusunda çalışan uygulamaların IBM MQ kaynaklarına erişmesine izin verir. Gelen ve giden iletişimi destekler.

Kaynak bağdaştırıcısının içerdiği

V 9.3.0 V 9.3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Java EE ve JMS 2.0'ı destekleyen kaynak bağdaştırıcısına ek olarak IBM MQ 9.3.0 , Jakarta Messaging' i destekleyen bir kaynak bağdaştırıcısı sağlar.

JM 3.0 **Jakarta Messaging için IBM MQ kaynak bağdaştırıcısı**

Jakarta Connectors Architecture , Jakarta EE ortamında çalışan uygulamaları IBM MQ ya da Db2gibi bir Kurumsal Bilgi Sistemine (EIS) bağlamanın standart bir yolunu sağlar. Jakarta Messaging için IBM MQ kaynak bağdaştırıcısı, Jakarta Connectors 2.0.0 arabirimlerini uygular ve IBM MQ classes for Jakarta Messagingarabirimini içerir. Bir uygulama sunucusunda çalışan Jakarta Messaging uygulamalarının ve iletilerle yönlendirilen Bean 'lerin (MDB) IBM MQ kuyruk yöneticisinin kaynaklarına erişmesine olanak sağlar. Kaynak bağdaştırıcısı hem noktadan noktaya iletişim etki alanını hem de yayınlama/abone olma etki alanını destekler.

JMS 2.0 **JMS 2.0 için IBM MQ kaynak bağdaştırıcısı**

Java Platform, Enterprise Edition Connector Architecture (JCA), Java EE ortamında çalışan uygulamaları IBM MQ ya da Db2gibi bir Kurumsal Bilgi Sistemine (EIS) bağlamanın standart bir

yolunu sağlar. JMS 2.0 için IBM MQ kaynak bağdaştırıcısı, JCA 1.7 arabirimlerini uygular ve IBM MQ classes for JMSarabirimini içerir. Bir uygulama sunucusunda çalışan JMS uygulamalarının ve iletilerle yönlendirilen Bean 'lerin (MDB) IBM MQ kuyruk yöneticisinin kaynaklarına erişmesine olanak sağlar. Kaynak bağdaştırıcısı hem noktadan noktaya iletişim etki alanını hem de yayınlama/abone olma etki alanını destekler.

IBM MQ kaynak bağdaştırıcısı, bir uygulama ile bir kuyruk yöneticisi arasında iki tip iletişimi destekler:

Giden iletişim
Bir uygulama bir kuyruk yöneticisine bağlantı başlatır ve ardından JMS iletileri JMS hedeflerine gönderir ve JMS hedeflerinden JMS iletileri zamanuyumlu bir şekilde alır.

Gelen iletişim
JMS hedefine gelen bir JMS iletisi, iletiyi zamanuyumsuz olarak işleyen bir MDB ' ye teslim edilir.

Kaynak bağdaştırıcısı IBM MQ classes for Javaöğesini de içerir. Sınıflar, kaynak bağdaştırıcısının konuşlandırıldığı bir uygulama sunucusunda çalışan uygulamalar için otomatik olarak kullanılabilir ve o uygulama sunucusunda çalışan uygulamaların bir IBM MQ kuyruk yöneticisinin kaynaklarına erişirken IBM MQ classes for Java API ' yı kullanmalarına izin verir.

Java EE ortamında IBM MQ classes for Java kullanımı kısıtlamalar ile desteklenir. Bu kısıtlamalarla ilgili bilgi için bkz. “Java EE içinde IBM MQ classes for Java uygulamalarının çalıştırılması” sayfa 336.

Kaynak bağdaştırıcısının hangi sürümünü kullanacağınız

V 9.3.0 V 9.3.0 Kullandığınız kaynak bağdaştırıcısının sürümü, bağdaştırıcıyı Jakarta EE ya da Java EE' ı destekleyen bir uygulama sunucusuna dağıtıp dağıtmadığınıza bağlıdır:

V 9.3.0 V 9.3.0 JM 3.0 **Jakarta EE**
IBM MQ 9.3.0' dan Jakarta Messaging 3.0 desteklenir. Jakarta Messaging için IBM MQ kaynak bağdaştırıcısı, Jakarta EE' ı destekleyen bir uygulama sunucusunda konuşlandırılmalıdır.

Java EE 7
IBM MQ 8.0 ve sonraki kaynak bağdaştırıcısı JCA v1.7 ' yi destekler ve JMS 2.0 desteği sağlar. Bu kaynak bağdaştırıcısının bir Java EE 7 ve sonraki bir uygulama sunucusunda konuşlandırılması gerekir (bkz. “IBM MQ kaynak bağdaştırıcısı destek bildirimi” sayfa 421).

IBM MQ 8.0 ya da sonraki kaynak bağdaştırıcısını, Java Platform, Enterprise Edition 7 belirtimiyle uyumlu olarak onaylanmış herhangi bir uygulama sunucusuna kurabilirsiniz. IBM MQ 8.0 ya da daha sonraki bir kaynak bağdaştırıcısını kullanarak bir uygulama, BINDINGS ya da CLIENT iletimini kullanarak bir kuyruk yöneticisine bağlanabilir.

Önemli: IBM MQ 8.0 ya da sonraki kaynak bağdaştırıcısı yalnızca JMS 2.0' yi destekleyen bir uygulama sunucusuna konuşlandırılabilir.

Kaynak bağdaştırıcısının WebSphere Application Server traditional ile kullanılması

IBM MQ 9.0' den IBM MQ kaynak bağdaştırıcısı, WebSphere Application Server traditional 9.0 içinde ya da daha sonraki bir sürümde önceden kurulur. Bu nedenle, yeni bir kaynak bağdaştırıcısı kurmak için herhangi bir gereksinim yoktur.

JM 3.0 WebSphere Application Server traditional şu anda Jakarta EE' yi desteklemez. Bkz. IBM MQ kaynak bağdaştırıcısı destek bildirimi.

Not: IBM MQ 9.0 ya da daha sonraki bir kaynak bağdaştırıcısı, CLIENT ya da BINDINGS iletim kipinde herhangi bir hizmet içi IBM MQ kuyruk yöneticisine bağlanabilir.

Kaynak bağdaştırıcısının WebSphere Liberty ile kullanılması

WebSphere Libertyiçinden IBM MQ ' e bağlanmak için IBM MQ kaynak bağdaştırıcısını kullanmanız gerekir. Liberty , IBM MQ kaynak bağdaştırıcısını içermediğinden, bunu Fix Central' dan ayrı olarak edinmeniz gerekir.

V 9.3.0 V 9.3.0 Kullandığınız kaynak bağdaştırıcısının sürümü, bu bağdaştırıcıyı Jakarta EE ya da Java EEürününü destekleyen bir Liberty sürümüne mi konuşlandırdığınıza bağlıdır.

Kaynak bağdaştırıcısını karşıdan yükleme ve kurma hakkında daha fazla bilgi için bkz. “Kaynak bağdaştırıcısının Liberty içine kurulması” sayfa 428.

İlgili kavramlar

“Kaynak bağdaştırıcısının gelen iletişim için yapılandırılması” sayfa 435
Gelen iletişimi yapılandırmak için bir ya da daha çok ActivationSpec nesnesinin özelliklerini tanımlayın.

“Kaynak bağdaştırıcısının giden iletişim için yapılandırılması” sayfa 452
Giden iletişim yapılandırmak için bir ConnectionFactory nesnesinin özelliklerini ve denetlenen bir hedef nesneyi tanımlayın.

“IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma” sayfa 78
IBM MQ classes for JMS ve IBM MQ classes for Jakarta Messaging , IBM MQile verilen Java ileti alışverişi sağlayıcılarıdır. JMS ve Jakarta Messaging belirtimlerinde tanımlanan arabirimleri gerçekleştirmenin yanı sıra, bu ileti alışverişi sağlayıcıları Java ileti sistemi API 'sine iki uzantı kümesi ekler.

“kullanmaIBM MQ classes for Java” sayfa 334
Bir Java ortamında IBM MQ kullanın. IBM MQ classes for Java , bir Java uygulamasının IBM MQ istemcisi olarak IBM MQ ' e bağlanmasına ya da IBM MQ kuyruk yöneticisine doğrudan bağlanmasına izin verir.

İlgili başvurular

Uygulama sunucusunun en son kaynak bağdaştırıcısı bakım düzeyini kullanacak şekilde yapılandırılması
IBM MQ kaynak bağdaştırıcısı için sorun belirleme

WebSphere Application Server Konular

IBM MQ kaynak bağdaştırıcısının bakımı
IBM MQ ileti sistemi sağlayıcısını kullanmak için JMS uygulamalarını Liberty 'ye konuşlandırma

IBM MQ kaynak bağdaştırıcısı destek bildirimi

Bir uygulama ile kuyruk yöneticisi arasındaki iletişim için kullanmanız gereken IBM MQ kaynak bağdaştırıcısı, Jakarta Messaging 3.0 API' yi mi, yoksa JMS 2.0 API ' yı mı kullandığınıza bağlıdır.

JMS 2.0 IBM MQ 8.0 ya da üstü, JMS 2.0 belirtimini uygulayan bir kaynak bağdaştırıcısıyla birlikte gönderilir. Yalnızca Java Platform, Enterprise Edition 7 (Java EE 7) uyumlu bir uygulama sunucusuna konuşlandırılabilir ve bu nedenle JMS 2.0' i destekler. Java Platform, Enterprise Edition için sertifikalı uygulama sunucularının bir listesi, Oracle' ın web sitesindebulunur.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Java EE ve JMS 2.0'ı destekleyen kaynak bağdaştırıcısına ek olarak IBM MQ 9.3.0 , Jakarta Messaging' i destekleyen bir kaynak bağdaştırıcısı sağlar. Aynı uygulamada hem Jakarta Messaging 3.0 API hem de JMS 2.0 API ' nin kullanılması desteklenmez. Daha fazla bilgi için IBM MQ sınıflarının JMS için kullanılmasıbaşlıklı konuya bakın.

WebSphere Liberty içinde devreye alma

WebSphere Liberty 8.5.5 Fix Pack 6 ve daha sonra ve WebSphere Application Server Liberty 9.0 ve daha sonra, IBM MQ 9.0 kaynak bağdaştırıcısının bunlara konuşlandırılabilmesi için Java EE 7 sertifikalı uygulama sunucularıdır.

V 9.3.0 V 9.3.0 Libertyile Jakarta Messaging için IBM MQ kaynak bağdaştırıcısını kullanmak üzere Jakarta EE' i destekleyen bir Liberty sürümü kullanmanız gerekir.

WebSphere Liberty , kaynak bağdaştırıcılarıyla çalışmak için aşağıdaki özelliklere sahiptir:

- V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging 3.0 kaynak bağdaştırıcılarıyla çalışmaya olanak sağlayan messaging-3.0 özelliği.

- JMS 2.0 kaynak bağdaştırıcılarıyla çalışmaya izin vermek için wmqJmsClient-2.0 özelliği.
- JMS 1.1 kaynak bağdaştırıcılarıyla çalışmaya izin vermek için wmqJmsClient-1.1 özelliği.

Önemli:

- V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging için IBM MQ kaynak bağdaştırıcısı, Jakarta EE' ı destekleyen bir Liberty sürümüne konuşlandırılmalıdır. Bu kaynak bağdaştırıcısı, Jakarta EEolmayan daha eski Java EE belirtimini destekleyen Liberty sürümleriyle kullanılamaz.
- JMS 2.0 JMS 2.0 özelliğini destekleyen IBM MQ 8.0 ya da üstü kaynak bağdaştırıcısı wmqJmsClient-2.0 özelliğiyle konuşlandırılmalıdır.

WebSphere Application Server traditional içinde devreye alma

WebSphere Application Server traditional 9.0 , önceden kurulmuş bir IBM MQ 9.0 kaynak bağdaştırıcısıyla birlikte sağlanır. Bu nedenle, yeni bir kaynak bağdaştırıcısı kurmak için herhangi bir gereksinim yoktur. Kurulu kaynak bağdaştırıcısı, desteklenen bir IBM MQsürümünde çalışan herhangi bir kuyruk yöneticisine CLIENT ya da BINDINGS iletim kipinde bağlanabilir. Daha fazla bilgi için bkz “IBM MQ 8.0 ya da daha sonraki kuyruk yöneticilerine bağlanırlık” sayfa 422.

Önemli:

- IBM MQ 9.0 kaynak bağdaştırıcısı IBM MQ 9.0öncesi WebSphere Application Server traditional sürümlerine konuşlandırılamaz; bu sürümler Java EE 7 sertifikalı değildir.
- JM 3.0 WebSphere Application Server traditional şu anda Jakarta EE' yi desteklemez.

WebSphere Application Serverile birlikte gönderilen kaynak bağdaştırıcısı sürümleriyle ilgili daha fazla bilgi için Hangi WebSphere MQ Resource Adapter (RA) sürümü WebSphere Application Serverile birlikte gönderilir? başlıklı teknik nota bakın.

Kaynak bağdaştırıcısının diğer uygulama sunucularıyla kullanılması

Diğer tüm Java EE 7 ya da Jakarta EE uyumlu uygulama sunucuları için, IBM MQ kaynak bağdaştırıcısı Kuruluş Doğrulama Sınaması 'nı (IVT) başarıyla tamamladıktan sonra ortaya çıkan sorunlar, IBM MQ ürün izleme ve diğer IBM MQ tanılama bilgilerinin araştırılması için IBM ' e raporlanabilir. IBM MQ kaynak bağdaştırıcısı IVT başarıyla çalıştırılamazsa, karşılaşılan sorunlar, uygulama sunucusuna özgü yanlış devreye alma ya da yanlış kaynak tanımlamalarından kaynaklanıyor olabilir ve sorunlar, uygulama sunucusu belgeleri ve o uygulama sunucusuna ilişkin destek kuruluşu kullanılarak araştırılmalıdır.

Java Yürütme Ortamı

Uygulama sunucusunu çalıştırmak için kullanılan Java Runtime (JRE), IBM MQ 9.0 ya da üstü istemciyle desteklenen bir yürütme ortamı olmalıdır. Daha fazla bilgi için bkz IBM MQ için Sistem Gereksinimleri. (Hangi sürümü ve işletim sistemini ya da bileşen raporunu görmek istediğinizi seçin ve **Desteklenen Yazılım** sekmesinin altında listelenen **Java** bağlantısını izleyin.)

IBM MQ 8.0 ya da daha sonraki kuyruk yöneticilerine bağlanırlık

Bir IBM MQ 8.0 ya da daha sonraki bir kuyruk yöneticisine bağlanırken, Java EE 7 onaylı bir uygulama sunucusuna konuşlandırılan kaynak bağdaştırıcısı kullanılarak tüm JMS 2.0 işlevselliği kullanılabilir. JMS 2.0 işlevselliğinden yararlanmak için kaynak bağdaştırıcısının, IBM MQ ileti alışverişi sağlayıcısı normal kipini kullanarak kuyruk yöneticisine bağlanması gerekir. Daha fazla bilgi için JMS **PROVIDERVERSION** özelliğini yapılandırmabaşlıklı konuya bakın.

V 9.3.0 V 9.3.0 JM 3.0 Bir IBM MQ 9.3 kuyruk yöneticisine bağlanırken, Jakarta EE sertifikalı bir uygulama sunucusuna konuşlandırılan kaynak bağdaştırıcısı kullanılarak tüm Jakarta Messaging 3.0 işlevselliği kullanılabilir.

MQ Uzantıları

JMS 2.0 belirtimi, belirli davranışların çalışma şeklinde değişiklikler sağlar. IBM MQ 8.0 ya da daha sonraki sürümler bu belirtimi uyguladığından, IBM MQ 8.0 ve sonraki sürümler ile ürünün önceki sürümleri arasında davranış değişiklikleri vardır. IBM MQ 8.0 ya da daha sonra IBM MQ classes for JMS , TRUEdeğerine ayarlandığında bu IBM MQ sürümlerinin bu davranışları IBM WebSphere MQ 7.5 ya da önceki sürümlerine geri döndürmesine neden olan Java sistem özelliği `com.ibm.mq.jms.SupportMQExtensions` desteğini içerir. Özelliğin varsayılan değeri FALSE' tur.

IBM MQ 9.0 ya da daha sonraki kaynak bağdaştırıcısı, `com.ibm.mq.jms.SupportMQExtensions` Java sistem özelliğiyle aynı etkiye ve varsayılan değere sahip `supportMQExtensions` adlı bir kaynak bağdaştırıcısı özelliğini de içerir. Bu kaynak bağdaştırıcısı özelliği varsayılan olarak `ra.xml` içinde false olarak ayarlanır.

Hem kaynak bağdaştırıcısı özelliği hem de Java sistem özelliği ayarlanırsa, sistem özelliğinin önceliği vardır.

WebSphere Application Server traditional 9.0içinde konuşlandırılmış kaynak bağdaştırıcısı içinde, geçişe yardımcı olmak için bu özelliğin otomatik olarak TRUE değerine ayarlandığını unutmayın.

Daha fazla bilgi için bkz "SupportMQExtensions özelliği" sayfa 315.

Genel sorunlar

Oturum arası ayrılması desteklenmiyor

Bazı uygulama sunucuları, aynı JMS oturumunun birden çok harekette kullanılabileceği, ancak aynı anda yalnızca bir tanesinde yer alabileceği, oturum etkileşim adı verilen bir yetenek sağlar. IBM MQ kaynak bağdaştırıcısı bu yeteneği desteklemez; bu da aşağıdaki sorunlara yol açabilir:

```
Bir iletiyi MQ kuyruğuna koyma girişimi başarısız oldu; neden kodu 2072
(MQRC_SYNCPOINT_NOT_KULLANILAMIYOR).
```

xa_close () çağrıları -3 (XAER_PROTO) neden koduyla başarısız olur ve uygulama sunucusundan erişilmekte olan IBM MQ kuyruk yöneticisinde AT040010 bağlantı denetimi tanıtıcısına sahip bir FDC oluşturulur. Bu yeteneğin nasıl devre dışı bırakılacağına ilişkin bilgi için uygulama sunucusu belgelerinize bakın.

Java Transaction API (JTA) XA hareket kurtarma için XA kaynaklarının nasıl kurtarılacağına ilişkin belirtim

JTA belirtiminin 3.4.8 bölümü, XA işlemsel kurtarmayı gerçekleştirmek için XA kaynaklarının yeniden yaratıldığı belirli bir mekanizmayı tanımlamıyor. Bu nedenle, XA hareketine dahil olan XA kaynaklarının nasıl kurtarılacağına ilişkin her bir hareket yöneticisi (ve bu nedenle uygulama sunucusu) söz konusu olacaktır. Bazı uygulama sunucuları için IBM MQ 9.0 kaynak bağdaştırıcısı, XA işlemsel kurtarma gerçekleştirmek için kullanılan uygulama sunucusuna özgü mekanizmaları uygulamıyor olabilir.

ManagedConnectionÜreticisinde eşleşen bağlantılar

Bir uygulama sunucusu, IBM MQ kaynak bağdaştırıcısı tarafından sağlanan bir ManagedConnectionFactory somut örneğinde matchManagedConnections yöntemini başlatabilir. ManagedConnection yalnızca yöntem, uygulama sunucusu tarafından yönteme geçirilen hem **`javax.security.auth.Subject`** hem de **`javax.resource.spi.ConnectionRequestInfo`** bağımsız değişkenleriyle eşleşen bir bağımsız değişken bulursa döndürülür.

IBM MQ kaynak bağdaştırıcısının sınırlamaları

IBM MQ kaynak bağdaştırıcısı tüm IBM MQ platformlarında desteklenir. Ancak, IBM MQ kaynak bağdaştırıcısını kullandığınızda, IBM MQ ürününün bazı özellikleri kullanılamaz ya da sınırlıdır.

IBM MQ kaynak bağdaştırıcısının sınırlamaları şunlardır:

- IBM MQ 8.0' den bu yana, kaynak bağdaştırıcısı JMS 2.0 işlevini sağlayan bir Java Platform, Enterprise Edition 7 (Java EE 7) kaynak bağdaştırıcısıdır. Sonuç olarak, IBM MQ 8.0 ya da daha sonraki bir kaynak bağdaştırıcısının Java EE 7 ya da daha sonraki bir sertifikalı uygulama sunucusuna kurulması gerekir. İstemcide ya da bağ tanımlarında herhangi bir hizmet içi kuyruk yöneticisine bağlanabilir.

- WebSphere Liberty uygulama sunucusu içinde çalışırken, dengelenmiş IBM MQ classes for Java desteklenmez. Diğer uygulama sunucularında IBM MQ classes for Java kullanılması önerilmez. Java EEiçindeki IBM MQ classes for Java ile ilgili dikkat edilmesi gereken noktalara ilişkin ayrıntılar için IBM teknik nota J2EE/JEE Ortamlarında WebSphere MQ Java Arabirimlerini Kullanme bakın.
- z/OSüzerinde WebSphere Liberty uygulama sunucusu içinde çalışırken wmqJmsClient-2.0 özelliği kullanılmalıdır. z/OSiçin soysal JCA desteği mümkün değildir.
- IBM MQ kaynak bağdaştırıcısı, Javadışındaki dillerde yazılmış kanal çıkış programlarını desteklemez.
- Bir uygulama sunucusu çalışırken, sslFipsGerekli özelliğinin değeri tüm JCA kaynakları için true ya da tüm JCA kaynakları için false olmalıdır. Bu, JCA kaynakları eşzamanlı olarak kullanılmasa da bir gereksinimdir. sslFipsGerekli özelliği farklı JCA kaynakları için farklı değerlere sahipse IBM MQ , TLS bağlantısı kullanılmasa bile MQRC_UNSUPPORTED_CIPHER_SUITE neden kodunu verir.
- Bir uygulama sunucusu için birden çok anahtar deposu belirtemezsiniz. Birden çok kuyruk yöneticisiyle bağlantı kurulursa, tüm bağlantıların aynı anahtar deposunu kullanması gerekir. Bu sınırlama WebSphere Application Serveriçin geçerli değildir.
- Birden çok uygun istemci bağlantı kanalı tanımlaması içeren bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsanız, bir hata durumunda kaynak bağdaştırıcısı farklı bir kanal tanımlaması seçebilir ve bu nedenle CCDT ' den farklı bir kuyruk yöneticisi hareket kurtarma sorunlarına neden olabilir. Kaynak bağdaştırıcısı, böyle bir yapılandırmanın kullanılmasını önlemek için herhangi bir işlem yapmaz ve işlem kurtarma için sorunlara neden olabilecek yapılandırmalardan kaçınmak sizin sorumluluğunuzdadır.
- Java EE taşıyıcısında (EJB/Servlet) çalışırken giden bağlantılar için bağlantı yeniden deneme işlevi desteklenmez. Bağdaştırıcı bir JEE taşıyıcısı bağlamında kullanıldığında, hareket yapılanışından ya da hareketsiz kullanımdan bağımsız olarak, giden JMS için bağlantı yeniden deneme işlemi desteklenmez.
- Java EE Connector Architecture sürüm 1.7 belirtiminin 9.1.9 bölümünde tanımlandığı şekilde, JMS bağlantılarının yeniden kimlik doğrulaması desteklenmez. IBM MQ kaynak adpater içindeki `ra.xml` dosyasının **`reauthentication-support`** adlı özelliği `false`değerine ayarlanmış olmalıdır. Uygulama sunucusunun bir JMS bağlantısını yeniden doğrulama girişimi, IBM MQ kaynak bağdaştırıcısının MQJCA1028 ileti koduyla bir javax.resource.spi.SecurityException oluşturmasıyla sonuçlanır.

İlgili görevler

MQI istemcisinde çalıştırma zamanında yalnızca FIPS onaylı CipherSpecs kullanılmasının belirtilmesi

İlgili başvurular

AIX, Linux, and Windows için Federal Bilgi İşleme Standartları (FIPS)

WebSphere Application Server ve IBM MQ kaynak bağdaştırıcısı

IBM MQ kaynak bağdaştırıcısı, WebSphere Application Serveriçindeki IBM MQ ileti alışverişi sağlayıcısıyla JMS ileti alışverişi gerçekleştiren uygulamalar tarafından kullanılır.

Önemli: IBM MQ kaynak bağdaştırıcısını WebSphere Application Server 6.0 ya da WebSphere Application Server 6.1ile kullanmayın.

WebSphere Application Server traditional 9.0 , IBM MQ 9.0 kaynak bağdaştırıcısının bir sürümünü içerir. IBM MQ 9.0 ya da sonraki kaynak bağdaştırıcısı, WebSphere Application Server' un önceki sürümlerine konuşlandırılamaz; bu sürümler Java EE 7 sertifikalı değildir.

V 9.3.0 V 9.3.0 JM 3.0 WebSphere Application Server traditional şu anda Jakarta EE' yi desteklemez. Bkz. IBM MQ kaynak bağdaştırıcısı destek bildirimi.

Bir IBM MQ kuyruk yöneticisinin kaynaklarına WebSphere Application Serveriçinden erişmek için JMS uygulamasını kullanmak istiyorsanız, WebSphere Application Serveriçindeki IBM MQ ileti alışverişi sağlayıcısını kullanın. IBM MQ ileti alışverişi sağlayıcısı, IBM MQ classes for JMS' un bir sürümünü içerir. Daha fazla bilgi için Hangi WebSphere MQ Resource Adapter (RA) sürümü WebSphere Application Serverile birlikte gönderilir?başlıklı teknik nota bakın.

Önemli: Uygulamanıza IBM MQ classes for JMS ya da IBM MQ classes for Java JAR dosyalarının hiçbirini eklemeyin. Bunun yapılması ClassCastKural Dışı Durumlarıyla sonuçlanabilir ve bakımı zor olabilir.

Liberty ve IBM MQ kaynak bağdaştırıcısı

IBM MQ kaynak bağdaştırıcısı, bir Liberty özelliği kullanılarak WebSphere Liberty içine kurulabilir. Kullandığınız özellik, kurmakta olduğunuz kaynak bağdaştırıcısı sürümüne bağlıdır. Diğer bir seçenek olarak, bazı kısıtlamalara bağlı olarak, genel Java Platform, Enterprise Edition Connector Architecture (Java EE JCA) desteğini kullanarak kaynak bağdaştırıcısını kurabilirsiniz.

Kaynak bağdaştırıcısını Liberty içine kurarken genel kısıtlamalar

wmqJmsClient-1.1 ya da wmqJmsClient-2.0 özelliği kullanıldığında ve soysal JCA desteği kullanıldığında kaynak bağdaştırıcısı için aşağıdaki kısıtlamalar geçerlidir:

- IBM MQ classes for Java , Libertyiçinde desteklenmez. Bunlar IBM MQ Liberty ileti sistemi özelliğiyle ya da genel JCA desteğiyle kullanılmamalıdır. Daha fazla bilgi için bkz. WebSphere MQ Java Arabirimlerinin J2EE/JEE Ortamlarında Kullanılması.
- IBM MQ kaynak bağdaştırıcısının iletim tipi BINDINGS_THEN_CLIENT. Bu iletim tipi, IBM MQ Liberty ileti sistemi özelliğinde desteklenmez.
- IBM MQ 9.0' den önce, Advanced Message Security (AMS) özelliği IBM MQ Liberty ileti sistemi özelliğine dahil edilmemiştir. Ancak, AMS bir IBM MQ 9.0 ya da daha sonraki bir kaynak bağdaştırıcısıyla desteklenir.

Not: IBM MQ 9.0.0.6 ve IBM MQ 9.1.0.1 ' den büyük IBM MQ sürümlerinde, ssl-1.0 özelliği yerine transportSecurity-1.0 özelliğini kullanmalısınız.

Daha fazla bilgi için bkz.

Libertyiçinde SSL iletişimini etkinleştirme
Liberty içinde SSL varsayılanları
İletim Güvenliği 1.0

Liberty özelliklerini kullanırken kısıtlamalar

WebSphere Liberty 8.5.5 Fix Pack 2 - WebSphere Liberty 8.5.5 Fix Pack 5 da dahil olmak üzere, yalnızca wmqJmsClient-1.1 özelliği kullanılabilir ve yalnızca JMS 1.1 kullanılabilir. WebSphere Liberty 8.5.5 Fix Pack 6 , JMS 2.0 kullanılabilmesi için wmqJmsClient-2.0 özelliğini ekledi.

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 desteklenir. Libertyile Jakarta Messaging için IBM MQ kaynak bağdaştırıcısını kullanmak üzere Jakarta EE' i destekleyen bir Liberty sürümü kullanmanız gerekir. Jakarta Messaging için kaynak bağdaştırıcısını Liberty generic messaging-3.0 özelliğiyle kullanmanız gerekir.

Kullanmanız gereken özellik, kullandığınız kaynak bağdaştırıcısı sürümüne bağlıdır:

- IBM MQ 8.0.0 Fix Pack 3 ve üstü IBM MQ 8.0 kaynak bağdaştırıcısı yalnızca wmqJmsClient-2.0 özelliğiyle kullanılabilir.
- IBM MQ 9.0 kaynak bağdaştırıcısı yalnızca wmqJmsClient-2.0 özelliğiyle kullanılabilir.
- V 9.3.0 V 9.3.0 JM 3.0 messaging-3.0 özelliği, Jakarta Messaging 3.0 kaynak bağdaştırıcılarıyla çalışmaya olanak sağlar.

Soysal JCA desteği kullanılırken kısıtlamalar

Soysal JCA desteği kullanıyorsanız, aşağıdaki kısıtlamalar geçerlidir:

- Soysal JCA desteğini kullanırken JMS düzeyini belirtmeniz gerekir. JMS 2.0 ve JCA 1.7 , yalnızca IBM MQ 8.0.0 Fix Pack 3 ve daha sonraki IBM MQ 8.0 kaynak bağdaştırıcılarıyla birlikte kullanılmalıdır.
- Soysal JCA desteğini kullanarak z/OS üzerinde IBM MQ kaynak bağdaştırıcısı çalıştırılamaz. IBM MQ kaynak bağdaştırıcısını z/OSüzerinde çalıştırmak için, bağdaştırıcının wmqJmsClient-1.1 ya da wmqJmsClient-2.0 özelliğiyle çalıştırılması gerekir.
- Kaynak bağdaştırıcısının yeri şu xml öğesi kullanılarak belirtilir:

```
JMS 3.0 <resourceAdapter id="mqJms" location="${server.config.dir}/
wmq.jakarta.jmsra.rar">
  <classloader apiTypeVisibility="spec, ibm-api, api, third-party"/>
</resourceAdapter>
```

```
JMS 2.0 <resourceAdapter id="mqJms" location="${server.config.dir}/wmq.jmsra.rar">
  <classloader apiTypeVisibility="spec, ibm-api, api, third-party"/>
</resourceAdapter>
```

Önemli: ID etiketinin değeri `wmqJms`dışında herhangi bir değer olabilir. Tanıtıcı olarak `wmqJms` kullanırsanız, Liberty kaynak bağdaştırıcısını düzgün şekilde yükleyemez. Bunun nedeni, `wmqJms` ' in IBM MQile ilgili belirli bir özelliğe başvurmak için dahili olarak kullanılan tanıtıcı olmasıdır. Gerçekten bir NullPointerkural dışı durumu yaratır.

Aşağıdaki örneklerde, JMS 2.0çalıştırılırken `server.xml` dosyasındaki bazı parçacıklar gösterilmektedir:

```
<!-- Enable features -->
  <featureManager>
    <feature>servlet-3.1</feature>
    <feature>jndi-1.0</feature>
    <feature>jca-1.7</feature>
    <feature>jms-2.0</feature>
  </featureManager>
```

İpucu: jca-1.7 ve jms-2.0 özelliklerinin ve wmqJmsClient-2.0 özelliğinin bulunmadığına dikkat edin.

```
<resourceAdapter id="mqJms" location="${server.config.dir}/wmq.jmsra.rar">
  <classloader apiTypeVisibility="spec, ibm-api, api, third-party"/>
</resourceAdapter>
```

İpucu: Tercih edilen kimlik için `mqJms` kullanımına dikkat edin. `wmqJms`kullanmayın.

```
<application id="WMQHTTP" location="${server.config.dir}/apps/WMQHTTP.war"
name="WMQHTTP" type="war">
    <classloader apiTypeVisibility="spec, ibm-api, api, third-party"
classProviderRef="mqJms"/>
</application>
```

İpucu: classloaderProviderRef to the resource adapter with the ID `mqJms`(Tanıtıcı aracılığıyla kaynak bağdaştırıcısına geri dön) seçeneğini not edin; bu, IBM MQ' e özgü sınıfların yüklenmesine izin vermektir.

Soysal JCA desteği kullanılarak izleme sırasında kısıtlamalar

İzleme ve günlüğe kaydetme, Liberty izleme sistemiyle bütünleştirilmez. Bunun yerine, IBM MQ kaynak bağdaştırıcısı izlemesi, İzleme IBM MQ classes for JMS uygulamalarındaaçıklandığı gibi Java sistem özellikleri ya da bir IBM MQ classes for JMS yapılandırma dosyası kullanılarak etkinleştirilmelidir. Libertyiçinde Java sistem özelliklerinin nasıl ayarlanacağına ilişkin ayrıntılar için WebSphere Liberty belgelerinebakın.

Örneğin, Liberty 19.0.0.9' da IBM MQ kaynak bağdaştırıcısının izlenmesini etkinleştirmek için Liberty file `jvm.options`kütüğüne bir giriş ekleyin:

1. `jvm.options`adlı bir metin dosyası oluşturun.
2. İzlemeyi etkinleştirmek için aşağıdaki JVM seçeneklerini bu dosyaya ekleyin:

   ```
   -Dcom.ibm.msg.client.commonservices.trace.status=ON
   -Dcom.ibm.msg.client.commonservices.trace.outputName=C:\Trace\MQRA-WLP_%PID%.trc
   ```

3. Bu ayarları tek bir sunucuya uygulamak için `jvm.options` adresini kaydedin:

   ```
   ${server.config.dir}/jvm.options
   ```

Bu değişiklikleri tüm Libertyuygulamasına uygulamak için `jvm.options` adresini kaydedin:

```
${wlp.install.dir}/etc/jvm.options
```

Bu, yerel olarak tanımlanmış bir `jvm.options` dosyası olmayan tüm JVM ' ler için geçerli olur.

4. Değişiklikleri etkinleştirmek için sunucuyu yeniden başlatın.

Bu, izlemenin `<path_to_trace_to>`dizininde `MQRA-WLP_<process identifier>.trc` adlı bir izleme dosyasına yazılması ile sonuçlanır.

İstemci kanal tanımlama çizelgeleriyle tam Liberty XA desteği

WebSphere Liberty 18.0.0.2 ' i IBM MQ 9.2.0 ya da daha sonraki bir sürümle kullanırken, XA hareketleriyle birlikte istemci kanal tanımlama çizelgesi (CCDT) içindeki kuyruk yöneticisi gruplarından yararlanabilirsiniz. Bu, artık işlem bütünlüğünü korurken kuyruk yöneticisi grupları tarafından sağlanan iş yükü dağıtımından ve kullanılabilirliğinden yararlanmanın mümkün olduğu anlamına gelir.

Bir kuyruk yöneticisine bağlantı hataları olması durumunda, hareketin çözülebilmesi için kuyruk yöneticisinin yeniden kullanılabilir olması gerekir. Hareket kurtarma işlemi Libertytarafından yönetilir ve kuyruk yöneticilerinin yeniden kullanılabilir olması için uygun bir süre tanınması için hareket yöneticisini yapılandırmanız gerekebilir. Daha fazla bilgi için WebSphere Liberty ürün belgelerinde Transaction Manager (transaction) başlıklı konuya bakın.

Bu bir istemci tarafı özelliğidir; başka bir deyişle, IBM MQ 9.2.0 ya da daha sonraki bir kuyruk yöneticisine değil, bir IBM MQ 9.2.0 ya da daha sonraki bir kaynak bağdaştırıcısına gereksinim duyarsınız.

IBM MQ kaynak bağdaştırıcısının takılması

IBM MQ kaynak bağdaştırıcısı bir kaynak arşivi (RAR) dosyası olarak sağlanır. RAR dosyasını uygulama sunucunuza kurun. Sistem yoluna dizin eklemeniz gerekebilir.

Bu görev hakkında

IBM MQ kaynak bağdaştırıcısı bir kaynak arşivi (RAR) dosyası olarak sağlanır:

- V 9.3.0 V 9.3.0 JM 3.0 Jakarta Messaging 3.0için bu dosya `wmq.jakarta.jmsra.rar`olarak adlandırılır. RAR dosyası, Jakarta Connectors Architecture (JCA) arabirimlerinin IBM MQ classes for Jakarta Messaging ve IBM MQ uygulamasını içerir.
- JMS 2.0 JMS 2.0için bu dosya `wmq.jmsra.rar`olarak adlandırılır. RAR dosyası, Java EE Connector Architecture (JCA) arabirimlerinin IBM MQ classes for JMS ve IBM MQ uygulamasını içerir.

Kaynak bağdaştırıcısını IBM MQ ürün kuruluşunun bir parçası olarak kurduğunuzda, RAR dosyası Çizelge 61 sayfa 427içinde gösterilen dizine IBM MQ classes for JMS ile birlikte kurulur.

Çizelge 61. Her platform için RAR dosyasını içeren IBM MQ dizini

Hizmet olarak sunulan	**Dizin**
AIX and Linux	*MQ_INSTALLATION_PATH*/java/lib/jca
IBM i	/QIBM/ProdData/mqm/java/lib/jca
Windows	*MQ_INSTALLATION_PATH*\java\lib\jca
z/OS	*MQ_INSTALLATION_PATH*/java/lib/jca

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Bir uygulama sunucusundan IBM MQ ' e bağlanmak için IBM MQ kaynak bağdaştırıcısını kullanmanız gerekir. Kullanmakta olduğunuz uygulama sunucusuna bağlı olarak, kaynak bağdaştırıcısı önceden kurulmuş olabilir ya da kendiniz kurmanız gerekebilir.

Çizelge 62. Kaynak bağdaştırıcısının uygulama sunucusuna kurulması	
Uygulama sunucusu	**Önceden kurulu mu, yoksa kurmanız mı gerekiyor?**
WebSphere Application Server traditional 9.0	IBM MQ 9.0 kaynak bağdaştırıcısı WebSphere Application Server traditional 9.0içinde önceden kurulur. Bu nedenle, WebSphere Application Server traditional 9.0içine yeni bir kaynak bağdaştırıcısı takmanıza gerek yoktur.
WebSphere Liberty	WebSphere Liberty , IBM MQ kaynak bağdaştırıcısını içermez, bu nedenle bunu Fix Central' dan ayrı olarak edinmeniz gerekir.
Diğer Java EE ya da Jakarta EE uygulama sunucusu	Kaynak bağdaştırıcısını, WebSphere Libertyiçin olduğu gibi Fix Centralolanağından ayrı olarak edinin.

Yordam

- IBM MQ from WebSphere Libertyya da başka bir Java EE ya da Jakarta EE uygulama sunucusuna bağlanıyorsanız, IBM MQ kaynak bağdaştırıcısını "Kaynak bağdaştırıcısının Liberty içine kurulması" sayfa 428başlıklı konuda açıklandığı gibi yükleyin ve kurun.
- Linux AIX

  AIX and Linux sistemlerindeki bağ tanımları bağlantıları için, Java Native Interface (JNI) kitaplıklarını içeren dizinin sistem yolunda olduğundan emin olun.

  IBM MQ classes for JMS kitaplıklarını da içeren bu dizinin konumu için bkz. "Java Native Interface (JNI) kitaplıklarının yapılandırılması" sayfa 92.

  Windows Windowssistemlerinde bu dizin, IBM MQ classes for JMSkuruluşu sırasında sistem yoluna otomatik olarak eklenir.

  İpucu: Sistem yolunu ayarlamanın bir alternatifi olarak IBM MQ kaynak bağdaştırıcısının, JNI kitaplığının yerini belirtmek için kullanılabilecek nativeLibraryadlı bir özelliği vardır. Örneğin, WebSphere Liberty içinde bu, aşağıdaki örnekte gösterildiği gibi yapılandırılır:

  ```
  <wmqJmsClient nativeLibraryPath="/opt/mqm/java/lib64"/>
  ```

  İşlemler hem istemci hem de bağ tanımlama kipinde desteklenir.

Kaynak bağdaştırıcısının Liberty içine kurulması

IBM MQ from WebSphere Libertyya da diğer Java EE ya da Jakarta EE uygulama sunucularına bağlanmak için IBM MQ kaynak bağdaştırıcısını kullanmanız gerekir. Liberty , IBM MQ kaynak bağdaştırıcısını içermediğinden, bunu Fix Central' dan ayrı olarak edinmeniz gerekir.

Başlamadan önce

Not: Bu konudaki bilgiler WebSphere Application Server traditional 9.0için geçerli değildir. IBM MQ 9.0 kaynak bağdaştırıcısı WebSphere Application Server traditional 9.0içinde önceden kurulur. Bu nedenle, bu durumda yeni bir kaynak bağdaştırıcısı kurulmasına gerek yoktur.

Bu görevi başlatmadan önce, makinenizde bir Java runtime environment (JRE) kurulu olduğundan ve JRE ' nin sistem yoluna eklendiğinden emin olun.

Bu kuruluş işleminde kullanılan Java kuruluş programı, kök kullanıcı ya da belirli bir kullanıcı olarak çalıştırılmasını gerektirmez. Tek gereksinim, çalıştırıldığı kullanıcının, dosyaların girmesini istediğiniz dizine yazma erişimi olması.

WebSphere Liberty 8.5.5 Fix Pack 1' e kadar olan Liberty sürümlerinde, bir EJB yalnızca `ejb-jar.xml`içindeki yapılandırma kullanılarak devreye alınırsa, Liberty Profilin kullandığı WebSphere Application Server sürümünde APAR PM89890 uygulanmış olmalıdır. Bu yapılandırma yöntemi, kaynak bağdaştırıcısının kuruluş doğrulama programı (IVT) için kullanılır; bu nedenle, IVT ' nin çalışması için bu APAR gereklidir.

IBM MQ 9.3.0' dan Jakarta Messaging 3.0 desteklenir. Libertyile Jakarta Messaging için IBM MQ kaynak bağdaştırıcısını kullanmak üzere Jakarta EE' i destekleyen bir Liberty sürümü kullanmanız gerekir. Örneğin, Liberty generic messaging-3.0 özelliğini kullanabilirsiniz.

Bu görev hakkında

Fix Central adresinden yükleyebileceğiniz kaynak bağdaştırıcısına ilişkin JAR dosyası yürütülebilir. Bu yürütülür dosyayı çalıştırdığınızda, kabul edilmesi gereken IBM MQ lisans sözleşmesi görüntülenir. IBM MQ kaynak bağdaştırıcısının takılacağı bir dizin ister. Kaynak bağdaştırıcısı RAR dosyası ve kuruluş doğrulama sınaması (IVT) programı bu dizine kurulur. Varsayılanı kabul edebilir ya da başka bir dizin belirtebilirsiniz; bu dizin bir uygulama sunucusunun kaynak bağdaştırıcıları dizini ya da sisteminizdeki başka bir dizin olabilir. Dizin yoksa, kuruluşun bir parçası olarak yaratılır.

IBM MQ 9.0öncesinde, karşıdan yüklenecek dosyanın adı *V.R.M.F*`-WS-MQ-Java-InstallRA.jar`biçimindeydi; örneğin, `8.0.0.6-WS-MQ-Java-InstallRA.jar`. IBM MQ 9.0' den dosya adının biçimi şöyledir: *V.R.M.F*`-IBM-MQ-Java-InstallRA.jar`; örneğin, `9.0.0.0-IBM-MQ-Java-InstallRA.jar`.

Kaynak bağdaştırıcısını karşıdan yükleyip kurduktan sonra, WebSphere Libertyiçinde yapılandırmaya hazırsınız.

Yordam

1. IBM MQ kaynak bağdaştırıcısını Fix Centraladresinden yükleyin.
   a) Bu bağlantıyı tıklatın: IBM MQ Kaynak Bağdaştırıcısı.
   b) Görüntülenen kullanılabilir düzeltmeler listesinde, IBM MQ sürümünüze ilişkin kaynak bağdaştırıcısını bulun.
   Örneğin:

   ```
   release level: 9.1.4.0-IBM-MQ-Java-InstallRA
   Continuous Delivery Release: 9.1.4 IBM MQ Resource Adapter for use with Application
   Servers
   ```

   Daha sonra kaynak bağdaştırıcısı dosya adını tıklatın ve karşıdan yükleme işlemini izleyin.
2. Dosyayı karşıdan yüklediğiniz dizinden aşağıdaki komutu girerek kuruluşu başlatın.
   IBM MQ 9.0' da komutun biçimi şöyledir:

   ```
   java -jar V.R.M.F-IBM-MQ-Java-InstallRA.jar
   ```

   Burada *V.R.M.F* , Sürüm, Yayın, Değişiklik ve Düzeltme Paketi numarasıdır ve *V.R.M.F*`-IBM-MQ-Java-InstallRA.jar` , Fix Centraladresinden yüklenen dosyanın adıdır.
   Örneğin, IBM MQ 9.1.4 yayın düzeyine ilişkin IBM MQ kaynak bağdaştırıcısını kurmak için aşağıdaki komutu kullanabilirsiniz:

   ```
   java -jar 9.1.4.0-IBM-MQ-Java-InstallRA.jar
   ```

   Not: Bu kuruluşu gerçekleştirmek için, makinenizde kurulu bir JRE olması ve sistem yoluna eklenmiş olması gerekir.

   Komutu girdiğinizde aşağıdaki bilgiler görüntülenir:

   ```
   IBM MQ 9.1programını kullanmadan, çıkarmadan ya da kurmadan önce kabul etmeniz gerekir
   1 'in koşulları. IBM Uluslararası Lisans Sözleşmesi-Değerlendirme
   ```

```
Programlar 2. IBM Uluslararası Program Lisans Sözleşmesi ve ek
lisans bilgileri. Lütfen aşağıdaki lisans sözleşmelerini dikkatle okuyun.

Lisans sözleşmesi,
--viewLicenseSözleşmesi seçeneği.
Lisans koşullarını şimdi görüntülemek için Enter, atlamak için 'x' tuşuna basın.
```

3. Lisans koşullarını inceleyin ve kabul edin:
   a) Lisansı görüntülemek için Enter tuşuna basın.

      Diğer bir seçenek olarak, x tuşuna basıldığında lisans görüntüsü atlanıyor.

      Lisans görüntülendikten sonra ya da x seçildikten hemen sonra, ek lisans koşullarını görüntülemeyi seçebileceğinizi bildiren aşağıdaki ileti görüntülenir:

      ```
      Ek lisans bilgileri,
      --viewLicenseBilgi seçeneği.
      Ek lisans bilgilerini şimdi görüntülemek için Enter, atlamak için 'x' tuşuna basın.
      ```

   b) Ek lisans koşullarını görüntülemek için Enter tuşuna basın.

      Diğer bir seçenek olarak, x tuşuna basıldığında ek lisans koşullarının görüntülenmesi atlanıyor.

      Ek lisans koşulları görüntülendikten sonra ya da x seçildikten hemen sonra, lisans sözleşmesini kabul etmenizi isteyen aşağıdaki ileti görüntülenir:

      ```
      Aşağıdaki "I Agree" (Kabul ediyorum) seçeneğini belirleyerek,
      lisans sözleşmesi ve geçerliyse,IBM dışı koşullar. Yapmazsanız
      kabul ediyorum, "Kabul etmiyorum" seçeneğini belirleyin.

      [ 1] I Agree (Kabul Ediyorum) ya da [ 2] I do not Agree (Kabul Ediyorum) seçeneğini
      belirleyin:
      ```

   c) Lisans sözleşmesini kabul etmek ve kuruluş dizinini seçmeye devam etmek için 1 'i seçin.

      Diğer bir seçenek olarak, 2 'yi seçerseniz kuruluş hemen sona erer.

      1 değerini seçtiyseniz, hedef kuruluş dizini seçmenizi isteyen aşağıdaki ileti görüntülenir:

      ```
      Ürün dosyaları için dizin girin ya da varsayılan değeri kabul etmek için boş bırakın.
      Varsayılan hedef dizin H: \Liberty\WMQ ' dur
      Ürün dosyaları için hedef dizin?
      ```

4. Kaynak bağdaştırıcısı için kuruluş dizinini belirtin:
   - Kaynak bağdaştırıcısını varsayılan konuma kurmak istiyorsanız, değer belirtmeden Enter tuşuna basın.
   - Kaynak bağdaştırıcısını varsayılana göre farklı bir yere kurmak istiyorsanız, kaynak bağdaştırıcısını kurmak istediğiniz dizinin adını belirtin ve Enter tuşuna basın.

   Dosyalar seçilen konuma kurulduktan sonra, aşağıdaki örnekte gösterildiği gibi bir onay iletisi görüntülenir:

   ```
   Dosyalar H: \Liberty\WMQ \wmq dizinine açılıyor
   Tüm ürün dosyaları başarıyla çıkarıldı.
   ```

   Kuruluş sırasında, seçilen kuruluş dizininde `wmq` adlı yeni bir dizin yaratılır ve daha sonra, aşağıdaki dosyalar `wmq` dizinine kurulur:

   - Kuruluş doğrulama sınaması programı, `wmq.jakarta.jmsra.ivt` (Jakarta Messaging 3.0) ya da `wmq.jmsra.ivt` (JMS 2.0).
   - IBM MQ RAR dosyası,`wmq.jakarta.jmsra.rar` (Jakarta Messaging 3.0 ya da `wmq.jmsra.rar` (JMS 2.0).

5. JMS 2.0

   İsteğe bağlı: WebSphere Liberty Profileiçinde Java EE 7 (JMS 2.0) kaynak bağdaştırıcısını yapılandırın.

   Liberty içinde kaynak bağdaştırıcısını yapılandırmak için yapmanız gereken adımlar şunlardır. Daha fazla bilgi için WebSphere Application Server ürün belgelerinebakın.

   a) IBM MQ kaynak bağdaştırıcısıyla çalışmaya izin vermek için wmqJmsClient-2.0 özelliğini `server.xml` dosyasına ekleyin.

      Daha fazla bilgi için bkz “Kaynak bağdaştırıcısının hangi sürümünü kullanacağınız” sayfa 420.

   b) Kurduğunuz `wmq.jmsra.rar` (JMS 2.0) dosyasına bir başvuru ekleyin.

   JNDI ile sunucu uygulamalarını ve MDB ' leri desteklemek için örnek bir yapılandırma şöyle görünebilir:

```
 <featureManager>
    <feature>wmqJmsClient-2.0</feature>
    <feature>servlet-3.0</feature>
    <feature>jmsMdb-3.1</feature>
    <feature>jndi-1.0</feature>
    </featureManager>

<variable name="wmqJmsClient.rar.location"
      value="H:\Liberty\WMQ\wmq\wmq.jmsra.rar"/>
```

6. JM 3.0
İsteğe bağlı: WebSphere Liberty Profileiçinde Jakarta EE 9 (Jakarta Messaging 3.0) kaynak bağdaştırıcısını yapılandırın.

   Liberty içinde kaynak bağdaştırıcısını yapılandırmak için yapmanız gereken adımlar şunlardır. Daha fazla bilgi için WebSphere Application Server ürün belgelerinebakın.

   a) IBM MQ kaynak bağdaştırıcısıyla çalışmaya izin vermek için wmqJmsClient-3.0 özelliğini `server.xml` dosyasına ekleyin.

   Daha fazla bilgi için bkz “Kaynak bağdaştırıcısının hangi sürümünü kullanacağınız” sayfa 420.

   b) Kurduğunuz `wmq.jakarta.jmsra.rar` (Jakarta Messaging 3.0) dosyasına bir başvuru ekleyin.

   JNDI ile sunucu uygulamalarını ve MDB ' leri desteklemek için örnek bir yapılandırma şöyle görünebilir:

```
 <featureManager>
    <feature>wmqJmsClient-3.0</feature>
    <feature>servlet-3.0</feature>
    <feature>jmsMdb-3.1</feature>
    <feature>jndi-1.0</feature>
    </featureManager>

<variable name="wmqJmsClient.rar.location"
      value="H:\Liberty\WMQ\wmq\wmq.jmsra.rar"/>
```

   Not: WebSphere Liberty Profileyerine Open Libertykullanıyorsanız, "wmqJmsClient-3.0" yerine "messagingClient-3.0" genel kaynak bağdaştırıcısı destek özelliğini kullanmanız gerekir ve yapılandırmanın diğer yönleri farklı olur. Daha fazla ayrıntı için lütfen Open Liberty belgelerine bakın.

IBM MQ kaynak bağdaştırıcısının yapılandırılması

IBM MQ kaynak bağdaştırıcısını yapılandırmak için çeşitli Java Platform, Enterprise Edition Connector Architecture (JCA) kaynaklarını ve isteğe bağlı olarak sistem özelliklerini tanımlarsınız. Kuruluş doğrulama sınamasını (IVT) çalıştırmak için kaynak bağdaştırıcısını da yapılandırmanız gerekir. IBM hizmeti,IBM dışı herhangi bir uygulama sunucusunun doğru yapılandırıldığını göstermek için bu programın çalıştırılmasını gerektirebileceğinden bu önemlidir.

Başlamadan önce

Bu görev, JMS ve IBM MQ classes for JMSile ilgili bilgi sahibi olduğunuzu varsayar. IBM MQ kaynak bağdaştırıcısını yapılandırmak için kullanılan özelliklerin çoğu, IBM MQ classes for JMS nesnelerinin özelliklerine eşdeğerdir ve aynı işleve sahiptir.

Bu görev hakkında

Her uygulama sunucusu kendi denetim arabirimi kümesini sağlar. Bazı uygulama sunucuları JCA kaynaklarını tanımlamak için grafik kullanıcı arabirimleri sağlar, ancak diğerleri yöneticinin XML konuşlandırma planlarını yazmasını gerektirir. Bu nedenle, her uygulama sunucusu için IBM MQ kaynak bağdaştırıcısının nasıl yapılandırılacağı hakkında bilgi sağlamak bu belgenin kapsamı dışındadır.

Bu nedenle, aşağıdaki adımlar yalnızca yapılandırmanız gereken öğeye odaklanır. JCA kaynak bağdaştırıcısının nasıl yapılandırılacağına ilişkin bilgi için uygulama sunucunuzla birlikte sağlanan belgelere bakın.

Yordam

Aşağıdaki kategorilerde JCA kaynaklarını tanımlayın:

- ResourceAdapter nesnesinin özelliklerini tanımlayın.

  Tanılama izleme düzeyi gibi kaynak bağdaştırıcısının genel özelliklerini gösteren bu özellikler “ResourceAdapter nesne özellikleri için yapılandırma” sayfa 433içinde açıklanır.
- ActivationSpec nesnesinin özelliklerini tanımlayın.

  Bu özellikler, bir MDB ' nin gelen iletişim için nasıl etkinleştirileceğini belirler. Daha fazla bilgi için bkz “Kaynak bağdaştırıcısının gelen iletişim için yapılandırılması” sayfa 435.
- Bir ConnectionFactory nesnesinin özelliklerini tanımlayın.

  Uygulama sunucusu, giden iletişim için bir JMS ConnectionFactory nesnesi yaratmak üzere bu özellikleri kullanır. Daha fazla bilgi için bkz “Kaynak bağdaştırıcısının giden iletişim için yapılandırılması” sayfa 452.
- Yönetilen hedef nesnenin özelliklerini tanımlayın.

  Uygulama sunucusu, giden iletişim için bir JMS Kuyruk nesnesi ya da JMS Konu nesnesi yaratmak üzere bu özellikleri kullanır. Daha fazla bilgi için bkz “Kaynak bağdaştırıcısının giden iletişim için yapılandırılması” sayfa 452.
- İsteğe bağlı: Kaynak bağdaştırıcısı için bir konuşlandırma planı tanımlayın.

  IBM MQ kaynak bağdaştırıcısı RAR dosyası, kaynak bağdaştırıcısı için bir konuşlandırma tanımlayıcısı içeren `META-INF/ra.xml`adlı bir dosya içerir. Bu konuşlandırma tanımlayıcısı, https://xmlns.jcp.org/xml/ns/javaee/connector_1_7.xsd adresindeki XML şeması tarafından tanımlanır ve kaynak bağdaştırıcısı ve sağladığı hizmetlerle ilgili bilgileri içerir. Bir uygulama sunucusu, kaynak bağdaştırıcısı için bir konuşlandırma planı da gerektirebilir. Bu konuşlandırma planı, uygulama sunucusuna özgüdür.

JVM sistem özelliklerini gerektiği gibi belirtin:

- TLS (Transport Layer Security; İletim Katmanı Güvenliği) kullanıyorsanız, aşağıdaki örnekte olduğu gibi, anahtar deposu dosyasının ve güvenli depo dosyasının yerlerini JVM sistem özellikleri olarak belirtin:

  ```
  java ... -Djavax.net.ssl.keyStore=
  key_store_location
           -Djavax.net.ssl.trustStore=trust_store_location
           -Djavax.net.ssl.keyStorePassword=key_store_password
  ```

  Bu özellikler bir ActivationSpec ya da ConnectionFactory nesnesinin özellikleri olamaz ve bir uygulama sunucusu için birden çok anahtar deposu belirtemezsiniz. Özellikler tüm JVM için geçerlidir ve bu nedenle uygulama sunucusunda çalışan diğer uygulamalar TLS bağlantıları kullanıyorsa uygulama sunucusunu etkileyebilir. Uygulama sunucusu da bu özellikleri farklı değerlere sıfırlayabilir. IBM MQ classes for JMSile TLS kullanma hakkında daha fazla bilgi için bkz. “TLS ' yi IBM MQ classes for JMS ile kullanma” sayfa 245.
- İsteğe bağlı: Gerekiyorsa, kaynak bağdaştırıcısını, uyarı iletilerini uygulama sunucunuzun standart çıkış günlüğüne kaydeden şekilde yapılandırın.

  Kaynak bağdaştırıcısı günlükleri, uyarı ve hata iletileri IBM MQ classes for JMSile aynı mekanizmayı kullanır. Daha fazla bilgi için bkz. IBM MQ classes for JMSiçin hataları günlüğe kaydetme. Bu, varsayılan olarak iletilerin `mqjms.log`adlı bir dosyaya gönderileceği anlamına gelir. Kaynak bağdaştırıcısını, uyarı iletilerini uygulama sunucunuzun standart çıkış günlüğüne kaydeden ek olarak yapılandırmak için, uygulama sunucunuz için aşağıdaki JVM sistem özelliğini ayarlayın:

  ```
  -Dcom.ibm.msg.client.commonservices.log.outputName=mqjms.log,stdout
  ```

  Bu özellik, IBM MQ classes for JMSile ilgili izlemeyi denetlemek için kullanılan özellikle aynıdır. IBM MQ classes for JMSile olduğu gibi, `jms.config` dosyasını gösteren bir sistem özelliği kullanılabilir (bkz. “IBM MQ classes for JMS/Jakarta Messaging yapılandırma dosyası” sayfa 94). JVM sistem özelliğinin nasıl ayarlanacağına ilişkin bilgi için uygulama sunucusu belgelerinize bakın.

Kuruluş doğrulama sınamasını çalıştırmak için kaynak bağdaştırıcısını yapılandırın

- Kaynak bağdaştırıcısını, IBM MQ kaynak bağdaştırıcısıyla birlikte sağlanan kuruluş doğrulama sınamasını (IVT) çalıştıracak şekilde yapılandırın.

  IVT programını çalıştırmak için nelerin yapılandırılması gerektiği hakkında bilgi için bkz. "Kaynak bağdaştırıcısı kuruluşunun doğrulanması" sayfa 470.

  IBM hizmeti,IBM dışı herhangi bir uygulama sunucusunun doğru şekilde yapılandırıldığını belirtmek için bu programın çalıştırılmasını gerektirebileceğinden bu önemlidir.

  Önemli: Programı çalıştırabilmeniz için önce kaynak bağdaştırıcısını yapılandırmanız gerekir.

ResourceAdapter nesne özellikleri için yapılandırma

ResourceAdapter nesnesi, tanılama izleme düzeyi gibi IBM MQ kaynak bağdaştırıcısının genel özelliklerini içerir. Bu özellikleri tanımlamak için, uygulama sunucunuzla birlikte sağlanan belgelerde açıklandığı gibi kaynak bağdaştırıcınızın olanaklarını kullanın.

ResourceAdapter nesnesi iki özellik kümesine sahiptir:

- Tanılama izlemesiyle ilişkili özellikler
- Kaynak bağdaştırıcısı tarafından yönetilen bağlantı havuzuyla ilişkili özellikler

Bu özellikleri tanımlama biçiminiz, uygulama sunucunuzun sağladığı denetim arabirimlerine bağlıdır. WebSphere Application Server traditionalkullanıyorsanız, bkz. "WebSphere Application Server traditional yapılandırması" sayfa 434 ya da WebSphere Libertykullanıyorsanız, bkz. "WebSphere Liberty yapılandırması" sayfa 435. Diğer uygulama sunucuları için, uygulama sunucunuza ilişkin ürün belgelerine bakın.

Tanılama izlemesiyle ilişkili özelliklerin tanımlanmasına ilişkin ek bilgi için IBM MQ Kaynak Bağdaştırıcısının İzlenmesi başlıklı konuya bakın.

Kaynak bağdaştırıcısı, iletileri MDB ' lere teslim etmek için kullanılan JMS bağlantılarından oluşan bir iç bağlantı havuzunu yönetir. Çizelge 63 sayfa 433 içinde, bağlantı havuzuyla ilişkilendirilmiş ResourceAdapter nesnesinin özellikleri listelenir.

Çizelge 63. Bağlantı havuzuyla ilişkilendirilmiş ResourceAdapter nesnesinin özellikleri

Özelliğin adı	Tip	Varsayılan değer	Açıklama
maxConnections	Dizgi	50	Bir IBM MQ kuyruk yöneticisine yönelik bağlantı sayısı üst sınırı ve konuşlandırılan veritabanı sayısı üst sınırı.
connectionConcurrency	Dizgi	1	Bir JMS bağlantısını paylaşmak için veritabanı sayısı üst sınırı. Bağlantı paylaşımı mümkün değildir ve bu özellik her zaman 1 değerine sahiptir.
reconnectionRetrySayısı	Dizgi	5	Bir bağlantı başarısız olursa, kaynak bağdaştırıcısı tarafından bir IBM MQ kuyruk yöneticisine yeniden bağlanma girişimi sayısı üst sınırı.
reconnectionRetryAralığı	Dizgi	300 000	Kaynak bağdaştırıcısının bir IBM MQ kuyruk yöneticisine yeniden bağlanmayı denemeden önce bekleyeceği süre (milisaniye).
startupRetrySayısı	Dizgi	0	Uygulama sunucusu başlatıldığında kuyruk yöneticisi çalışmıyorsa, başlatma sırasında MDB ' ye bağlanma ve bağlanma denemesi için varsayılan sayı.
startupRetryAralığı	Dizgi	30.000	Başlatma bağlantısı girişimleri arasındaki varsayılan uyku süresi (milisaniye).

Çizelge 63. Bağlantı havuzuyla ilişkilendirilmiş ResourceAdapter nesnesinin özellikleri (devamı var)			
Özelliğin adı	**Tip**	**Varsayılan değer**	**Açıklama**
supportMQExtensions	Dizgi	yanlış	IBM MQ JMS davranışınıJMS 2.0 öncesi davranışa geri çevirir. Daha fazla bilgi için bkz "SupportMQExtensions özelliği" sayfa 315.
nativeLibraryYol	Dizgi	<empty>	Bağ tanımlama kipi bağlantılarına izin vermek üzere IBM MQ JNI kitaplığını yüklemek için kullanılacak yol. Windows Windows sistemlerinde sistem yolunun, eşleşen IBM MQ kuruluşunun yerini de içermesi gerekir.

Uygulama sunucusunda bir MDB konuşlandırıldığında, maxConnection özelliğiyle belirtilen bağlantı sayısı üst sınırı aşılmadığı sürece, yeni bir JMS bağlantısı yaratılır ve kuyruk yöneticisiyle bir etkileşim başlatılır. Bu nedenle, veritabanı sayısı üst sınırı bağlantı sayısı üst sınırına eşittir. Konuşlandırılan veritabanı sayısı bu üst sınıra ulaşırsa, başka bir MDB ' yi konuşlandırma girişimi başarısız olur. Bir MDB durdurulursa, bağlantısı başka bir MDB tarafından kullanılabilir.

Genel olarak, birçok veritabanı dağıtılacaksa, maxConnections özelliğinin değerini artırmanız gerekir.

reconnectionRetryCount ve reconnectionRetryAralık özellikleri, bir IBM MQ kuyruk yöneticisine yönelik bağlantılar başarısız olduğunda, örneğin bir ağ hatası nedeniyle kaynak bağdaştırıcısının davranışını yönetir. Bir bağlantı başarısız olduğunda, kaynak bağdaştırıcısı, reconnectionRetryInterval özelliğiyle belirtilen bir aralık için bu bağlantı tarafından sağlanan tüm MDB ' lere ileti teslimini askıya alır. Kaynak bağdaştırıcısı daha sonra kuyruk yöneticisine yeniden bağlanmayı dener. Girişim başarısız olursa, kaynak bağdaştırıcısı, reconnectionRetryCount özelliği tarafından belirtilen sınıra ulaşılıncaya kadar reconnectionRetryInterval özelliği tarafından belirtilen aralıklarla yeniden bağlanma girişiminde bulunur. Tüm girişimler başarısız olursa, MDB ' ler el ile yeniden başlatılıncaya kadar teslim kalıcı olarak durdurulur.

Genel olarak, ResourceAdapter nesnesi denetim gerektirmez. Ancak, örneğin AIX and Linux sistemlerinde tanılama izlemesini etkinleştirmek için aşağıdaki özellikleri ayarlayabilirsiniz:

```
traceEnabled:    true
traceLevel:      10
```

Kaynak bağdaştırıcısı başlatılmamışsa, bu özellikler etkili olmaz; örneğin, IBM MQ kaynaklarını kullanan uygulamalar yalnızca istemci taşıyıcısında çalışıyorsa. Bu durumda, tanılama izlemesinin özelliklerini Java Virtual Machine (JVM) sistem özellikleri olarak ayarlayabilirsiniz. Aşağıdaki örnekte olduğu gibi, **java** komutunda -D işaretini kullanarak özellikleri ayarlayabilirsiniz:

```
java ... -DtraceEnabled=true -DtraceLevel=6
```

ResourceAdapter nesnesinin tüm özelliklerini tanımlamanız gerekmez. Belirtilmeyen özellikler varsayılan değerlerini alır. Yönetilen bir ortamda, özellikleri belirtmenin iki yolunu karıştırmamanız daha iyi olur. Bunları karıştırırsanız, JVM sistem özellikleri ResourceAdapter nesnesinin özelliklerinden önceliklidir.

WebSphere Application Server traditional yapılandırması

WebSphere Application Server traditionaliçinde kaynak bağdaştırıcısı için de aynı özellikler kullanılabilir, ancak bunlar kaynak bağdaştırıcısının özellikler panosunda ayarlanmalıdır (WebSphere Application Server traditional ürün belgelerinde JMS sağlayıcısı ayarları konusuna bakın). İzleme, WebSphere Application Server traditional yapılandırmasının tanılama bölümü tarafından denetlenir. Daha fazla bilgi için WebSphere Application Server traditional ürün belgelerinde Working with Diagnostic Providers (Tanılama Sağlayıcılarıyla Çalışma) başlıklı konuya bakın.

WebSphere Liberty yapılandırması

Kaynak bağdaştırıcısı, aşağıdaki örnekte gösterildiği gibi `server.xml` dosyasındaki XML öğeleri kullanılarak yapılandırılır:

JM 3.0

```
<featureManager>
...
    <feature>messaging-3.0</feature>
...
</featureManager>
    <variable name="wmqJmsClient.rar.location"
        value="F:/_rtc_wmq8005/_build/ship/lib/jca/wmq.jakarta.jmsra.rar"/>
...
        <wmqJmsClient supportMQExtensions="true" logWriterEnabled="true"/>
```

JMS 2.0

```
<featureManager>
...
    <feature>wmqJmsClient-2.0</feature>
...
</featureManager>
    <variable name="wmqJmsClient.rar.location"
        value="F:/_rtc_wmq8005/_build/ship/lib/jca/wmq.jmsra.rar"/>
...
        <wmqJmsClient supportMQExtensions="true" logWriterEnabled="true"/>
```

Bu XML öğesi eklenerek izleme etkinleştirilir:

```
<logging traceSpecification="JMSApi=all:WAS.j2c=all:"/>
```

Kaynak bağdaştırıcısının gelen iletişim için yapılandırılması

Gelen iletişimi yapılandırmak için bir ya da daha çok ActivationSpec nesnesinin özelliklerini tanımlayın.

ActivationSpec nesnesinin özellikleri, iletiyle yönlendirilen bir Bean 'in (MDB) bir IBM MQ kuyruğundan JMS iletilerini nasıl aldığını belirler. MDB ' nin işlem davranışı, konuşlandırma tanımlayıcısında tanımlanır.

ActivationSpec nesnesi iki özellik kümesine sahiptir:

- Bir IBM MQ kuyruk yöneticisine JMS bağlantısı yaratmak için kullanılan özellikler
- Belirli bir kuyruğa geldiklerinde iletileri zamanuyumsuz olarak teslim eden bir JMS bağlantı tüketicisi oluşturmak için kullanılan özellikler

ActivationSpec nesnesinin özelliklerini tanımlama şekliniz, uygulama sunucunuz tarafından sağlanan denetim arabirimlerine bağlıdır.

IBM MQ kuyruk yöneticisine JMS bağlantısı yaratmak için kullanılan özellikler

Çizelge 64 sayfa 436 içindeki tüm özellikler isteğe bağlıdır.

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
applicationName	Dizgi	• Çağıran sınıf adı (varsa), 28 karakterden uzun olmayacak şekilde ayarlanır. Yoksa, `WebSphere MQ Client for Java` dizgisi kullanılır.	Bir uygulamanın kuyruk yöneticisine kayıtlı olduğu ad. Bu uygulama adı **DISPLAY CONN MQSC/PCF** komutuyla gösterilir (burada alan **APPLTAG** olarak adlandırılır) ya da IBM MQ Explorer **Uygulama Bağlantıları** ekranında (alanın **App name** olarak adlandırıldığı yerde).
brokerCCDurSubQueue [1]	Dizgi	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Kuyruk adı	Bağlantı tüketicisinin sürekli abonelik iletileri aldığı kuyruğun adı
brokerCCSubKuyruk [1]	Dizgi	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Kuyruk adı	Bağlantı tüketicisinin sürekli olmayan abonelik iletilerini aldığı kuyruğun adı
brokerControlKuyruk [1]	Dizgi	• **SYSTEM.BROKER.CONTROL.QUEUE** • Kuyruk adı	Aracı denetim kuyruğunun adı
brokerQueueManager [1]	Dizgi	• **"" (boş dizgi)** • Kuyruk yöneticisi adı	Aracının çalıştığı kuyruk yöneticisinin adı
brokerSubKuyruk [1]	Dizgi	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Kuyruk adı	Sürekli olmayan bir ileti tüketicisinin iletileri aldığı kuyruğun adı
brokerVersion [1]	Dizgi	• **belirlenmedi** -Aracı V6 'dan V7' ye geçirildikten sonra, bu özelliği RFH2 üstbilgilerinin artık kullanılmayacak şekilde ayarlayın. Geçişten sonra bu özellik artık ilgili değildir. • **V1** -Bir IBM MQ yayınlama/abone olma broker.This değeri varsayılan değerdir. • **V2** -Yerel kipte IBM Integration Bus aracısını kullanmak için. TRANSPORT değeri DIRECT ya da DIRECTHTTP olarak ayarlandıysa, bu değer varsayılan değerdir.	Kullanılmakta olan aracının sürümü
ccdtURL	Dizgi	• **boş değerli** • Birörnek kaynak yeri belirleyici (URL)	İstemci kanal tanımlama çizelgesini (CCDT) içeren dosyanın adını ve yerini tanımlayan ve dosyaya nasıl erişilebileceğini belirten URL

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
CCSID	Dizgi	• **819** • Java sanal makinesi (JVM) tarafından desteklenen bir kodlanmış karakter takımı tanıtıcısı	Bağlantıya ilişkin kodlanmış karakter takımı tanıtıcısı
kanal	Dizgi	• **SYSTEM.DEF.SVRCONN** • MQI kanalının adı	Kullanılacak MQI kanalının adı
cleanupInterval [1] (Temizleme Aralığı)	int	• **3600 000** • Artı bir tamsayı	Yayınlama/abone olma temizleme yardımcı programının arka plan çalıştırmaları arasındaki milisaniye cinsinden aralık
cleanupLevel [1]	Dizgi	• **GüVENLI** • YOK • güçlü • ZORLA • NONDUR	Aracıya dayalı abonelik deposu için temizleme düzeyi
clientID	Dizgi	• **boş değerli** • Bir istemci tanıtıcısı	Bağlantıya ilişkin istemci tanıtıcısı
cloneSupport	Dizgi	• **DEVRE Dışı** -Bir kerede tek bir sürekli konu abonesi çalıştırılabilir. • ENABLED-Aynı sürekli konu abonesinin iki ya da daha fazla eşgörünümü aynı anda çalışabilir, ancak her yönetim ortamının ayrı bir Java sanal makinesinde (JVM) çalışması gerekir.	Aynı sürekli konu abonesinin iki ya da daha fazla eşgörünümünün aynı anda çalışıp çalışamayacağını belirler
connectionFactoryArama	Dizgi	• **boş değerli** • Bir ConnectionFactory nesnesine ilişkin JNDI adı	Bu özellik ayarlanırsa, ActivationSpec , uygulama sunucusunun JNDI ad alanında belirtilen JNDI adıyla bir JMS ConnectionFactory nesnesini arar ve tek bir kural dışı durumla birlikte bir IBM MQ kuyruk yöneticisine JMS bağlantısı yaratmak için bu nesnenin özelliklerini kullanır. JMS bağlantısı yaratılırken kullanılacak tek ActivationSpec özelliği clientID' dir. Daha fazla bilgi için bkz “ActivationSpec connectionFactoryLookup ve destinationLookup özellikleri” sayfa 448.

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
connectionNameListesi	Dizgi	• **localhost (1414)** • Her öğenin biçimi aldığı, virgüllerle ayrılmış öğelerden oluşan bir dizgi: `HOSTNAME(PORT)` Burada *HOSTNAME* , bir DNS adı ya da bir IP adresidir.	Gelen iletişim için kullanılan TCP/IP bağlantı adlarının listesi. Belirtildiğinde, **connectionNameList** **hostname** ve **port** özelliklerinin yerini alır. Bu özellik, çok eşgörünümlü kuyruk yöneticilerine yeniden bağlanmak için kullanılır. **connectionNameList** , formda **localAddress**ile benzer, ancak bununla karıştırılmamalıdır. **localAddress** , yerel iletişimin özelliklerini belirtirken, **connectionNameList** uzak kuyruk yöneticisine nasıl erişileceğini belirtir.
V 9.3.0 V 9.3.0 dynamicallyBalanced [4]	Boole	• **yanlış** • doğru	Bu MDB ' nin tek tip bir kümede uygulama dengeleme özelliğinin bir parçası olarak farklı bir kuyruk yöneticisinden ileti almasının istenip istenemeyeceğini belirler.
failIfSusturma	Boole	• **doğru** • yanlış	Kuyruk yöneticisi susturma durumundaysa, belirli yöntemlere yönelik çağrıların başarısız olup olmayacağını belirler
headerCompression	Dizgi	• **YOK** • SYSTEM-RLE ileti üstbilgisi sıkıştırması gerçekleştirilir	Bir bağlantıdaki üstbilgi verilerini sıkıştırmak için kullanılabilecek tekniklerin listesi
hostName	Dizgi	• **localhost** • Anasistem adı • Bir IP adresi	Kuyruk yöneticisinin bulunduğu sistemin anasistem adı ya da IP adresi. **hostname** ve **port** özellikleri, belirtildiğinde **connectionNameList** özelliği tarafından değiştirilir.

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
localAddress	Dizgi	• **boş değerli** • Şu biçimde bir dizgi: `[ host_name ][(low_port [, high_port ])]` Burada *anasistem_adı* bir anasistem adı ya da IP adresi, *alt_kapı* ve *yüksek_kapı* TCP kapı numaralarıdır ve köşeli ayraçlar isteğe bağlı bir bileşeni gösterir	Bir kuyruk yöneticisine yönelik bağlantı için, bu özellik aşağıdakilerden birini ya da her ikisini de belirtir: • Kullanılacak yerel ağ arabirimi • Kullanılacak yerel kapı ya da yerel kapı aralığı **localAddress** , formda **connectionNameList**ile benzer, ancak bununla karıştırılmamalıdır. **localAddress** , yerel iletişimin özelliklerini belirtirken, **connectionNameList** uzak kuyruk yöneticisine nasıl erişileceğini belirtir.
messageCompression	Dizgi	• **YOK** • Boş karakterlerle ayrılmış olarak aşağıdaki değerlerden birinin ya da daha fazlasının listesi: RLE ZLIBFAST ZLIBHIGH	Bir bağlantıdaki ileti verilerini sıkıştırmada kullanılabilecek tekniklerin listesi
messageRetention [1]	Boole	• **true** -İstenmeyen iletiler giriş kuyruğunda kalır • false-İstenmeyen iletiler, yok etme seçeneklerine göre ele alınmaz	Bağlantı tüketicisinin giriş kuyruğunda istenmeyen iletileri alıkoyup saklamadığını belirler
messageSelection [1]	Dizgi	• **İstemci** • Aracı	İleti seçiminin IBM MQ classes for JMS tarafından mı, yoksa aracı tarafından mı yapılacağını belirler. brokerVersion 1 değerine sahip olduğunda aracı tarafından ileti seçimi desteklenmez.
parola	Dizgi	• **boş değerli** • Parola	Kuyruk yöneticisine bağlantı yaratılırken kullanılacak varsayılan parola

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
pollingInterval [1]	int	• **5000** • Pozitif bir tamsayı	Bir oturum içindeki her ileti dinleyicinin kuyruğunda uygun bir ileti yoksa, bu değer, her ileti dinleyicisinin kuyruğundan bir ileti almaya yeniden çalışmasından önce geçmesi gereken milisaniye cinsinden aralık üst sınırıdır. Sık sık oturumdaki ileti dinleyicilerden herhangi biri için uygun bir ileti yoksa, bu özelliğin değerini artırmayı düşünebilirsiniz. Bu özellik yalnızca TRANSPORT, BIND ya da CLIENT değerine sahipse geçerlidir.
kapı	int	• **1414** • TCP kapı numarası	Kuyruk yöneticisinin dinlediği kapı. **hostname** ve **port** özellikleri, belirtildiğinde **connectionNameList** özelliği tarafından değiştirilir.
providerVersion	dizgi	• **belirtilmedi** • Aşağıdaki biçimlerden birindeki bir dizgi – V.R.M.F – V.R.M – V.R – V Burada V, R, M ve F sıfırdan büyük ya da sıfıra eşit tamsayı değerleridir.	MDB ' nin bağlanmak istediği kuyruk yöneticisinin sürümü, yayını, değişiklik düzeyi ve düzeltme paketi.
queueManager	Dizgi	• **"" (boş dizgi)** • Kuyruk yöneticisi adı	Bağlanılacak kuyruk yöneticisinin adı
receiveExit [3]	Dizgi	• **boş değerli** • Virgüllerle ayrılmış bir ya da daha çok öğeden oluşan bir dizgi; burada her öğe, IBM MQ classes for Java arabirimini gerçekleştiren bir sınıfın tam olarak nitelenmiş adıdır (MQReceiveExit).	Bir kanal alma çıkış programını ya da art arda çalıştırılacak bir alma çıkış programı sırasını tanımlar
receiveExitBaşlatma	Dizgi	• **boş değerli** • Virgüllerle ayrılmış bir ya da daha fazla kullanıcı verisi öğesini içeren bir dizgi	Kanala geçirilen kullanıcı verileri, çağrıldığında çıkış programlarını alır

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
rescanInterval [1]	int	• **5000** • Pozitif bir tamsayı	Noktadan noktaya iletişim etki alanındaki bir ileti tüketicisi, almak istediği iletileri seçmek için bir ileti seçici kullandığında, IBM MQ classes for JMS , IBM MQ kuyruğunda kuyruğun **MsgDeliverySequence** özniteliği tarafından belirlenen sırada uygun iletileri arar. IBM MQ classes for JMS uygun bir ileti bulduğunda ve bunu tüketiciye teslim ettiğinde IBM MQ classes for JMS , kuyruktaki geçerli konumundan sonraki uygun iletiyi aramaya devam eder. IBM MQ classes for JMS , kuyruğun sonuna ulaşıncaya kadar ya da bu özelliğin değerinin belirlediği milisaniye cinsinden zaman aralığının süresi doluncaya kadar kuyruğu bu şekilde aramaya devam eder. Her durumda IBM MQ classes for JMS , aramaya devam etmek için kuyruğun başlangıcına geri döner ve yeni bir zaman aralığı başlar.
securityExit [3]	Dizgi	• **boş değerli** • IBM MQ classes for Java arabirimini gerçekleştiren bir sınıfın tam olarak nitelenmiş adı, MQSecurityExit	Kanal güvenliği çıkış programını tanımlar
securityExitBaşlatma	Dizgi	• **boş değerli** • Kullanıcı verileri dizgisi	Çağrıldığında bir kanal güvenliği çıkış programına geçirilen kullanıcı verileri
sendExit [3]	Dizgi	• **boş değerli** • Virgüllerle ayrılmış bir ya da daha çok öğeden oluşan bir dizgi; burada her öğe, IBM MQ classes for Java arabirimini gerçekleştiren bir sınıfın tam olarak nitelenmiş adıdır (MQSendExit).	Kanal gönderme çıkış programını ya da art arda çalıştırılacak çıkış gönderme programlarını tanımlar
sendExitInit	Dizgi	• **boş değerli** • Virgüllerle ayrılmış bir ya da daha fazla kullanıcı verisi öğesini içeren bir dizgi	Çağrıldığında çıkış programlarını kanala gönderen kullanıcı verileri

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)			
Özelliğin adı	**Tip**	**Geçerli değerler (koyu varsayılan değer)**	**Açıklama**
shareConvİzin Verilir	Boole	• NO-Bir istemci bağlantısı yuvasını paylaşamıyor. • **YES** (EVET)-Bir istemci bağlantısı yuvasını paylaşabilir.	Bir istemci bağlantısının, kanal tanımlamaları eşleşiyorsa, aynı işlemden aynı kuyruk yöneticisine yapılan diğer üst düzey JMS bağlantılarıyla yuvasını paylaşıp paylaşamayacağını belirler.
sparseSubscriptions [1]	Boole	• **false** -Abonelikler sık sık eşleşen iletiler alır. • true-Abonelikler nadiren eşleşen iletiler alır. Bu değer, abonelik kuyruğunun göz atmak için açılabilmesini gerektirir.	TopicSubscriber nesnesinin ileti alma ilkesini denetler
sslCertMağazaları	Dizgi	• **boş değerli** • Boşluklarla ayrılmış bir ya da daha çok LDAP URL dizesi. Her LDAP URL şu biçimdedir: `ldap://host_name [: port ]` Burada *anasistem_adı* bir anasistem adı ya da IP adresidir, *kapı* bir TCP kapı numarasıdır ve köşeli parantezler isteğe bağlı bir bileşeni gösterir.	TLS bağlantısında kullanılmak üzere sertifika iptal listelerini (CRL) bulunduran LDAP sunucuları
sslCipherÜrün Grubu	Dizgi	• **boş değerli** • CipherSuite adı	TLS bağlantısı için kullanılacak CipherSuite
sslFipsGerekli [2]	Boole	• **yanlış** • doğru	TLS bağlantısının IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) tarafından desteklenen bir CipherSuite kullanması gerekip gerekmediğini belirler
sslPeerAdı	Dizgi	• **boş değerli** • Ayırt edici adlar için şablon	TLS bağlantısı için, kuyruk yöneticisi tarafından sağlanan sayısal sertifikada ayırt edici adı denetlemek için kullanılan bir şablon
sslResetSayısı	int	• **ERROR! SEGMENT DATA CORRUPTED, SEGDATA=0** • 0-999 999 999 aralığında bir tamsayı	TLS tarafından kullanılan gizli anahtarlar yeniden anlaşılmadan önce bir TLS bağlantısı tarafından gönderilen ve alınan toplam bayt sayısı

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
sslSocketÜreticisi	Dizgi	javax.net.ssl.SSLSocketFactory arabiriminin somutlamasını sağlayan bir sınıfın tam olarak nitelenmiş sınıf adını gösteren dizgi. İsteğe bağlı olarak, oluşturucu yöntemine geçirilecek ve ayraç içine alınmış bir bağımsız değişken de içerilir.	Yönetilen nesne kapsamında kurulan tüm bağlantılar, SSLSocketFactory arabiriminin bu somutlamasından elde edilen yuvaları kullanır.
statusRefreshInterval [1] (Durum Yenileme)	int	• **60000** • Pozitif bir tamsayı	Bir abonenin kuyruk yöneticisine olan bağlantısını kaybettiği zaman algılanan, uzun süren hareketin yenilenmesi arasındaki milisaniye cinsinden aralık. Bu özellik yalnızca **subscriptionStore** QUEUEdeğerine sahipse geçerlidir.
subscriptionStore [1]	Dizgi	• **Aracı** • GEÇİŞ YAPIN • kuyruk	IBM MQ classes for JMS etkin aboneliklerle ilgili kalıcı verilerin nerede saklanacağını belirler

Çizelge 64. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
transportType	Dizgi	• **İstemci** • Bağ Tanımları • BINDINGS_THEN_CLIENT	Bir kuyruk yöneticisine yönelik bağlantının istemci kipini mi, yoksa bağ tanımlama kipini mi kullandığını belirler. `BINDINGS_THEN_CLIENT` değeri belirtilirse, kaynak bağdaştırıcısı önce bağ tanımlama kipinde bağlantı kurmaya çalışır. Bu bağlantı başarısız olursa, kaynak bağdaştırıcısı istemci kipi bağlantısı kurmayı dener. z/OS Bir WebSphere Application Server for z/OS sisteminde çalışan bir etkinleştirme belirtimi `BINDINGS_THEN_CLIENT` iletim kipini kullanacak şekilde yapılandırıldıysa ve önceden kurulmuş bir bağlantı kesildiyse, etkinleştirme belirtiminin yeniden bağlanma girişimleri önce `BINDINGS` iletim kipini kullanmayı dener. `BINDINGS` iletim kipi bağlantı girişimi başarısız olursa, etkinleştirme belirtimi daha sonra `CLIENT` iletim kipi bağlantısını dener.
kullanıcı adı	Dizgi	• **boş değerli** • Kullanıcı adı	Bir kuyruk yöneticisine bağlantı yaratırken kullanılacak varsayılan kullanıcı adı
wildcardFormat	Dizgi	• CHAR-Aracı sürüm 1 'de kullanıldığı şekilde, yalnızca genel arama karakterlerini tanır • **TOPIC** -Aracı sürüm 2 'de kullanıldığı şekilde, yalnızca konu düzeyi genel arama karakterlerini tanır.	Genel arama karakteri sözdiziminin hangi sürümünün kullanılacağı

Notlar:

1. Bu özellik, IBM MQ classes for JMS' un 70 sürümüyle kullanılabilir.
2. sslFipsRequired özelliğini kullanma hakkında önemli bilgi için bkz. "IBM MQ kaynak bağdaştırıcısının sınırlamaları" sayfa 423.
3. Kaynak bağdaştırıcısının bir çıkış bulabilmesi için nasıl yapılandırılacağı hakkında bilgi için bkz. "IBM MQ classes for JMS ' nin kanal çıkışlarını kullanacak şekilde yapılandırılması" sayfa 271.

4. V 9.3.0 dynamicallyBalanced özelliği, XA hareket desteğiyle birlikte desteklenmez. dynamicallyBalanced "true" ise, MDB, XA işlemlerini devre dışı bırakacak şekilde yapılandırılmalıdır.

JMS bağlantı tüketicisi oluşturmak için kullanılan özellikler

Not: `destination` ve **`destinationType`** belirtik olarak tanımlanmalıdır. Çizelge 65 sayfa 445 içindeki diğer tüm özellikler isteğe bağlıdır.

Çizelge 65. JMS bağlantı tüketicisi yaratmak için kullanılan ActivationSpec nesnesinin özellikleri

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
Hedef	Dizgi	Hedef adı	İletilerin alınacağı hedef. **`useJNDI`** özelliği, bu özelliğin değerinin nasıl yorumlanacağını belirler.
destinationLookup	Dizgi	• **boş değerli** • Hedef nesneye ilişkin JNDI adı	Bu özellik ayarlanırsa, ActivationSpec , uygulama sunucusunun JNDI ad alanında belirtilen JNDI adına sahip bir JMS Hedef nesnesini arar ve daha sonra, ActivationSpeciçinde belirtilen diğer özelliklere tercih ederek JMS bağlantı tüketicisi yaratmak için o nesnenin özelliklerini kullanır. Daha fazla bilgi için bkz “ActivationSpec connectionFactoryLookup ve destinationLookup özellikleri” sayfa 448.
destinationType	Dizgi	• V 9.3.0 V 9.3.0 jakarta.jms.Queue (Jakarta Messaging 3.0) • V 9.3.0 V 9.3.0 jakarta.jms.Topic (Jakarta Messaging 3.0) • javax.jms.Queue (JMS 2.0) • javax.jms.Topic (JMS 2.0)	Hedef, kuyruk ya da konu tipi
maxMessages	int	• **1** • Artı bir tamsayı	Bir sunucu oturumuna aynı anda atanabilecek ileti sayısı üst sınırı. Etkinleştirme belirtimi bir XA hareketindeki bir MDB ' ye ileti sağlıyorsa, bu özelliğin ayarından bağımsız olarak 1 değeri kullanılır.
maxPoolDerinlik	int	• **10** • Artı bir tamsayı	Bağlantı tüketicisi tarafından kullanılan sunucu oturumu havuzundaki sunucu oturumu sayısı üst sınırı
messageSelector	Dizgi	• **boş değerli** • SQL92 ileti seçici ifadesi	Teslim edilecek iletileri belirten bir ileti seçici ifadesi

Çizelge 65. JMS bağlantı tüketicisi yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
nonASFTimeout	int	• **ERROR! SEGMENT DATA CORRUPTED, SEGDATA=0** • Artı bir tamsayı	Pozitif bir değer, ASF dışı teslimatın kullanıldığını gösterir. Değer, bir alma isteğinin henüz ulaşmamış olabilecek iletileri (bekleme çağrısı ile alma) bekleyeceği süreyi milisaniye cinsinden belirtir. Varsayılan değer olan 0, ASF tesliminin kullanıldığını gösterir. Bu parametre aşağıdaki durumda geçerlidir: • Uygulama, WebSphere Application Server 7.0 ya da daha sonraki bir yayın düzeylerinde çalışıyor. • Uygulama, wmqJmsClient özelliğinin uygun düzeyini kullanarak WebSphere Liberty içinde çalışıyor. Daha fazla bilgi için bkz "Liberty ve IBM MQ kaynak bağdaştırıcısı" sayfa 425.
nonASFRollbackEtkin	Boole	• **false** -MDB başarısız olsa da ileti tüketilir. • true-MDB içindeki hata, iletinin kuyruğa geri dönmesine neden olur.	MDB işlemden geçmediyse, ileti tesliminin bir IBM MQ eşitleme noktası içinde olup olmadığını belirtir. MDB işlem yapıldıysa ya da **nonASFTimeout** , 0olarak ayarlandıysa yoksayılır.
poolTimeout	int	• **300000** • Artı bir tamsayı	Kullanılmayan bir sunucu oturumunun, etkinlik dışı durum nedeniyle kapatılmadan önce sunucu oturumu havuzunda açık tutulduğu süre (milisaniye)
readAheadİzin verilir	int	• **DESTINATION** -Kuyruk ya da konu tanımlamasına başvurarak önden okumaya izin verilip verilmediğini belirleyin. • DISABLED-Önden okumaya izin verilmez. • ENABLED-Önden okumaya izin verilir. • QUEUE-Kuyruk tanımına başvurarak önden okumaya izin verilip verilmediğini belirleyin. • KONU-Konu tanımlamasına başvurarak önden okumaya izin verilip verilmediğini belirleyin.	Etkinleştirme belirtimine göz atma iş parçacığının, yıkıcı kullanım için sunucu oturumlarını kapatmadan önce, hedeften iç arabelleğe birden çok iletiye göz atmak için önden okuma özelliğini kullanıp kullanmasına izin verilip verilmediğini belirler. **Not:** Önden okuma özelliğinin etkinleştirilmesi, JMSCC0108 iletilerinin ya da performansta bir artışa ya da MDB işleme hızının hedeften gelen göz atma iletilerinin hızına ayak uyduramamasına neden olabilir.

Çizelge 65. JMS bağlantı tüketicisi yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
readAheadClosePolicy	int	• **ALL** -İç önden okuma arabelleğindeki tüm iletiler, durdurulmadan önce MDB ' ye teslim edilir. • CURRENT-İletileri daha sonra atılacak iç önden okuma arabelleğinde bırakarak, yalnızca yürürlükteki MDB çağrısı tamamlanır.	MDB yönetici tarafından durdurulduğunda iç önden okuma arabelleğindeki iletilere ne olur?
receiveCCSID	int	• **0** -JVM kullan `Charset.defaultCharset` • 1208- UTF-8 • Desteklenen bir kodlanmış karakter takımı tanıtıcısı	Kuyruk yöneticisi ileti dönüştürmesi için hedef CCSID ' yi belirleyen hedef özellik. **`receiveConversion`** , `QMGR`olarak ayarlanmadıkça değer yoksayılır.
receiveConversion	Dizgi	• **CLIENT_MSG** • QMGR	Veri dönüştürme işleminin kuyruk yöneticisi tarafından gerçekleştirilip gerçekleştirilmeyeceğini belirleyen hedef özellik.
sharedSubscription	Boole	• **False** -MDB, aboneliği paylaşılan abonelik olarak açmamalıdır. • True-MDB, aboneliği paylaşılan abonelik olarak açmalıdır (JMS 2.0 ' in belirttiği kurallarla birlikte, Java.netadresindeki JMS 2.0 belirtimine bakın).	Bir MDB ' nin paylaşılan bir abonelikten nasıl yönlendirildiğini denetler. Bu özelliğin nasıl kullanılacağına ilişkin daha fazla bilgi için bkz. “sharedSubscription özelliğinin nasıl tanımlanacağına ilişkin örnekler” sayfa 451.
startTimeout	int	• **10 000** • Artı bir tamsayı	İletiyi teslim etmek için çalışmanın zamanlanmasından sonra bir iletinin MDB ' ye tesliminin başlaması gereken süre (milisaniye). Bu süre geçerse, ileti kuyruğa geri alınır.
subscriptionDurability	Dizgi	• **NonDurable** -Sürekli olmayan abonelik, konuya abone olan bir MDB ' ye ileti teslim etmek için kullanılır. • Sürekli-Sürekli abonelik, konuya abone olan bir MDB ' ye ileti göndermek için kullanılır.	Konuya abone olan bir MDB ' ye ileti teslim etmek için sürekli ya da sürekli olmayan bir aboneliğin kullanılıp kullanılmayacağı
subscriptionName	Dizgi	• **"" (boş dizgi)** • Abonelik adı	Sürekli aboneliğin adı

Çizelge 65. JMS bağlantı tüketicisi yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)			
Özelliğin adı	**Tip**	**Geçerli değerler (koyu varsayılan değer)**	**Açıklama**
useJNDI	Boole	• **false** -Hedef adı verilen özellik, bir IBM MQ kuyruğunun ya da konusunun adı olarak yorumlanır. • true-Hedef adı verilen özellik, uygulama sunucusunun JNDI ad alanında aşağıdaki nesnelerden birinin adı olarak yorumlanır: – V 9.3.0 V 9.3.0 jakarta.jms.Queue (Jakarta Messaging 3.0) – V 9.3.0 V 9.3.0 jakarta.jms.Topic (Jakarta Messaging 3.0) – javax.jms.Queue (JMS 2.0) – javax.jms.Topic (JMS 2.0)	Deprecated Hedef adı verilen özelliğin değerinin nasıl yorumlanacağını belirler **Not:** Bu özellik IBM MQ 9.0içinde kullanımdan kaldırılmıştır. Bunun yerine destinationLookup özelliği kullanılmalıdır.

Özellik çakışmaları ve bağımlılıkları

ActivationSpec nesnesi çakışan özelliklere sahip olabilir. Örneğin, bağ tanımlama kipinde bir bağlantı için TLS özelliklerini belirtebilirsiniz. Bu durumda davranış, **destinationType** özelliği tarafından belirlendiği şekilde, noktadan noktaya iletişim ya da yayınlama/abone olma olan iletim tipi ve ileti alışverişi etki alanı tarafından belirlenir. Belirtilen iletim tipi ya da ileti alışverişi etki alanı için geçerli olmayan özellikler yoksayılır.

Diğer özelliklerin tanımlanmasını gerektiren bir özellik tanımlarsanız, ancak bu diğer özellikleri tanımlamazsanız, ActivationSpec nesnesi bir MDB ' nin konuşlandırılması sırasında validate () yöntemi çağrıldığında bir InvalidPropertykural dışı durumu verir. Kural dışı durum, uygulama sunucusunun yöneticisine uygulama sunucusuna bağlı olarak bildirilir. Örneğin, subscriptionDurability özelliğini Durable olarak ayarlarsanız, sürekli abonelikler kullanmak istediğinizi belirtirseniz, **subscriptionName** özelliğini de tanımlamanız gerekir.

ccdtURL ve **channel** adlı özelliklerin her ikisi de tanımlandıysa, InvalidPropertykural dışı durumu yayınlanır. Ancak, yalnızca **ccdtURL** özelliğini tanımlarsanız, **channel** adlı özelliği SYSTEM.DEF.SVRCONN, kural dışı durum yayınlanmaz ve bir JMS bağlantısını başlatmak için **ccdtURL** özelliğiyle tanıtılan istemci kanal tanımlama çizelgesi kullanılır.

ActivationSpec connectionFactoryLookup ve destinationLookup özellikleri

JMS 2.0 belirtimi iki yeni ActivationSpec özelliğini tanıttı. connectionFactoryLookup ve destinationLookup özellikleri, diğer ActivationSpec özelliklerine göre kullanılacak yönetilen bir nesnenin JNDI adıyla birlikte sağlanabilir.

Örneğin, JNDI içinde bir bağlantı üreticisi tanımlanmışsa ve bu nesnenin JNDI adı bir etkinleştirme belirtimine ilişkin connectionFactoryArama özelliğinde belirtilirse, JNDI içinde tanımlanan bağlantı üreticisinin tüm özellikleri, Çizelge 64 sayfa 436içindeki özelliklerin yerine kullanılır.

JNDI içinde bir hedef tanımlıysa ve JNDI adı ActivationSpec' in destinationLookup özelliğinde ayarlandıysa, Çizelge 65 sayfa 445içindeki değerlere tercih etmek için kullanılan değerler kullanılır. Bu iki özelliğin nasıl kullanıldığıyla ilgili daha fazla bilgi için bkz. “ActivationSpec connectionFactoryLookup ve destinationLookup özellikleri” sayfa 448.

Bu iki özellik, Çizelge 64 sayfa 436 ve Çizelge 65 sayfa 445içinde tanımlandığı şekilde ActivationSpec özelliklerinin tercih edilmesinde kullanılan ConnectionFactory ve Hedef nesnelerin JNDI adlarını belirtmek için kullanılabilir.

Bu özelliklerin ayrıntılı olarak nasıl çalıştığını açıklayan aşağıdaki noktalara dikkat edilmesi önemlidir.

connectionFactoryArama
JNDI tarafından aranan ConnectionFactory , Çizelge 64 sayfa 436içinde listelenen özelliklerin kaynağı olarak kullanılır. ConnectionFactory nesnesi gerçekte herhangi bir JMS bağlantısı yaratmak için kullanılmaz, yalnızca nesnenin özellikleri sorgulanır. ConnectionFactory içindeki bu özellikler, ActivationSpeciçinde tanımlanan özellikleri geçersiz kılar. Bunun tek bir istisnası var. ActivationSpec için **ClientID** özelliği ayarlandıysa, bu özelliğin değeri ConnectionFactory' de belirtilen değeri geçersiz kılar. Bunun nedeni, ortak bir senaryonun birden çok ActivationSpecsiçeren tek bir ConnectionFactory kullanmasıdır. Bu, yönetimi basitleştirir. Ancak JMS 2.0 belirtimi, ConnectionFactory ' den oluşturulan her JMS Connection 'da benzersiz bir **ClientID**olması gerektiğini belirtir. Bu nedenle ActivationSpecs , ConnectionFactory' de ayarlanan herhangi bir değeri geçersiz kılma yeteneğine sahip olmalıdır. ActivationSpecüzerinde **ClientID** ayarlanmazsa, bağlantı üreticisinin herhangi bir değeri kullanılır.

destinationLookup
ActivationSpeciçinde bir **Destination** ve bir **UseJndi** özelliği tanımlanır. **UseJndi** işareti `true`olarak ayarlanırsa, hedef özellikte belirtilen metin bir JNDI adı olarak kabul edilir ve bu JNDI adına sahip bir hedef nesne JNDI' den aranır.

destinationLookup özelliği tam olarak aynı şekilde davranır. Ayarlandıysa, özellik tarafından belirtilen JNDI adına sahip bir hedef nesne JNDItarafından aranır. Bu özellik, **useJNDI** özelliğinden önceliklidir.

Deprecated useJNDI özelliği, **destinationLookup** özelliği JMS 2.0 belirtimi ya da aynı işlevi gerçekleştirmenin daha sonraki bir eşdeğeri olduğundan IBM MQ 9.0 adresinde kullanımdan kaldırılmıştır.

IBM MQ classes for JMS içinde eşdeğeri olmayan ActivationSpec özellikleri

ActivationSpec nesnesinin özelliklerinin çoğu, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messaging nesnelerinin özelliklerine ya da IBM MQ classes for JMS IBM MQ classes for Jakarta Messaging yöntemlerinin parametrelerine eşdeğerdir. Ancak, IBM MQ classes for JMS ya da IBM MQ classes for Jakarta Messagingiçinde üç ayarlama özelliği ve bir kullanılabilirlik özelliği eşdeğeri yoktur:

startTimeout
Kaynak bağdaştırıcısı bir İş nesnesini bir MDB ' ye ileti gönderecek şekilde zamanladıktan sonra, uygulama sunucusunun iş yöneticisinin kaynakların kullanılabilir olmasını bekleyeceği süre (milisaniye). İleti teslimi başlamadan önce bu süre geçerse, İş nesnesi zamanaşımına uğrarsa, ileti kuyruğa geri alınır ve kaynak bağdaştırıcısı iletiyi yeniden teslim etmeye çalışabilir. Etkinleştirildiyse, tanılama izlemesine bir uyarı yazılır, ancak ileti teslim etme işlemini başka bir şekilde etkilemez. Bu koşulun yalnızca uygulama sunucusu çok yüksek bir yüke maruz kaldığında oluşmasını bekleyebilirsiniz. Koşul düzenli olarak oluşursa, iş yöneticisine ileti teslimini zamanlaması için daha uzun süre vermek üzere bu özelliğin değerini artırmayı düşünün.

maxPoolDerinlik
Bir bağlantı tüketicisi tarafından kullanılan sunucu oturumu havuzundaki sunucu oturumu sayısı üst sınırı. Bir sunucu oturumu yaratıldığında, bir kuyruk yöneticisiyle etkileşim başlatır. Bağlantı tüketicisi, bir iletiyi MDB ' ye teslim etmek için bir sunucu oturumu kullanır. Daha büyük bir havuz derinliği, yüksek hacimli durumlarda eşzamanlı olarak daha fazla ileti gönderilmesini sağlar, ancak uygulama sunucusunun daha fazla kaynağını kullanır. Birçok veritabanı konuşlandırılacaksa, uygulama sunucusundaki yükü yönetilebilir bir düzeyde tutmak için havuz derinliğini küçültmeyi düşünün. Her bağlantı tüketicisi kendi sunucu oturumu havuzunu kullanır; bu özellik, tüm bağlantı tüketicilerinin kullanabileceği toplam sunucu oturumu sayısını tanımlamaz.

poolTimeout
Kullanılmayan bir sunucu oturumunun, etkinlik dışı durum nedeniyle kapatılmadan önce sunucu oturumu havuzunda açık tutulduğu süre (milisaniye). İleti iş yükünde geçici bir artış, yükü dağıtmak

için ek sunucu oturumlarının yaratılmasına neden olur, ancak ileti iş yükü normale döndükten sonra ek sunucu oturumları havuzda kalır ve kullanılmaz.

Bir sunucu oturumu her kullanıldığında, bir zaman damgasıyla işaretlenir. Belirli aralıklarla bir leştirici iş parçacığı, her sunucu oturumunun bu özellik tarafından belirtilen süre içinde kullanılıp kullanılmadığını denetler. Sunucu oturumu kullanılmamışsa, kapatılır ve sunucu oturumu havuzundan kaldırılır. Belirtilen süre geçtikten hemen sonra bir sunucu oturumu kapatılamayabilir; bu özellik, kaldırma işleminden önce boşta durma süresi alt sınırını gösterir.

useJNDI
Bu özelliğin açıklaması için bkz. Çizelge 65 sayfa 445.

MDB ' yi konuşlandırma

Bir MDB 'yi konuşlandırmak için, önce MDB' nin gerektirdiği özellikleri belirterek ActivationSpec nesnesinin özelliklerini tanımlayın. Aşağıdaki örnek, belirtik olarak tanımlayabileceğiniz tipik bir özellik kümeidir:

V 9.3.0 V 9.3.0 JM 3.0

```
channel:           SYSTEM.DEF.SVRCONN
destination:       SYSTEM.DEFAULT.LOCAL.QUEUE
destinationType:   jakarta.jms.Queue
hostName:          192.168.0.42
messageSelector:   color='red'
port:              1414
queueManager:      ExampleQM
transportType:     CLIENT
```

JMS 2.0

```
channel:           SYSTEM.DEF.SVRCONN
destination:       SYSTEM.DEFAULT.LOCAL.QUEUE
destinationType:   javax.jms.Queue
hostName:          192.168.0.42
messageSelector:   color='red'
port:              1414
queueManager:      ExampleQM
transportType:     CLIENT
```

Uygulama sunucusu, daha sonra bir MDB ile ilişkilendirilmiş bir ActivationSpec nesnesi yaratmak için özellikleri kullanır. ActivationSpec nesnesinin özellikleri, iletilerin MDB ' ye nasıl teslim edileceğini belirler. MDB dağıtımlı hareketler gerektiriyorsa, ancak kaynak bağdaştırıcısı dağıtılmış hareketleri desteklemiyorsa MDB ' nin konuşlandırılması başarısız olur. Dağıtılmış hareketlerin desteklenmesi için kaynak bağdaştırıcısının nasıl kurulacağına ilişkin bilgi için bkz. “IBM MQ kaynak bağdaştırıcısının takılması” sayfa 427.

Aynı hedeften birden fazla MDB ileti alıyorsa, diğer MDB ' ler iletiyi almaya uygun olsa bile, noktadan noktaya iletişim etki alanında gönderilen bir ileti yalnızca bir MDB tarafından alınır. Özellikle, iki MDB farklı ileti seçiciler kullanıyorsa ve gelen bir ileti her iki ileti seçiciyle eşleşiyorsa, iletiyi yalnızca bir MDB alır. Bir iletiyi almak için seçilen MDB tanımlı değil ve iletiyi alan belirli bir MDB ' ye güvenemezsiniz. Yayınlama/abone olma etki alanında gönderilen iletiler, tüm uygun veritabanı yöneticileri tarafından alınır.

Bazı durumlarda, MDB ' ye teslim edilen bir ileti IBM MQ kuyruğuna geri işlenebilir. Bu geri alma işlemi, örneğin, daha sonra geriye işlenen bir iş birimi içinde bir ileti teslim edilirse gerçekleşebilir. Geriye işlenen bir ileti yeniden teslim edilir, ancak hatalı biçimlendirilmiş bir ileti defalarca bir MDB ' nin başarısız olmasına neden olabilir ve bu nedenle teslim edilemez. Böyle bir mesaj zehirli mesaj olarak adlandırılır. IBM MQ 'i, IBM MQ classes for JMS ' in daha ayrıntılı inceleme için bir zehirli iletiyi otomatik olarak başka bir kuyruğa aktarması ya da iletiyi atması için yapılandırabilirsiniz.

Zehirli iletilerin nasıl işleneceğine ilişkin ayrıntılar için bkz. “IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 225.

İlgili kavramlar

MQI istemcisinde çalıştırma zamanında yalnızca FIPS onaylı CipherSpecs kullanılmasının belirtilmesi
AIX, Linux, and Windows için Federal Bilgi İşleme Standartları (FIPS)

İlgili görevler

WebSphere Application Server ' de JMS kaynaklarının yapılandırılması

sharedSubscription özelliğinin nasıl tanımlanacağına ilişkin örnekler

Bir WebSphere Liberty `server.xml` dosyasında etkinleştirme belirtiminin sharedSubscription özelliğini tanımlayabilirsiniz. Diğer bir seçenek olarak, ek açıklamaları kullanarak özelliği iletiyle yönlendirilen bir bean (MDB) içinde tanımlayabilirsiniz.

Örnek: Liberty `server.xml` dosyasında tanımlama

Bir WebSphere Liberty `server.xml` dosyasında, aşağıdaki örnekte gösterildiği gibi bir etkinleştirme belirtimi tanımlarsınız. Bu örnek, yerel anasistem/kapı 1490 'da bir kuyruk yöneticisine kalıcı bir paylaşılan abonelik yaratır.

```
<jmsActivationSpec id="SubApp/SubscribingEJB/SubscribingMDB" authDataRef="JMSConnectionAlias">
<properties.wmqJms hostName="localhost" port="1490" maxPoolDepth="5"
subscriptionName="MySubName"
subscriptionDurability="DURABLE" sharedSubscription="true"/>
</jmsActivationSpec>
```

Örnek: MDB içinde tanımlama

Aşağıdaki örnekte gösterildiği gibi, ek açıklamaları kullanarak MDB içinde sharedSubscription özelliğini de tanımlayabilirsiniz:

```
@ActioncationConfigProperty(propertyName ="sharedSubscription",
propertyValue = "true")
```

Aşağıdaki örnekte, ek açıklamalar yöntemini kullanan bir MDB kodu parçası gösterilmektedir:

V 9.3.0 V 9.3.0 JM 3.0

```
/**
 * Message-Driven Bean example using Annotations for configuration
 */
@MessageDriven(
  activationConfig = {
     @ActivationConfigProperty(
        propertyName = "destinationType", propertyValue = "jakarta.jms.Topic"),
     @ActivationConfigProperty(
        propertyName = "sharedSubscription", propertyValue = "TRUE"),
     @ActivationConfigProperty(
        propertyName = "destination", propertyValue = "JNDI_TOPIC_NAME")
   },
   mappedName = "Stock/IBM")
public class SubscribingMDB implements MessageListener {

   // Default constructor.
   public SubscribingMDB() {
   }

   // @see MessageListener#onMessage(Message)
   public void onMessage(Message message) {
      // implement business logic here
   }

}
```

JMS 2.0

```
/**
 * Message-Driven Bean example using Annotations for configuration
 */
@MessageDriven(
  activationConfig = {
     @ActivationConfigProperty(
        propertyName = "destinationType", propertyValue = "javax.jms.Topic"),
     @ActivationConfigProperty(
        propertyName = "sharedSubscription", propertyValue = "TRUE"),
```

```
      @ActivationConfigProperty(
         propertyName = "destination", propertyValue = "JNDI_TOPIC_NAME")
   },
   mappedName = "Stock/IBM")
public class SubscribingMDB implements MessageListener {

   // Default constructor.
   public SubscribingMDB() {
   }

   // @see MessageListener#onMessage(Message)
   public void onMessage(Message message) {
      // implement business logic here
   }

}
```

İlgili kavramlar

Aboneler ve abonelikler

Abonelik dayanıklılığı

“Klonlanan ve paylaşılan abonelikler” sayfa 313

IBM MQ 8.0 ya da daha sonra, birden çok tüketicinin aynı aboneliğe erişmesine izin vermek için iki yöntem vardır. Bu iki yöntem, klonlanmış abonelikler kullanılarak ya da paylaşılan abonelikler kullanılarak sağlanır.

Kaynak bağdaştırıcısının giden iletişim için yapılandırılması

Giden iletişim yapılandırmak için bir ConnectionFactory nesnesinin özelliklerini ve denetlenen bir hedef nesneyi tanımlayın.

Giden iletişimi kullanma örneği

Giden iletişimi kullanırken, uygulama sunucusunda çalışan bir uygulama bir kuyruk yöneticisiyle bağlantı başlatır ve kuyruklarına ileti gönderir ve kuyruklarından iletileri zamanuyumlu olarak alır. Örneğin, doGet() sunucu uygulamacığı yöntemi giden iletişimi kullanır:

V 9.3.0 V 9.3.0 JM 3.0

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {

...

// Look up ConnectionFactory and Queue objects from the JNDI namespace

     InitialContext ic = new InitialContext();
     ConnectionFactory cf = (jakarta.jms.ConnectionFactory) ic.lookup("myCF");
     Queue q = (jakarta.jms.Queue) ic.lookup("myQueue");

// Create and start a connection

     Connection c = cf.createConnection();
     c.start();

// Create a session and message producer

     Session s = c.createSession(false, Session.AUTO_ACKNOWLEDGE);
     MessageProducer pr = s.createProducer(q);

// Create and send a message

     Message m = s.createTextMessage("Hello, World!");
     pr.send(m);

// Create a message consumer and receive the message just sent

     MessageConsumer co = s.createConsumer(q);
     Message mr = co.receive(5000);

// Close the connection

      c.close();
}
```

JMS 2.0

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {

...

// Look up ConnectionFactory and Queue objects from the JNDI namespace

     InitialContext ic = new InitialContext();
     ConnectionFactory cf = (javax.jms.ConnectionFactory) ic.lookup("myCF");
     Queue q = (javax.jms.Queue) ic.lookup("myQueue");

// Create and start a connection

     Connection c = cf.createConnection();
     c.start();

// Create a session and message producer

     Session s = c.createSession(false, Session.AUTO_ACKNOWLEDGE);
     MessageProducer pr = s.createProducer(q);

// Create and send a message

     Message m = s.createTextMessage("Hello, World!");
     pr.send(m);

// Create a message consumer and receive the message just sent

     MessageConsumer co = s.createConsumer(q);
     Message mr = co.receive(5000);

// Close the connection

      c.close();
}
```

Sunucu uygulaması bir HTTP GET isteği aldığında, JNDI ad alanından bir ConnectionFactory nesnesi ve bir Kuyruk nesnesi alır ve bir IBM MQ kuyruğuna ileti göndermek için nesneleri kullanır. Daha sonra sunucu uygulaması gönderdiği iletiyi alır.

Giden iletişim için gereken kaynaklar

Giden iletişimi yapılandırmak için aşağıdaki kategorilerde Java EE Connector Architecture (JCA) kaynaklarını tanımlayın:

- Uygulama sunucusunun bir JMS ConnectionFactory nesnesi oluşturmak için kullandığı ConnectionFactory nesnesinin özellikleri.
- Uygulama sunucusunun bir JMS Kuyruk nesnesi ya da JMS Konu nesnesi yaratmak için kullandığı denetlenen hedef nesnenin özellikleri.

Bu özellikleri tanımlama biçiminiz, uygulama sunucunuz tarafından sağlanan yönetim arabirimlerine bağlıdır. Uygulama sunucusu tarafından yaratılan ConnectionFactory, Kuyruk ve Konu nesneleri, bir uygulama tarafından alınabilecekleri bir JNDI ad alanına bağlanır.

Genellikle, uygulamaların bağlanması gerekebilecek her kuyruk yöneticisi için bir ConnectionFactory nesnesi tanımlarsınız. Uygulamaların noktadan noktaya iletişim etki alanında erişmesi gerekebilecek her kuyruk için bir Kuyruk nesnesi tanımlarsınız. Ve uygulamaların yayınlamak ya da abone olmak isteyebileceği her konu için bir Konu nesnesi tanımlarsınız. ConnectionFactory nesnesi etki alanından bağımsız olabilir. Diğer bir seçenek olarak, noktadan noktaya iletişim etki alanı için etki alanına özgü bir QueueConnectionFactory nesnesi ya da yayınlama/abone olma etki alanı için bir TopicConnectionFactory nesnesi olabilir.

İpucu: JMS 2.0ile hem bağlantılar hem de bağlamlar oluşturmak için bir bağlantı üreticisi kullanılabilir. Sonuç olarak, hem bağlantıların hem de bağlamların karışımını içeren bir bağlantı üreticisiyle ilişkilendirilmiş bir bağlantı havuzu olabilir. Bir bağlantı üreticisinin yalnızca bağlantı yaratmak ya da bağlam yaratmak için kullanılması önerilir. Bu, bağlantı üreticisine ilişkin bağlantı havuzunun yalnızca tek tipli nesneler içermesini sağlayarak havuzu daha verimli hale getirir.

ConnectionFactory nesnesinin özellikleri

Çizelge 66 sayfa 454 içinde bir ConnectionFactory nesnesinin özellikleri listelenir. Uygulama sunucusu, bir JMS ConnectionFactory nesnesi yaratmak için bu özellikleri kullanır.

Çizelge 66. ConnectionFactory nesnesinin özellikleri

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
applicationName	Dizgi	• Çağıran sınıf adı (varsa), 28 karakterden uzun olmayacak şekilde ayarlanır. Yoksa, `WebSphere MQ Client for Java` dizgisi kullanılır.	Bir uygulamanın kuyruk yöneticisine kayıtlı olduğu ad. Bu uygulama adı, **`DISPLAY CONN MQSC/PCF`** komutuyla (alanın **APPLTAG**olarak adlandırıldığı) ya da IBM MQ Gezgin **Uygulama Bağlantıları** görüntüsünde (burada alan **App name**olarak adlandırılır) gösterilir.
brokerCCSubKuyruğu	Dizgi	• **SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE** • Kuyruk adı	Bağlantı tüketicisinin sürekli olmayan abonelik iletileri aldığı kuyruğun adı.
brokerControlKuyruğu	Dizgi	• **SYSTEM.BROKER.CONTROL.QUEUE** • Kuyruk adı	Aracı denetim kuyruğunun adı.
brokerPubKuyruğu	Dizgi	• **SYSTEM.BROKER.DEFAULT.STREAM** • Kuyruk adı	Yayınlanan iletilerin gönderildiği kuyruğun adı (akış kuyruğu).
brokerQueueYöneticisi	Dizgi	• **"" (boş dizgi)** • Kuyruk yöneticisi adı	Aracının çalıştığı kuyruk yöneticisinin adı.
brokerSubKuyruğu	Dizgi	• **SYSTEM.JMS.ND.SUBSCRIBER.QUEUE** • Kuyruk adı	Sürekli olmayan bir ileti tüketicisinin iletileri aldığı kuyruğun adı. Ek bilgi için BROKERSUBQ özelliğine bakın.
brokerVersion	Dizgi	• **belirlenmedi** -Aracı V6 'dan V7' ye geçirildikten sonra, bu özelliği RFH2 üstbilgilerinin artık kullanılmayacak şekilde ayarlayın. Geçişten sonra bu özellik artık ilgili değildir. • **V1** - IBM MQ Yayınla/Abone Ol aracısını kullanmak için. TRANSPORT, BIND ya da CLIENT olarak ayarlandıysa, bu değer varsayılan değerdir. • **V2** -Yerel kipte IBM Integration Bus aracısını kullanmak için. TRANSPORT değeri DIRECT ya da DIRECTHTTP olarak ayarlandıysa, bu değer varsayılan değerdir.	Kullanılmakta olan aracının sürümü.

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
ccdtURL	Dizgi	• **boş değerli** • Birörnek kaynak yeri belirleyici (URL)	İstemci kanal tanımlama çizelgesini (CCDT) içeren dosyanın adını ve yerini tanımlayan ve dosyaya nasıl erişilebileceğini belirten URL .
CCSID	Dizgi	• **819** • Java sanal makinesi (JVM) tarafından desteklenen bir kodlanmış karakter takımı tanıtıcısı	Bağlantıya ilişkin kodlanmış karakter takımı tanıtıcısı.
kanal	Dizgi	• **SYSTEM.DEF.SVRCONN** • MQI kanalının adı	Kullanılacak MQI kanalının adı.
cleanupInterval	int	• **3600 000** • Artı bir tamsayı	Yayınlama/abone olma temizleme yardımcı programının arka plan çalıştırmaları arasındaki milisaniye cinsinden aralık.
cleanupLevel	Dizgi	• **GüVENLI** • YOK • güçlü • ZORLA • NONDUR	Aracıya dayalı bir abonelik deposunun temizleme düzeyi.
clientID	Dizgi	• **boş değerli** • Bir istemci tanıtıcısı	Bağlantıya ilişkin istemci tanıtıcısı.
cloneSupport	Dizgi	• **DEVRE Dışı** -Bir kerede tek bir sürekli konu abonesi çalıştırılabilir. • ENABLED-Aynı sürekli konu abonesinin iki ya da daha fazla eşgörünümü aynı anda çalışabilir, ancak her yönetim ortamının ayrı bir Java sanal makinesinde (JVM) çalışması gerekir.	Aynı sürekli konu abonesinin iki ya da daha fazla eşgörünümünün aynı anda çalışıp çalışamayacağını belirler.

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
connectionNameListesi	Dizgi	• **localhost (1414)** • Her öğenin biçimi aldığı, virgüllerle ayrılmış öğelerden oluşan bir dizgi: `HOSTNAME(PORT)` Burada *HOSTNAME* , bir DNS adı ya da bir IP adresidir.	Giden iletişim için kullanılan TCP/IP bağlantı adlarının listesi. **connectionNameList** , **hostname** ve **port** özelliklerinin yerini alır. Bu özellik, çok eşgörünümlü kuyruk yöneticilerine yeniden bağlanmak için kullanılır. **connectionNameList** , formda **localAddress**ile benzer, ancak bununla karıştırılmamalıdır. **localAddress** , yerel iletişimin özelliklerini belirtirken, **connectionNameList** uzak kuyruk yöneticisine nasıl erişileceğini belirtir.
failIfSusturma	Boole	• **doğru** • yanlış	Kuyruk yöneticisi susturma durumundaysa, belirli yöntemlere yönelik çağrıların başarısız olup olmayacağını belirler.
headerCompression	Dizgi	• **YOK** • SYSTEM-RLE ileti üstbilgisi sıkıştırması gerçekleştirilir.	Bir bağlantıdaki üstbilgi verilerinin sıkıştırılması için kullanılabilecek tekniklerin listesi.
hostName	Dizgi	• **localhost** • Anasistem adı • Bir IP adresi	Kuyruk yöneticisinin bulunduğu sistemin anasistem adı ya da IP adresi. **hostname** ve **port** özellikleri, belirtildiğinde **connectionNameList** özelliği tarafından değiştirilir.
localAddress	Dizgi	• **boş değerli** • Şu biçimde bir dizgi: `[ host_name ][(low_port [, high_port ])]` Burada *anasistem_adı* bir anasistem adı ya da IP adresi, *alt_kapı* ve *yüksek_kapı* TCP kapı numaralarıdır ve köşeli ayraçlar isteğe bağlı bir bileşeni gösterir	Bir kuyruk yöneticisine yönelik bağlantı için, bu özellik aşağıdakilerden birini ya da her ikisini de belirtir: • Kullanılacak yerel ağ arabirimi • Kullanılacak yerel kapı ya da yerel kapı aralığı **localAddress** , formda **connectionNameList**ile benzer, ancak bununla karıştırılmamalıdır. **localAddress** , yerel iletişimin özelliklerini belirtirken, **connectionNameList** uzak kuyruk yöneticisine nasıl erişileceğini belirtir.

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
messageCompression	Dizgi	• **YOK** • Boş karakterlerle ayrılmış olarak aşağıdaki değerlerden birinin ya da daha fazlasının listesi: RLE ZLIBFAST ZLIBHIGH	Bir bağlantıdaki ileti verilerini sıkıştırmada kullanılabilecek tekniklerin listesi.
messageSelection	Dizgi	• **İstemci** • Aracı	İleti seçiminin IBM MQ classes for JMS tarafından mı, yoksa aracı tarafından mı yapılacağını belirler. **`brokerVersion`** 1değerine sahip olduğunda aracı tarafından ileti seçimi desteklenmez.
parola	Dizgi	• **boş değerli** • Parola	Kuyruk yöneticisine bağlantı yaratılırken kullanılacak varsayılan parola.
pollingInterval	int	• **5000** • Pozitif bir tamsayı	Bir oturum içindeki her ileti dinleyicinin kuyruğunda uygun bir ileti yoksa, bu değer, her ileti dinleyicisinin kuyruğundan bir ileti almaya yeniden çalışmasından önce geçmesi gereken milisaniye cinsinden aralık üst sınırıdır. Sık sık oturumdaki ileti dinleyicilerden herhangi biri için uygun bir ileti yoksa, bu özelliğin değerini artırmayı düşünebilirsiniz. Bu özellik yalnızca **TRANSPORT** , `BIND` ya da `CLIENT`değerine sahipse geçerlidir.
kapı	int	• **1414** • TCP kapı numarası	Kuyruk yöneticisinin dinlediği kapı. **`hostname`** ve **`port`** özellikleri, belirtildiğinde **`connectionNameList`** özelliği tarafından değiştirilir.
providerVersion	dizgi	• **belirtilmedi** • Aşağıdaki biçimlerden birindeki bir dizgi – V.R.M.F – V.R.M – V.R – V Burada V, R, M ve F sıfırdan büyük ya da sıfıra eşit tamsayı değerleridir.	Uygulamanın bağlanmayı planladığı kuyruk yöneticisinin sürümü, yayın düzeyi, değişiklik düzeyi ve düzeltme paketi.

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
pubAckAralığı	int	• **25** • Artı bir tamsayı	IBM MQ classes for JMS aracıdan bir alındı bildirimi istemeden önce bir yayınlayıcı tarafından yayınlanan iletilerin sayısı.
queueManager	Dizgi	• **"" (boş dizgi)** • Kuyruk yöneticisi adı	Bağlanılacak kuyruk yöneticisinin adı.
receiveExit [3]	Dizgi	• **boş değerli** • Virgüllerle ayrılmış bir ya da daha çok öğeden oluşan bir dizgi; burada her öğe, IBM MQ classes for Java arabirimini gerçekleştiren bir sınıfın tam olarak nitelenmiş adıdır (MQReceiveExit).	Bir kanal alma çıkış programını ya da arka arkaya çalıştırılacak bir alma çıkış programları sırasını tanımlar.
receiveExitBaşlatma	Dizgi	• **boş değerli** • Virgüllerle ayrılmış bir ya da daha fazla kullanıcı verisi öğesini içeren bir dizgi	Kanala geçirilen kullanıcı verileri, çağrıldıklarında çıkış programlarını alır.
rescanInterval	int	• **5000** • Pozitif bir tamsayı	Noktadan noktaya iletişim etki alanındaki bir ileti tüketicisi, almak istediği iletileri seçmek için bir ileti seçici kullandığında, IBM MQ classes for JMS , IBM MQ kuyruğunda kuyruğun **MsgDeliverySequence** özniteliği tarafından belirlenen sırada uygun iletileri arar. IBM MQ classes for JMS uygun bir ileti bulduğunda ve bunu tüketiciye teslim ettiğinde IBM MQ classes for JMS , kuyruktaki geçerli konumundan sonraki uygun iletiyi aramaya devam eder. IBM MQ classes for JMS , kuyruğun sonuna ulaşıncaya kadar ya da bu özelliğin değerinin belirlediği milisaniye cinsinden zaman aralığının süresi doluncaya kadar kuyruğu bu şekilde aramaya devam eder. Her durumda IBM MQ classes for JMS , aramaya devam etmek için kuyruğun başlangıcına geri döner ve yeni bir zaman aralığı başlar.
securityExit [3]	Dizgi	• **boş değerli** • IBM MQ classes for Java arabirimini gerçekleştiren bir sınıfın tam olarak nitelenmiş adı, MQSecurityExit	Bir kanal güvenliği çıkış programını tanımlar.

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
securityExitBaşlatma	Dizgi	• **boş değerli** • Kullanıcı verileri dizgisi	Çağrıldığında kanal güvenliği çıkış programına geçirilen kullanıcı verileri.
sendCheckSayısı	int	• **ERROR! SEGMENT DATA CORRUPTED, SEGDATA=0** • Pozitif bir tamsayı	Hareket etmeyen tek bir JMS oturumunda zamanuyumsuz koyma hatalarının denetlenmesi arasında izin verilecek gönderme çağrılarının sayısı.
sendExit [3]	Dizgi	• **boş değerli** • Virgüllerle ayrılmış bir ya da daha çok öğeden oluşan bir dizgi; burada her öğe, IBM MQ classes for Java arabirimini gerçekleştiren bir sınıfın tam olarak nitelenmiş adıdır (MQSendExit).	Kanal gönderme çıkış programını ya da art arda çalıştırılacak çıkış gönderme programlarını tanımlar.
sendExitInit	Dizgi	• **boş değerli** • Virgüllerle ayrılmış bir ya da daha fazla kullanıcı verisi öğesini içeren bir dizgi	Kanala geçirilen kullanıcı verileri, çağrıldığında çıkış programlarını gönderir.
shareConvİzin Verilir	Boole	• NO-Bir istemci bağlantısı yuvasını paylaşamıyor. • **YES** (EVET)-Bir istemci bağlantısı yuvasını paylaşabilir.	Bir istemci bağlantısının, kanal tanımlamaları eşleşiyorsa, aynı işlemden aynı kuyruk yöneticisine diğer üst düzey JMS bağlantılarıyla yuvasını paylaşıp paylaşamayacağını belirler.
sparseSubscriptions	Boole	• **false** -Abonelikler sık sık eşleşen iletiler alır. • true-Abonelikler nadiren eşleşen iletiler alır. Bu değer, abonelik kuyruğunun göz atmak için açılabilmesini gerektirir.	TopicSubscriber nesnesinin ileti alma ilkesini denetler.
sslCertMağazaları	Dizgi	• **boş değerli** • Boşluklarla ayrılmış bir ya da daha çok LDAP URL dizesi. Her LDAP URL şu biçimdedir: `ldap://host_name [: port ]` Burada *anasistem_adı* bir anasistem adı ya da IP adresidir, *kapı* bir TCP kapı numarasıdır ve köşeli parantezler isteğe bağlı bir bileşeni gösterir.	TLS bağlantısında kullanılmak üzere sertifika iptal listelerini (CRL ' ler) tutan LDAP sunucuları.
sslCipherÜrün Grubu	Dizgi	• **boş değerli** • CipherSuite adı	TLS bağlantısı için kullanılacak CipherSuite .

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
sslFipsGerekli [2]	Boole	• **yanlış** • doğru	Bir TLS bağlantısının, IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) tarafından desteklenen bir CipherSuite kullanması gerekip gerekmediğini belirler.
sslPeerAdı	Dizgi	• **boş değerli** • Ayırt edici adlar için şablon	TLS bağlantısı için, kuyruk yöneticisi tarafından sağlanan sayısal sertifikada ayırt edici adı denetlemek için kullanılan bir şablon.
sslResetSayısı	int	• **ERROR! SEGMENT DATA CORRUPTED, SEGDATA=0** • 0-999 999 999 aralığında bir tamsayı	TLS tarafından kullanılan gizli anahtarlar yeniden anlaşılmadan önce bir TLS bağlantısı tarafından gönderilen ve alınan toplam bayt sayısı.
sslSocketÜreticisi	Dizgi	javax.net.ssl.SSLSocketFactory arabiriminin somutlamasını sağlayan bir sınıfın tam olarak nitelenmiş sınıf adını gösteren dizgi; isteğe bağlı olarak, oluşturucu yöntemine geçirilecek bir bağımsız değişken de içinde olmak üzere, ayraç içine alınmış olarak.	Yönetilen hedef nesne kapsamında kurulan tüm bağlantılar, SSLSocketFactory arabiriminin bu somutlamasından elde edilen yuvaları kullanır.
statusRefreshAralığı	int	• **60000** • Pozitif bir tamsayı	Bir abonenin kuyruk yöneticisine olan bağlantısını kaybettiği zaman algılanan, uzun süren hareketin yenilenmesi arasındaki milisaniye cinsinden aralık. Bu özellik yalnızca **SUBSTORE** QUEUEdeğerine sahipse geçerlidir.
subscriptionStore	Dizgi	• **Aracı** • GEÇİŞ YAPIN • kuyruk	IBM MQ classes for JMS , etkin aboneliklerle ilgili kalıcı verilerin nerede saklanacağını belirler.
targetClientEşleştirme	Boole	• **doğru** • yanlış	Gelen bir iletinin JMSReplyTo üstbilgi alanıyla tanıtılan kuyruğa gönderilen bir yanıt iletisinin MQRFH2 üstbilgisi, yalnızca gelen iletinin bir MQRFH2 üstbilgisi varsa olup olmadığını belirler. Bu özelliği bir etkinleştirme belirtimi için de yapılandırabilirsiniz. Daha fazla bilgi için bkz “Etkinleştirme belirtimi için targetClientMatching özelliğinin yapılandırılması” sayfa 468.

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
temporaryModel	Dizgi	• **SYSTEM.DEFAULT.MODEL.QUEUE** • SYSTEM.JMS.TEMPQ.MODEL • Herhangi bir dizgi	JMS geçici kuyruklarının yaratıldığı model kuyruğunun adı. `SYSTEM.DEFAULT.MODEL.QUEUE` : • Uygulamanız, kalıcı olmayan iletileri kabul edecek geçici bir kuyruk kullanıyor. • Kuyruk yöneticisinde, ConnectionFactory ' in bir kerede gösterdiği geçici bir kuyruk yalnızca bir uygulama yaratır. `SYSTEM.DEFAULT.MODEL.QUEUE` , aynı anda yalnızca bir uygulama tarafından açılabilir. `SYSTEM.JMS.TEMPQ.MODEL` : • Uygulamanız kalıcı iletileri kabul edecek geçici bir kuyruk kullandığında. • Birden çok uygulama ConnectionFactory ' in gösterdiği kuyruk yöneticisine bağlanabiliyorsa ve bu uygulamaların aynı anda geçici kuyruklar oluşturmaları gerekiyorsa. **DEFPSIST** özniteliği YESdeğerine ayarlanmış olarak yeni bir model kuyruğu tanımlayın ve **DEFSOPT** özniteliği şu durumda SHARED değerine ayarlandı: • Uygulamanız kalıcı olmayan iletileri kabul edecek geçici bir kuyruk kullandığında ve birden çok uygulama ConnectionFactory ' in işaret ettiği kuyruk yöneticisine bağlandığında, bu uygulamaların aynı anda geçici kuyruklar oluşturmaları gerekir. Yeni model kuyruğu yaratıldığında, **`temporaryModel`** özelliğini yeni model kuyruğunun adına ayarlayın.

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
tempQPrefix	Dizgi	• **"" (boş dizgi)** • Bir IBM MQ dinamik kuyruğunun adını oluşturmak için kullanılabilecek bir önek. Öneki oluşturmaya ilişkin kurallar, IBM MQ nesne tanımlayıcısı, MQOD yapısı içindeki **DynamicQName** alanının içeriğini oluşturmaya ilişkin kurallarla aynıdır, ancak son boş olmayan karakter bir yıldız işareti (*) olmalıdır. Özelliğin değeri boş dizgiyse, IBM MQ classes for JMS AMQ.* değerini kullanır. Dinamik bir kuyruk yaratırken.	Bir IBM MQ dinamik kuyruğunun adını oluşturmak için kullanılan önek.
tempTopicÖneki	Dizgi	IBM MQ konu dizesi için yalnızca geçerli karakterlerden oluşan boş olmayan herhangi bir dizgi	Geçici konular oluştururken JMS , "TEMP/*TEMPTOPICPREFIX*/*unique_id* " biçiminde bir konu dizgisi oluşturur ya da bu özellik varsayılan değerle bırakılırsa, yalnızca "TEMP/*unique_id*". Boş olmayan bir **TEMPTOPICPREFIX** belirtilmesi, abonelerin bu bağlantı altında yaratılan geçici konulara ilişkin yönetilen kuyrukları yaratmak için belirli model kuyruklarının tanımlanmasına olanak sağlar.

Çizelge 66. ConnectionFactory nesnesinin özellikleri (devamı var)			
Özelliğin adı	**Tip**	**Geçerli değerler (koyu varsayılan değer)**	**Açıklama**
transportType	Dizgi	• **İstemci** • Bağ Tanımları • BINDINGS_THEN_CLIENT	Bir kuyruk yöneticisine yönelik bağlantının istemci kipini mi, yoksa bağ tanımlama kipini mi kullandığını belirler. BINDINGS_THEN_CLIENT değeri belirtilirse, kaynak bağdaştırıcısı önce bağ tanımlama kipinde bağlantı kurmaya çalışır. Bu bağlantı girişimi başarısız olursa, kaynak bağdaştırıcısı istemci kipi bağlantısı kurmayı dener. z/OS Bir WebSphere Application Server for z/OS sisteminde çalışan bir etkinleştirme belirtimi BINDINGS_THEN_CLIENT iletim kipini kullanacak şekilde yapılandırıldıysa ve önceden kurulmuş bir bağlantı kesildiyse, etkinleştirme belirtiminin yeniden bağlanma girişimleri önce BINDINGS iletim kipini kullanmayı dener. BINDINGS iletim kipi bağlantı girişimi başarısız olursa, etkinleştirme belirtimi daha sonra CLIENT iletim kipi bağlantısını dener.
kullanıcı adı	Dizgi	• **boş değerli** • Kullanıcı adı	Bir kuyruk yöneticisine bağlantı yaratırken kullanılacak varsayılan kullanıcı adı.
wildcardFormat	int	• CHAR-Aracı sürüm 1 'de kullanıldığı şekilde, yalnızca genel arama karakterlerini tanır • TOPIC-Aracı sürüm 2 'de kullanıldığı şekilde, yalnızca konu düzeyi genel arama karakterlerini tanır	Genel arama karakteri sözdiziminin hangi sürümünün kullanılacağı.

Notlar:

1. sslFipsRequired özelliğini kullanma hakkında önemli bilgi için bkz. “IBM MQ kaynak bağdaştırıcısının sınırlamaları” sayfa 423.
2. Kaynak bağdaştırıcısının bir çıkış bulabilmesi için nasıl yapılandırılacağı hakkında bilgi için bkz. “IBM MQ classes for JMS ' nin kanal çıkışlarını kullanacak şekilde yapılandırılması” sayfa 271.

Aşağıdaki örnek, bir ConnectionFactory nesnesinin tipik bir özellik kümesini göstermektedir:

```
channel:         SYSTEM.DEF.SVRCONN
hostName:        192.168.0.42
port:            1414
queueManager:    ExampleQM
transportType:   CLIENT
```

Denetlenen hedef nesnenin özellikleri

Uygulama sunucusu, bir JMS Kuyruk nesnesi ya da JMS Konu nesnesi yaratmak için denetlenen hedef nesnenin özelliklerini kullanır.

Çizelge 67 sayfa 464 içinde bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler listelenir.

Çizelge 67. Bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
CCSID	Dizgi	• **1208** • Java sanal makinesi (JVM) tarafından desteklenen bir kodlanmış karakter takımı tanıtıcısı	Hedefe ilişkin kodlanmış karakter takımı tanıtıcısı.
Kodlama	Dizgi	• **Yerel** • Üç karakterden oluşan bir dizgi: – İlk karakter ikili tamsayıların gösterimini belirtir: - *N* , normal kodlamayı gösterir. - *R* ters kodlamayı belirtir. – İkinci karakter paketlenmiş ondalık tamsayıların gösterimini belirtir: - *N* , normal kodlamayı gösterir. - *R* ters kodlamayı belirtir. – Üçüncü karakter, kayan noktalı sayıların gösterimini belirtir: - *N* standart IEEE kodlamasını belirtir. - *R* , ters IEEE kodlamasını belirtir. - *3* , zSeries kodlamasını belirtir. NATIVE, NNN dizesine eşdeğerdir.	Hedef için ikili tamsayıların, paketlenmiş ondalık tamsayıların ve kayan noktalı sayıların gösterimi.
Son kullanma tarihi	Dizgi	• **APP** -İletinin süre bitimi, ileti üreticisi tarafından belirlenir. • UNLIM-Bir iletinin süresi hiçbir zaman dolmaz. • 0-Bir iletinin süresi hiçbir zaman dolmaz. • Bir iletinin milisaniye cinsinden süre bitimini gösteren pozitif bir tamsayı.	Hedefe gönderilen bir iletinin süre bitimi.
failIfSusturma	Dizgi	• **doğru** • yanlış	Kuyruk yöneticisi susturma durumundaysa, hedefe erişme girişiminin başarısız olup olmadığını belirler.

Çizelge 67. Bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
messageBodyBiçemi	Dizgi	• **Belirtilmedi** • JMS • MQ	**`messageBodyStyle`** özelliğini JMS kuyruklarında ve konularında ayarlayabilirsiniz: UNSPECIFIED (varsayılan) • Gönderirken, WMQ_TARGET_CLIENT değerine bağlı olarak IBM MQ classes for JMS bir MQRFH2 üstbilgisi oluşturun ve ekleyin. • Alma sırasında, IBM MQ classes for JMS JMS ileti özelliklerini MQRFH2' deki değerlere göre ayarlayın (varsa). MQRFH2 , JMS ileti gövdesinin bir parçası olarak sunulmaz. JMS • Gönderirken IBM MQ classes for JMS otomatik olarak bir MQRFH2 üstbilgisi oluşturur ve IBM MQ iletisinde üstbilgiyi içerir. • Alma sırasında, IBM MQ classes for JMS JMS ileti özelliklerini MQRFH2' deki değerlere göre ayarlayın (varsa). MQRFH2 , JMS ileti gövdesinin bir parçası olarak sunulmaz. MQ • IBM MQ classes for JMS gönderirken bir MQRFH2oluşturmayın. • IBM MQ classes for JMS , alırken JMS ileti gövdesinin bir parçası olarak MQRFH2 ' yi sunar.

Çizelge 67. Bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler (devamı var)

Özelliğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
Kalıcılık	Dizgi	• **APP** -Bir iletinin sürekliliği, ileti üreticisi tarafından belirlenir. • QDEF-Bir iletinin sürekliliği, IBM MQ kuyruğunun **DefPersistence** özniteliği tarafından belirlenir. • PERS-Bir ileti kalıcı. • Hayır-Bir ileti kalıcı değil. • HIGH-Bir iletinin kalıcı olarak saklanması, IBM MQ kuyruğunun **NonPersistentMessageClass** özniteliği, "JMS kalıcı iletiler" sayfa 244içindeki açıklamaya göre belirlenir.	Hedefe gönderilen bir iletinin kalıcılığı.
öncelik	Dizgi	• **APP** -Bir iletinin önceliği, ileti üreticisi tarafından belirlenir. • QDEF-Bir iletinin önceliği, IBM MQ kuyruğunun **DefPriority** özniteliğine göre belirlenir. • 0, en düşük öncelik, 9, en yüksek öncelik aralığında bir tamsayı.	Hedefe gönderilen bir iletinin önceliği.
putAsync' e izin verilir	Dizgi	• QUEUE-Kuyruk tanımına başvurarak zamanuyumsuz girişlere izin verilip verilmediğini saptayın. • KONU-Konu tanımlamasına başvurarak zamanuyumsuz girişlere izin verilip verilmediğini saptayın. • DESTINATION-Kuyruk ya da konu tanımlamasına başvurarak zamanuyumsuz girişlere izin verilip verilmediğini belirler. • DISABLED-Zamanuyumsuz girişlere izin verilmez. • ENABLED-Zamanuyumsuz girişlere izin verilir.	İleti üreticilerinin bu hedefe ileti göndermek için zamanuyumsuz girişler kullanmalarına izin verilip verilmediğini belirler.
readAheadİzin verilir	int	• **DESTINATION** -Kuyruk ya da konu tanımlamasına başvurarak önden okumaya izin verilip verilmediğini belirleyin. • DISABLED-Önden okumaya izin verilmez. • ENABLED-Önden okumaya izin verilir. • QUEUE-Kuyruk tanımına başvurarak önden okumaya izin verilip verilmediğini belirleyin. • KONU-Konu tanımlamasına başvurarak önden okumaya izin verilip verilmediğini belirleyin.	İleti tüketicilerinin ve kuyruk tarayıcılarının, hedeften kalıcı olmayan iletileri almadan önce bir iç arabelleğe almak için önden okuma özelliğini kullanıp kullanmalarına izin verilip verilmediğini belirler.

Çizelge 67. Bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler (devamı var)			
Özelliğin adı	**Tip**	**Geçerli değerler (koyu varsayılan değer)**	**Açıklama**
receiveCCSID	int	• **0** -JVM kullan `Charset.defaultCharset` • 1208- UTF-8 • Desteklenen bir kodlanmış karakter takımı tanıtıcısı	Kuyruk yöneticisi ileti dönüştürmesi için hedef CCSID ' yi belirleyen hedef özellik. **`receiveConversion`** , `QMGR`olarak ayarlanmadıkça değer yoksayılır.
receiveConversion	Dizgi	• **CLIENT_MSG** • QMGR	Veri dönüştürme işleminin kuyruk yöneticisi tarafından gerçekleştirilip gerçekleştirilmeyeceğini belirleyen hedef özellik.
targetClient	Dizgi	• **JMS** -İletinin hedefi bir JMS uygulamasıdır. • MQ -Bir iletinin hedefiJMS IBM MQ dışı bir uygulamadır.	Hedefe gönderilen bir iletinin hedefinin bir JMS uygulaması olup olmadığını belirler. JMS uygulaması olan bir hedef içeren bir ileti, MQRFH2 üstbilgisi içeriyor.

Çizelge 68 sayfa 467 içinde bir Kuyruk nesnesine özgü özellikler listelenir.

Çizelge 68. Kuyruk nesnesine özgü özellikler			
Özelliğin adı	**Tip**	**Geçerli değerler (koyu varsayılan değer)**	**Açıklama**
baseQueueManagerName	Dizgi	• **"" (boş dizgi)** • Kuyruk yöneticisi adı	Temel IBM MQ kuyruğunun sahibi olan kuyruk yöneticisinin adı.
baseQueueAdı	Dizgi	• **"" (boş dizgi)** • Kuyruk adı	Temel IBM MQ kuyruğunun adı.

Çizelge 69 sayfa 467 içinde, bir Konu nesnesine özgü özellikler listelenir.

Çizelge 69. Konu nesnesine özgü özellikler			
Özelliğin adı	**Tip**	**Geçerli değerler (koyu varsayılan değer)**	**Açıklama**
baseTopicAdı	Dizgi	• **"" (boş dizgi)** • Konu adı	Temeldeki konunun adı.
brokerCCDurSubQueue >	Dizgi	• **SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE** • Kuyruk adı	Bağlantı tüketicisinin sürekli abonelik iletileri aldığı kuyruğun adı.
brokerDurSubQueue	Dizgi	• **SYSTEM.JMS.D.SUBSCRIBER.QUEUE** • Kuyruk adı	Sürekli bir konu abonesinin iletileri aldığı kuyruğun adı. Ek bilgi için IBM MQ Explorer belgelerinde **BROKEDURRSUBQ** özelliğine bakın.

Çizelge 69. Konu nesnesine özgü özellikler (devamı var)			
Özelliğin adı	**Tip**	**Geçerli değerler (koyu varsayılan değer)**	**Açıklama**
brokerPubKuyruğu	Dizgi	• **Ayarlanmadı** • Kuyruk adı	Yayınlanan iletilerin gönderildiği kuyruğun adı (akış kuyruğu). Bu özelliğin değeri, ConnectionFactory nesnesinin **brokerPubQueue** özelliğinin değerini geçersiz kılar. Ancak, bu özelliğin değerini ayarlamazsanız, bunun yerine ConnectionFactory nesnesinin **brokerPubQueue** özelliğinin değeri kullanılır.
brokerPubQueueManager	Dizgi	• **"" (boş dizgi)** • Kuyruk yöneticisi adı	Konu üzerinde yayınlanan iletilerin gönderildiği kuyruğun iyesi olan kuyruk yöneticisinin adı.
brokerVersion	Dizgi	• **Ayarlanmadı** • 1 • 2	Kullanılmakta olan aracının sürümü. Bu özelliğin değeri, ConnectionFactory nesnesinin **brokerVersion** özelliğinin değerini geçersiz kılar. Ancak, bu özelliğin değerini ayarlamazsanız, bunun yerine ConnectionFactory nesnesinin **brokerVersion** özelliğinin değeri kullanılır.

Aşağıdaki örnek, bir Kuyruk nesnesine ilişkin özellikler kümesini göstermektedir:

```
expiry:               UNLIM
persistence:          QDEF
baseQueueManagerName: ExampleQM
baseQueueName:        SYSTEM.JMS.TEMPQ.MODEL
```

Aşağıdaki örnekte, bir Konu nesnesine ilişkin özellikler kümesi gösterilmektedir:

```
expiry:               UNLIM
persistence:          NON
baseTopicName:        myTestTopic
```

İlgili görevler

MQI istemcisinde çalıştırma zamanında yalnızca FIPS onaylı CipherSpecs kullanılmasının belirtilmesi

WebSphere Application Server ' de JMS kaynaklarının yapılandırılması

İlgili başvurular

AIX, Linux, and Windows için Federal Bilgi İşleme Standartları (FIPS)

Etkinleştirme belirtimi için targetClientMatching özelliğinin yapılandırılması

İstek iletileri MQRFH2 üstbilgisi içermediğinde MQRFH2 üstbilgisinin yanıt iletilerine eklenmesi için **targetClientMatching** özelliğini bir etkinleştirme belirtimi için yapılandırabilirsiniz. Bu, bir uygulamanın bir yanıt iletisinde tanımladığı ileti özelliklerinin, ileti gönderildiğinde dahil edileceği anlamına gelir.

Bu görev hakkında

Bir MDB uygulaması, bir IBM MQ JCA kaynak bağdaştırıcısı etkinleştirme belirtimi aracılığıyla MQRFH2 üstbilgisi içermeyen iletiler tüketirse ve daha sonra, istek iletisinin `JMSReplyTo` alanından yaratılan JMS Hedefine yanıt iletileri gönderirse, istek iletileri MQRFH2 üstbilgisini içermese bile, istek iletileri bir MQRFH2 üstbilgisini içermelidir; tersi durumda, uygulamanın bir yanıt iletisinde tanımladığı tüm ileti özellikleri kaybolur.

`targetClientMatching` özelliği, gelen bir iletinin JMSReplyTo üstbilgi alanıyla tanıtılan kuyruğa gönderilen bir yanıt iletisinin MQRFH2 üstbilgisinin yalnızca gelen iletinin bir MQRFH2 üstbilgisine sahip olup olmadığını tanımlar. Bu özelliği, WebSphere Application Server traditional ve WebSphere Libertyiçinde bir etkinleştirme belirtimi için yapılandırabilirsiniz.

`targetClientMatching` özelliğinin değerini `false`olarak ayarlarsanız, MQRFH2içermeyen bir gelen istek iletisinin JMSReplyTo üstbilgisinden oluşturulan bir JMS Hedefine gönderilen bir yanıt iletisine MQRFH2 üstbilgisi eklenebilir. Bunun nedeni, JMS Hedefindeki **`targetClient`** özelliğinin `0`değerine ayarlanmış olmasıdır; bu, iletilerin bir MQRFH2 üstbilgisi içerdiği anlamına gelir. Giden iletideki MQRFH2 üstbilgisinin varlığı, IBM MQ kuyruğuna gönderildiğinde iletide kullanıcı tanımlı ileti özelliklerinin saklanmasına izin verir.

`targetClientMatching` özelliği `true` olarak ayarlanırsa ve bir istek iletisi MQRFH2 üstbilgisi içermiyorsa, yanıt iletisine MQRFH2 üstbilgisi eklenmez.

Yordam

- WebSphere Application Server traditionaliçinde, **`targetClientMatching`** özelliğini IBM MQ etkinleştirme belirtiminde özel bir özellik olarak tanımlamak için yönetim konsolunu kullanın:
  a) Gezinme bölmesinde **Resources-> JMS-> Activation belirtimleri**(Kaynaklar-> JMS-> Etkinleştirme belirtimleri) seçeneklerini tıklatın.
  b) Görüntülemek ya da değiştirmek istediğiniz etkinleştirme belirtiminin adını seçin.
  c) **Özel özellikler-> Yeni** seçeneklerini tıklatın ve yeni özel özelliğin ayrıntılarını girin.
  Özelliğin adını `targetClientMatching`, tipini `java.lang.Boolean` ve değerini `false`olarak ayarlayın.
- WebSphere Libertyiçinde, `server.xml`içindeki bir etkinleştirme belirtiminin tanımlamasında **`targetClientMatching`** özelliğini belirtin.
  Örneğin:

```
<jmsActivationSpec id="SimpleMDBApplication/SimpleEchoMDB/SimpleEchoMDB">
<properties.wmqJms destinationRef="MDBRequestQ"
queueManager="MY_QMGR" transportType="BINDINGS" targetClientMatching="false"/>
<authData password="********" user="tom"/>
</jmsActivationSpec>
```

İlgili kavramlar

“JMS uygulamasında hedef oluşturma” sayfa 213
Bir JMS uygulaması, Java Naming and Directory Interface (JNDI) ad alanından yönetilen nesneler olarak hedefleri almak yerine, yürütme sırasında dinamik olarak hedef yaratmak için bir oturumu kullanabilir. Bir uygulama, IBM MQ kuyruğunu ya da konusunu tanımlamak ve isteğe bağlı olarak bir Kuyruk ya da Konu nesnesine ilişkin bir ya da daha çok özellik belirtmek için tek tip kaynak tanıtıcısını (URI) kullanabilir.

“Kaynak bağdaştırıcısının giden iletişim için yapılandırılması” sayfa 452
Giden iletişim yapılandırmak için bir ConnectionFactory nesnesinin özelliklerini ve denetlenen bir hedef nesneyi tanımlayın.

IBM MQ ileti odaklı bean duraklatmasında WebSphere Liberty

Bir etkinleştirme belirtimine ilişkin **`maxSequentialDeliveryFailures`** özelliği, kaynak bağdaştırıcısının MDB ' yi duraklatmadan önce dayandığı ileti odaklı bean (MDB) eşgörünümüne yönelik sıralı ileti teslimi başarısızlığı sayısı üst sınırını tanımlar.

Başlamadan önce

WebSphere Libertyiçinde MDB ' nin duraklatılmasına neden olabilecek olaylar kümesini bilmeniz gerekir. Kaynak bağdaştırıcısı, aşağıdakilerden birini bir ileti teslim hatası olarak kabul eder:

- MDB ' nin **onMessage** yönteminden denetimsiz bir kural dışı durum yayınlanıyor.
- İletiyi MDB ' ye teslim etmeden önce kaynak bağdaştırıcısının işlenmesinde oluşan bir `JMSException` .
- Kaynak bağdaştırıcısının işlenmesinde oluşan ve iletiyi MDB ' ye teslim eden bir `JMSException` .
- Geriye işlenmekte olan iletiyi kullanmak için kullanılan XA hareketi ya da yerel hareket.
- İletiyi MDB ' ye teslim etmek için uygulama sunucusunda kullanılabilir bir iş parçacığı yok.

Bu görev hakkında

`maxSequentialDeliveryFailures` özelliğinin varsayılan değeri, MDB 'nin hiçbir zaman duraklatılmadığı anlamına gelen *-1*' dir. Diğer herhangi bir negatif değer *-1*ile aynı şekilde işlenir. Bir değer:

- *0* , MDB ' nin ilk hatada durakladığını gösterir.
- *1* , MDB ' nin iki sıralı hata üzerinde durakladığını gösterir.
- *2* , MDB ' nin üç sıralı hata üzerinde durakladığını gösterir ve bu şekilde devam eder.

Bu özelliği, yalnızca WebSphere Libertyiçinde ve Liberty düzeyi 18.0.0.4ya da daha yüksek olduğunda bir etkinleştirme belirtimi için yapılandırabilirsiniz.

Uyarı: Özniteliği Libertydışındaki bir uygulama sunucusu ortamında varsayılan olmayan bir değere ayarlarsanız, değer yoksayılır ve günlüğe bir uyarı iletisi yazılır.

Ayrıca, IBM MQ kaynak bağdaştırıcısını soysal kaynak bağdaştırıcısı olarak WebSphere Liberty içine kurmak da mümkündür. Bu işlem, tüm IBM MQ ve WebSphere Application Server bütünleştirme yeteneklerini devre dışı bırakır ve kaynak bağdaştırıcısının Libertyiçinde çalıştığını saptamasını engeller. Bu nedenle, **`maxSequentialDeliveryFailures`** ayarının *0* değerinden büyük ya da ona eşit olması desteklenmez ve günlükte bir uyarı iletisiyle sonuçlanır.

Yordam

- WebSphere Libertyiçinde, `server.xml`içindeki bir etkinleştirme belirtiminin tanımlamasında **`maxSequentialDeliveryFailures`** özelliğini belirtin.

  Örneğin:

```
<jmsActivationSpec>
     <properties.wmqJms destinationRef="jndi/MDBQ"
                        transportType="BINDINGS"
                        queueManager="MQ21"
                        maxSequentialDeliveryFailures="1"/>
</jmsActivationSpec>
```

İlgili kavramlar

“Kaynak bağdaştırıcısının giden iletişim için yapılandırılması” sayfa 452
Giden iletişim yapılandırmak için bir ConnectionFactory nesnesinin özelliklerini ve denetlenen bir hedef nesneyi tanımlayın.

Kaynak bağdaştırıcısı kuruluşunun doğrulanması

IBM MQ kaynak bağdaştırıcısına ilişkin kuruluş doğrulama sınaması (IVT) programı bir EAR dosyası olarak sağlanır. Programı kullanmak için, programı konuşlandırmanız ve bazı nesneleri JCA kaynakları olarak tanımlamanız gerekir.

Bu görev hakkında

Kuruluş doğrulama sınaması (IVT) programı, `wmq.jakarta.jmsra.ivt.ear` (Jakarta Messaging 3.0) ya da `wmq.jmsra.ivt.ear` (JMS 2.0) adlı bir EAR dosyası olarak sağlanır. Bu dosya, IBM MQ classes for JMS ile birlikte IBM MQ kaynak bağdaştırıcısı RAR dosyası, `wmq.jakarta.jmsra.rar` (Jakarta Messaging 3.0) ya da `wmq.jmsra.rar` (JMS 2.0) ile aynı dizine kurulur. Bu dosyaların nereye kurulduğu hakkında bilgi için bkz. "IBM MQ kaynak bağdaştırıcısının takılması" sayfa 427.

IVT programını uygulama sunucunuzda konuşlandırmanız gerekir. IVT programı, bir iletinin IBM MQ kuyruğuna gönderilebileceğini ve bu kuyruktan alınabileceğini sınayan bir sunucu uygulaması ve MDB içerir. IVT programını kullanarak, IBM MQ kaynak bağdaştırıcısının dağıtımlı hareketleri destekleyecek şekilde doğru yapılandırıldığını doğrulayabilirsiniz. IBM MQ kaynak bağdaştırıcısınıIBM dışı bir uygulama sunucusunda konuşlandırıyorsanız, IBM Service sizden uygulama sunucunuzun doğru yapılandırıldığını doğrulamak için IVT ' nin çalıştığını göstermenizi isteyebilir.

IVT programını çalıştırmadan önce, JCA kaynakları olarak bir ConnectionFactory nesnesi, bir Kuyruk nesnesi ve büyük olasılıkla bir Etkinleştirme Belirtimi nesnesi tanımlamanız ve uygulama sunucunuzun bu tanımlamalardan JMS nesneleri oluşturduğundan ve bunları bir JNDI ad alanına bağladığından emin olmanız gerekir. Nesnelerin özelliklerini, kendi QueueManager' ınızın anasistem ve kapı ayarlarıyla eşleşecek şekilde seçebilirsiniz, ancak aşağıdaki özellikler kümesi basit bir örnektir:

```
ConnectionFactory object:
channel:              SYSTEM.DEF.SVRCONN
hostName:             localhost
port:                 1550
queueManager:         QM1
transportType:        CLIENT
Queue object:
baseQueueManagerName: QM1
baseQueueName:        TEST.QUEUE
```

ConnectionFactory, Kuyruk ve Etkinleştirme Belirtimi nesnelerini tanımlamak için kullanılan düzenek, uygulama sunucunuza bağlı olarak değişir. Örneğin, WebSphere Libertyiçinde bu özellikleri ayarlamak için uygulama sunucusunun `server.xml` dosyasına aşağıdaki girdileri ekleyin:

JMS 3.0
```
<!-- IVT Connection factory -->
<jmsQueueConnectionFactory connectionManagerRef="ConMgrIVT" jndiName="IVTCF">
   <properties.wmqJms channel="SYSTEM.DEF.SVRCONN" hostname="localhost" port="1550"
transportType="CLIENT"/>
</jmsQueueConnectionFactory>
<connectionManager id="ConMgrIVT" maxPoolSize="10"/>

<!-- IVT Queues -->
<jmsQueue id="IVTQueue" jndiName="IVTQueue">
   <properties.wmqJms baseQueueName="TEST.QUEUE"/>
</jmsQueue>

<!-- IVT Activation Spec -->
<jmsActivationSpec id="wmq.jakarta.jmsra.ivt/WMQ_IVT_MDB/WMQ_IVT_MDB">
   <properties.wmqJms destinationRef="IVTQueue"
transportType="CLIENT"
queueManager="QM1"
hostName="localhost"
port="1550"
maxPoolDepth="1"/>
</jmsActivationSpec>
```

JMS 2.0
```
<!-- IVT Connection factory -->
<jmsQueueConnectionFactory connectionManagerRef="ConMgrIVT" jndiName="IVTCF">
   <properties.wmqJms channel="SYSTEM.DEF.SVRCONN" hostname="localhost" port="1550"
transportType="CLIENT"/>
</jmsQueueConnectionFactory>
<connectionManager id="ConMgrIVT" maxPoolSize="10"/>

<!-- IVT Queues -->
<jmsQueue id="IVTQueue" jndiName="IVTQueue">
   <properties.wmqJms baseQueueName="TEST.QUEUE"/>
</jmsQueue>

<!-- IVT Activation Spec -->
```

```
<jmsActivationSpec id="wmq.jmsra.ivt/WMQ_IVT_MDB/WMQ_IVT_MDB">
   <properties.wmqJms destinationRef="IVTQueue"
transportType="CLIENT"
queueManager="QM1"
hostName="localhost"
port="1550"
maxPoolDepth="1"/>
</jmsActivationSpec>
```

Varsayılan olarak IVT programı, JNDI ad alanında jms/ivt/IVTCF adı ve jms/ivt/IVTQueue adıyla bağ tanımlanacak bir ConnectionFactory nesnesinin bağlanmasını bekler. Farklı adlar kullanabilirsiniz, ancak kullanırsanız, IVT programının ilk sayfasındaki nesnelerin adlarını girmeniz ve EAR dosyasını uygun şekilde değiştirmeniz gerekir.

IVT programını konuşlandırdıktan ve uygulama sunucusu JMS nesnelerini yarattıktan ve bunları JNDI ad alanına bağladıktan sonra, aşağıdaki adımları tamamlayarak IVT programını başlatabilirsiniz.

Yordam

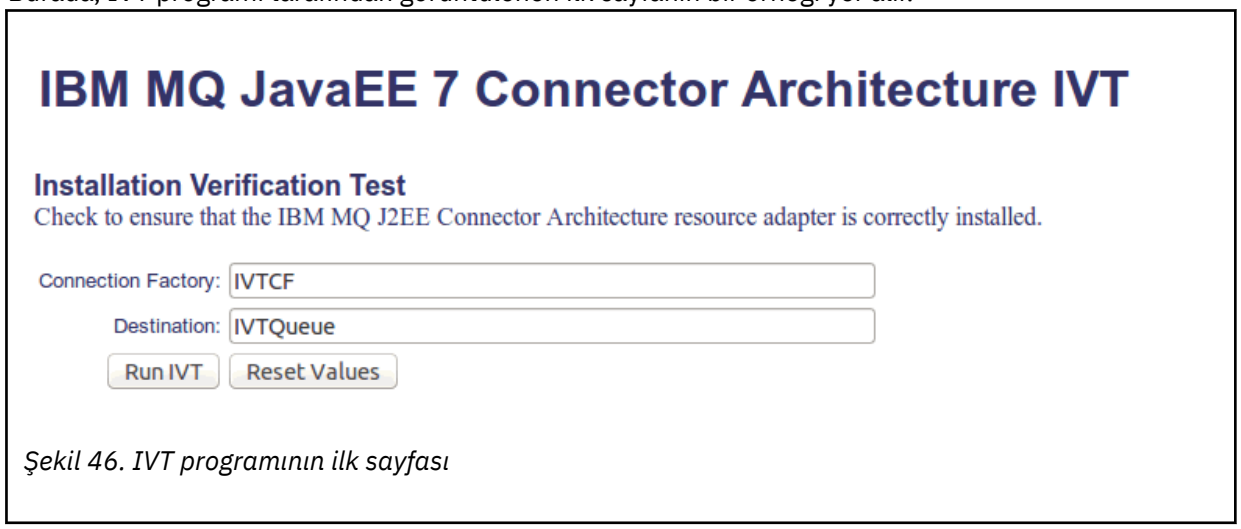
1. Web tarayıcınıza aşağıdaki biçimde bir URL girerek IVT programını başlatın:

   ```
   http://app_server_host: port/WMQ_IVT/
   ```

   Burada *app_server_host* , uygulama sunucunuzun çalıştığı sistemin IP adresi ya da anasistem adıdır; *kapı* ise uygulama sunucusunun dinlediği TCP kapısının numarasıdır. Örnek:

   ```
   http://localhost:9080/WMQ_IVT/
   ```

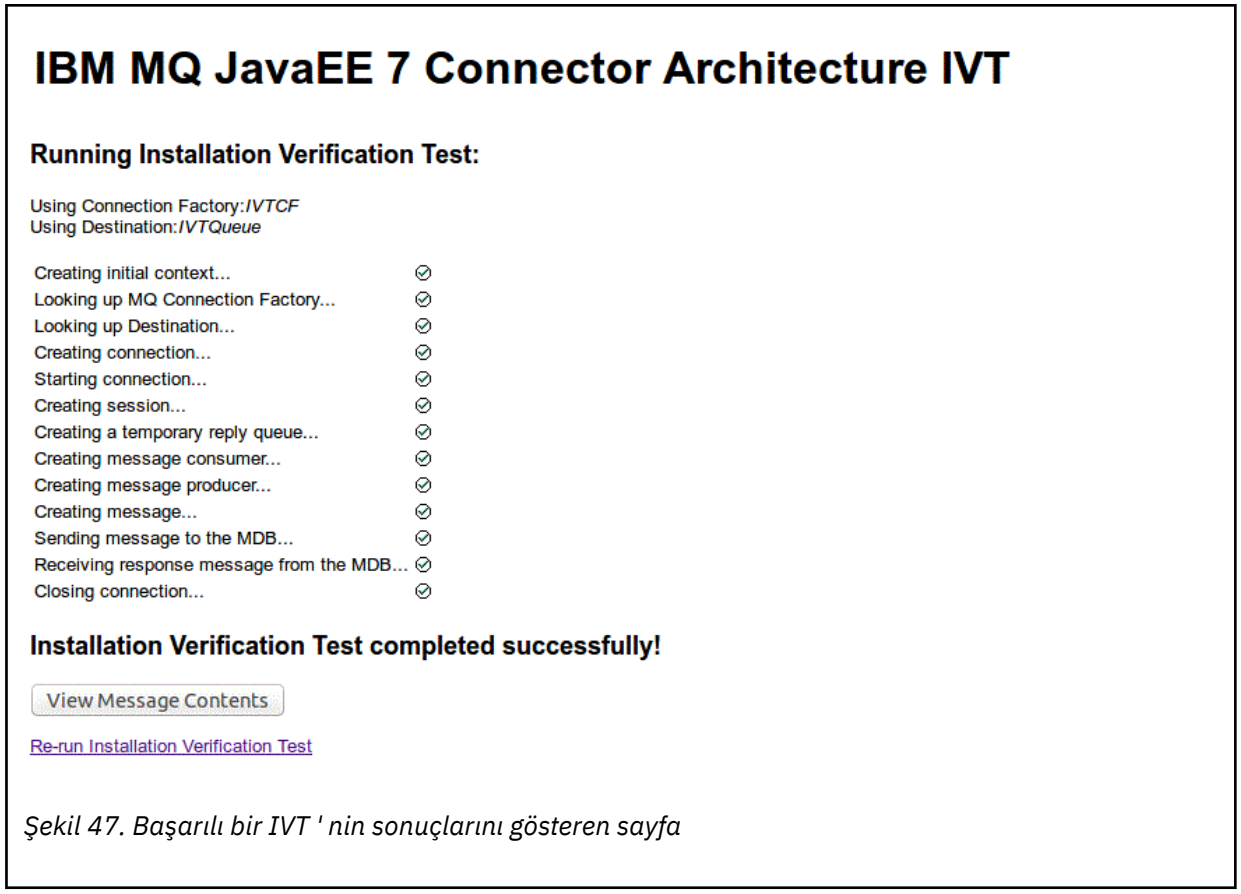
   Burada, IVT programı tarafından görüntülenen ilk sayfanın bir örneği yer alır.

   Şekil 46. IVT programının ilk sayfası

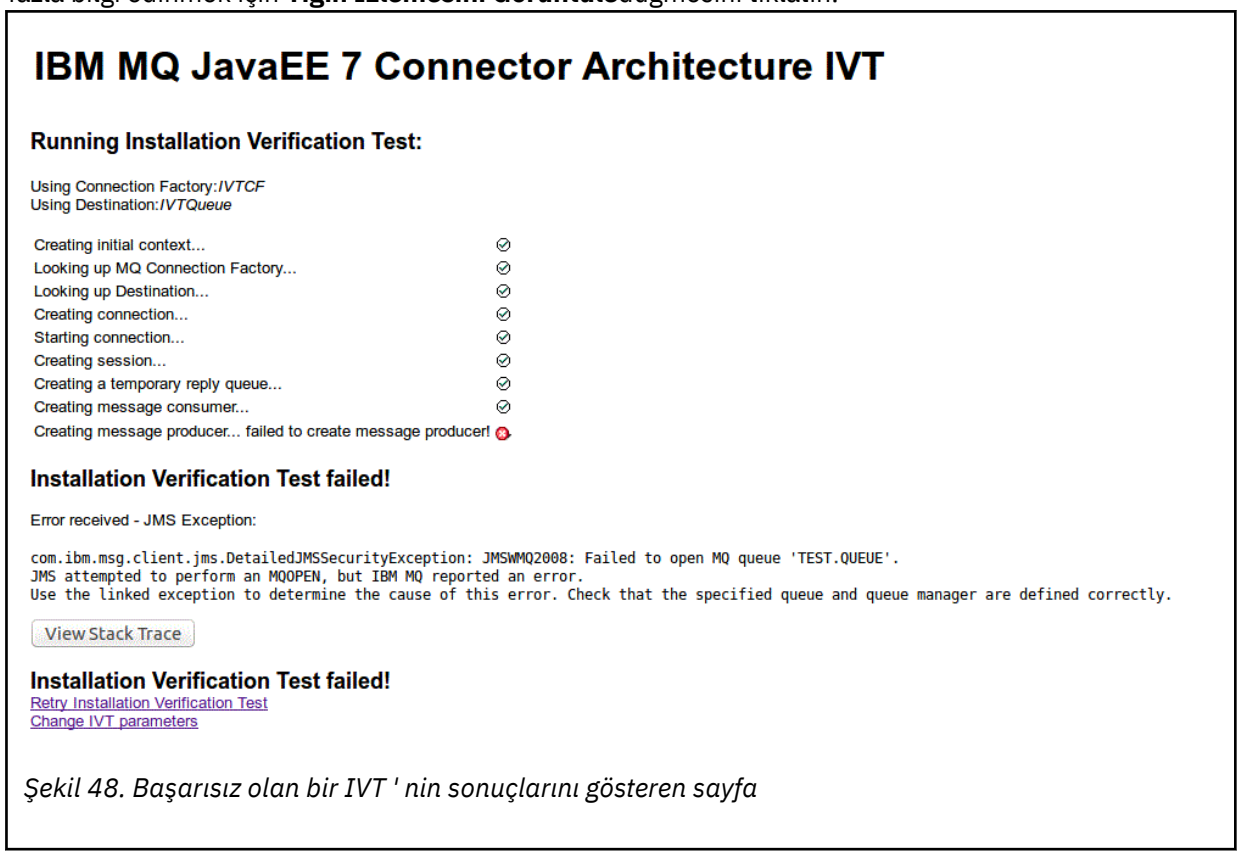
2. Sınamayı çalıştırmak için **IVT Çalıştır**' ı tıklatın.

   Burada, IVT başarılı olursa görüntülenen sayfanın bir örneği yer alır.



Şekil 47. Başarılı bir IVT ' nin sonuçlarını gösteren sayfa

Burada, IVT başarısız olursa görüntülenen sayfanın bir örneği yer alır. Hatanın nedeniyle ilgili daha fazla bilgi edinmek için **Yığın İzlemesini Görüntüle**düğmesini tıklatın.



Şekil 48. Başarısız olan bir IVT ' nin sonuçlarını gösteren sayfa

Windows Kaynak bağdaştırıcısının GlassFish Server 'da kurulması ve sınanması

IBM MQ kaynak bağdaştırıcısını Windows işletim sisteminde GlassFish Server sunucusuna kurmak için öncelikle bir etki alanı oluşturmanız ve başlatmanız gerekir. Daha sonra kaynak bağdaştırıcısını konuşlandırabilir ve yapılandırabilir ve IVT (kuruluş doğrulama testi) uygulamasını konuşlandırabilir ve çalıştırabilirsiniz.

Başlamadan önce

- Bu yönergeler GlassFish Server sürüm 4 için hazırdır.
- GlassFish Server 'ın bu sürümü Jakarta EE' yi desteklemiyor.

Bu görev hakkında

Bu görev, çalışmakta olan bir GlassFish Server uygulama sunucunuzun olduğunu ve bu sunucuya ilişkin standart yönetim görevlerini bildiğinizi varsayar. Bu görev, yerel sisteminizde bir IBM MQ kuruluşu olduğunu ve standart yönetim görevlerini bildiğinizi de varsayar.

Not: Aşağıdaki görev adımlarını tamamlamak için, aşağıdaki nesneler yapılandırılmış olarak çalışan bir IBM MQ kuruluşunuz olmalıdır:

- 1414 numaralı kapıda başlatılan ve SYSTEM.DEF.SVRCONNve İstemci iletimi kullanılarak bağlanır.
- Q1adlı bir kuyruk.

Yordam

1. GlassFish Server **asadmin** kabuk programını başlatın.
   a) Windows komut satırını açın ve *GlassFish*/bin dizinine gidin; burada *GlassFish* , GlassFish Server sürüm 4 'ün kurulu olduğu dizindir.
   b) Komut satırına **asadmin** komutunu girin.
   asadmin komutu, komut satırında yeni bir etki alanı oluşturmanızı sağlayan bir kabuk programı açar.
   GlassFish Server sürüm 4 sisteminizde başlatılır.
2. Bir etki alanı oluşturun ve daha sonra, bir etki alanı başlatın.
   a) Yeni bir etki alanı yaratmak için kapı ve etki alanı adını belirterek **create-domain** komutunu kullanın. Komut satırında şu komutu girin:

   ```
   create-domain --adminport port domain_name
   ```

   Burada *kapı* kapı numarasıdır ve *etki_alanı_adı* , etki alanının kullanmasını istediğiniz addır.

   Not: create-domain komutuyla ilişkilendirilmiş birçok isteğe bağlı parametre vardır. Ancak, bu görev için yalnızca `-- adminport` parametresi gerekir. Daha fazla bilgi için GlassFish Server sürüm 4 ile ilgili ürün belgelerine bakın.

   Belirttiğiniz kapı kullanımdaysa, aşağıdaki ileti görüntülenir:

   etki_alanı_adı kapısı için kapı kullanımda

   Belirlediğiniz etki alanı adı kullanımdaysa, belirlediğiniz adın kullanımda olduğunu ve kullanılamayan tüm etki alanı adlarının bir listesini belirten bir ileti alırsınız.
   b) Bir kullanıcı adı ve parola girmeniz istendiğinde, bir web tarayıcısı aracılığıyla uygulama sunucusunda oturum açmak için kullanılacak kimlik bilgilerini girin.
   Komut başarıyla tamamlanırsa, `Command create-domain executed successfully.` iletisi de içinde olmak üzere, komut satırında etki alanı yaratılmasını özetleyen bir ileti görüntülenir.

   Başarıyla bir etki alanı oluşturdunuz.

c) Komut satırına aşağıdaki komutu girerek etki alanınızı başlatın:

```
start-domain domain_name
```

Burada *etki_alanı_adı* , önceden belirttiğiniz etki alanı adıdır.

3. GlassFish uygulama sunucusuna erişmek için bir web tarayıcısı kullanın.
   a) Bir web tarayıcısının adres çubuğunda şu komutu girin:

```
localhost:port
```

Burada *kapı* , etki alanınızı oluştururken daha önce belirttiğiniz kapıdır.

GlassFish Konsolu görüntülenir.

   b) GlassFish Console yüklendiğinde ve sizden bir kullanıcı adı ve parola istendiğinde, adım 2b' de belirttiğiniz kimlik bilgilerini girin.
4. Kaynak bağdaştırıcısını GlassFish Server 4 'e yükleyin.
   a) **Ortak Görevler** araç çubuğunda, **Uygulamalar** sayfasını görüntülemek için **Uygulamalar** menü öğesini seçin.
   b) **Uygulamaları ya da Modülleri Konuşlandır** sayfasını açmak için **Konuşlandır** düğmesini tıklatın.
   c) **Göz At** düğmesini tıklatın ve `wmq.jmsra.rar` dosyasının konumuna gidin. Dosyayı seçin ve **Tamam**düğmesini tıklatın.
5. Bir bağlantı havuzu yaratın.
   a) Araç çubuğunda **Kaynaklar**' ın altında **Bağlayıcılar** menü öğesini seçin.
   b) Daha sonra, **Bağlayıcı Bağlantı Havuzları** sayfasını açmak için **Bağlayıcı Bağlantı Havuzları** menü öğesini seçin.
   c) **Yeni Bağlayıcı Bağlantı Havuzu (Adım 1/2)** sayfasını açmak için **Yeni** seçeneğini tıklatın.
   d) **Yeni Bağlayıcı Bağlantı Havuzu (Adım 1/2)** sayfasında, havuz adını jms/ivt/IVTCF-Connection-Pool olarak **Pool Name** (Havuz Adı) alanına girin.
   e) **Kaynak Bağdaştırıcısı** alanında `wmq.jmsra`seçeneğini belirleyin.
   f) **Bağlantı Tanımlaması** alanına `javax.jms.ConnectionFactory`girin.
   g) **Next**(İleri) seçeneğini belirleyin ve **Finish**(Son) seçeneğini belirleyin.
6. Bağlayıcı kaynaklarını yaratın.
   a) Araç çubuğunda, **Bağlayıcılar** menüsü altında, **Bağlayıcı Kaynakları** sayfasını açmak için **Bağlayıcı Kaynağı** seçeneğini belirleyin.
   b) **Yeni Bağlayıcı Kaynağı** sayfasını açmak için **Yeni**seçeneğini belirleyin.
   c) **JNDI Adı** alanına `IVTCF`girin.
   d) **Havuz Adı** alanına `jms/ivt/IVTCF-Connection-Pool`girin.
   e) Diğer tüm alanları boş bırakın.
   f) Aşağıdaki özellik/değer çiftlerinin her biri için, **Özellik Ekle**' yi tıklatın ve aşağıdaki örnekte gösterildiği gibi özellik adını ve değeri girin:
      - ad: anasistem; değer: localhost
      - ad: kapı; değer 1414
      - ad: kanal; değer: SYSTEM.DEF.SVRCONN
      - ad: queueManager; değer: QM
      - ad: transportType; değer: CLIENT

      Not: Kendi yapılandırma ayarlarınız için doğru değerleri kullandığınızdan emin olun; bu, bu örnekte gösterilenlerden farklı olabilir.

g) Araç çubuğunda, **Bağlayıcılar**' ın altında, **Yönetim Nesnesi Kaynakları** sayfasını açmak için **Yönetim Nesnesi Kaynakları** menü öğesini seçin.

h) **Denetim Nesnesi Kaynakları** sayfasında, **Yeni Denetim Nesnesi Kaynağı** sayfasını açmak için **Yeni** düğmesini tıklatın.

i) **JNDI Adı** alanına `IVTQueue`girin.

j) **Kaynak Bağdaştırıcısı** alanına `wmq.jmsra`girin.

k) **Kaynak Tipi** alanına `javax.jms.Queue`girin.

l) **Sınıf Adı** alanını olduğu gibi bırakın.

m) Aşağıdaki özellik/değer çiftlerinin her biri için, **Özellik Ekle**' yi tıklatın ve aşağıdaki örnekte gösterildiği gibi özellik adını ve değeri girin:

- ad: ad; değer: IVTQueue
- ad: baseQueueManagerName; değer QM
- ad: baseQueueAd; değer: Q1

Not: Kendi yapılandırma ayarlarınız için doğru değerleri kullandığınızdan emin olun; bu, bu örnekte gösterilenlerden farklı olabilir.

n) **Tamam**'ı tıklatın.

o) **Enabled** (Etkin) onay kutusunu seçin ve **Enable**(Etkinleştir) düğmesini tıklatın.

7. `wmq.jmsra.ivt.ear` EAR dosyasını GlassFish Server sunucusuna konuşlandırın.

a) **Uygulamalar** sayfasını görüntülemek için araç çubuğundaki **Uygulamalar** seçeneğini tıklatın.

b) IVT uygulamasını eklemek için **Deploy** (Konuşlandır) düğmesini tıklatın.

c) **Konum** alanında, `wmq.jmsra.ivt.ear`konumuna gidin ve seçin.

d) **Virtual Servers** (Sanal Sunucular) alanında **server**(sunucu) seçeneğini belirleyin ve **OK**(Tamam) düğmesini tıklatın.

8. IVT programını başlatın.

a) **Uygulamalar** sayfasını görüntülemek için araç çubuğundaki **Uygulamalar** seçeneğini tıklatın.

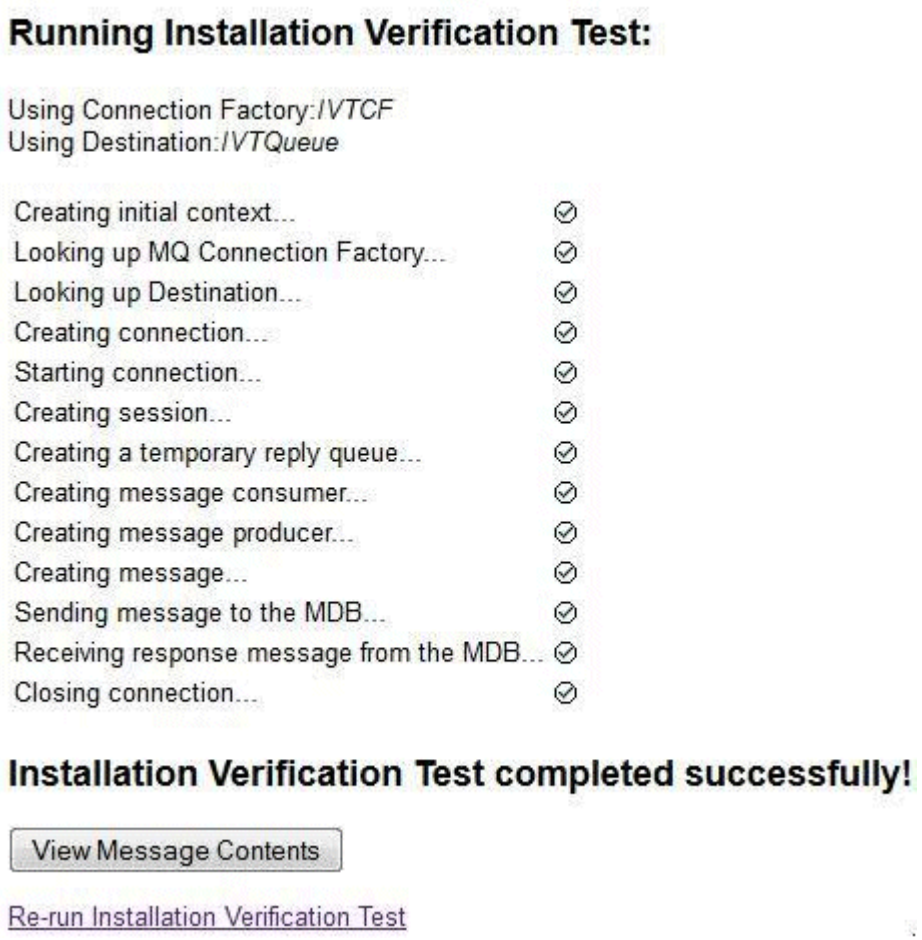
b) Devreye Alınan Uygulamalar tablosunda `wmq.jmsra.ivt` simgesini tıklatın.

c) Modüller ve Bileşenler tablosunda **Başlat** düğmesini tıklatın.

d) http: bağlantısını seçin.

e) **Run IVT**(IVT Çalıştır) seçeneğini tıklatın.

IVT programını başlatmış ve başarılı olursanız aşağıdaki çıktı görüntülenir:



Şekil 49. Başarılı IVT çıkışı

Kaynak bağdaştırıcısının WildFly içinde kurulması ve sınanması

IBM MQ kaynak bağdaştırıcısını WildFly V10içine kuruyorsanız, IBM MQ kaynak bağdaştırıcısı için bir altsistem tanımlaması eklemek üzere bazı yapılanış dosyası değişikliklerini gerçekleştirmeniz gerekir. Daha sonra, kuruluş doğrulama testi (IVT) uygulamasını kurarak ve çalıştırarak kaynak bağdaştırıcısını devreye alabilir ve sınayabilirsiniz.

Başlamadan önce

- Bu yönergeler WildFly V10içindir.
- WildFly 'in bu sürümü Jakarta EE' yi desteklemez.

Bu görev hakkında

Bu görev, çalışmakta olan bir WildFly uygulama sunucunuzun olduğunu ve bu sunucuya ilişkin standart yönetim görevlerini bildiğinizi varsayar. Bu görev, bir IBM MQ kuruluşunuz olduğunu ve standart yönetim görevlerine alışık olduğunuzu da varsayar.

Yordam

1. ExampleQMadlı bir IBM MQ kuyruk yöneticisi yaratın ve “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023içinde açıklandığı gibi ayarlayın.

   Kuyruk yöneticisini ayarlarken aşağıdaki noktalara dikkat edin:

   - Dinleyici 1414 numaralı kapıda başlatılmalıdır.

- Kullanılacak kanala SYSTEM.DEF.SVRCONN.
- IVT uygulaması tarafından kullanılan kuyruk TEST.QUEUE.

Model kuyruğu SYSTEM.DEFAULT.MODEL.QUEUE kuyruğuna DSP ve PUT yetkisi de verilmesi gerekir.

2. *WildFly_Home*/standalone/configuration/standalone-full.xml yapılanış kütüğünü düzenleyin ve aşağıdaki altsistemi ekleyin:

```
<subsystem xmlns="urn:jboss:domain:resource-adapters:4.0">
    <resource-adapters>
        <resource-adapter id="wmq.jmsra">
            <archive>
                wmq.jmsra.rar
            </archive>
            <transaction-support>NoTransaction</transaction-support>
            <connection-definitions>
                <connection-definition class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
                                       jndi-name="java:jboss/jms/ivt/IVTCF" enabled="true"
use-java-context="true"
                                       pool-name="IVTCF">
                    <config-property name="channel">SYSTEM.DEF.SVRCONN
                    </config-property>
                    <config-property
                                      name="hostName">localhost
                    </config-property>
                    <config-property name="transportType">
                        CLIENT
                    </config-property>
                    <config-property name="queueManager">
                        ExampleQM
                    </config-property>
                    <config-property name="port">
                        1414
                    </config-property>
                </connection-definition>
                <connection-definition class-
name="com.ibm.mq.connector.outbound.ManagedConnectionFactoryImpl"
                                       jndi-name="java:jboss/jms/ivt/JMS2CF" enabled="true"
use-java-context="true"
                                       pool-name="JMS2CF">
                    <config-property name="channel">
                        SYSTEM.DEF.SVRCONN
                    </config-property>
                    <config-property name="hostName">
                        localhost
                    </config-property>
                    <config-property name="transportType">
                        CLIENT
                    </config-property>
                    <config-property name="queueManager">
                        ExampleQM
                    </config-property>
                    <config-property name="port">
                        1414
                    </config-property>
                </connection-definition>
            </connection-definitions>
        <admin-objects>
            <admin-object class-name="com.ibm.mq.connector.outbound.MQQueueProxy"
                          jndi-name="java:jboss/jms/ivt/IVTQueue" pool-name="IVTQueue">
                <config-property name="baseQueueName">
                    TEST.QUEUE
                </config-property>
                </admin-object>
            </admin-objects>
        </resource-adapter>
    </resource-adapters>
</subsystem>
```

3. wmq.jmsra.rar dosyasını *WildFly_Home*/standalone/deploymentsdizinine kopyalayarak kaynak bağdaştırıcısını sunucunuza konuşlandırın.
4. wmq.jmsra.ivt.ear dosyasını *WildFly_Home*/standalone/deploymentsdizininde kullanarak IVT uygulamasını konuşlandırın.
5. Bir komut istemi getirerek, *WildFly_Home*/bin dizinine gidip komutu çalıştırarak uygulama sunucusunu başlatın:

```
standalone.bat -c standalone-full.xml
```

6. IVT uygulamasını çalıştırın.

   Daha fazla bilgi için bkz “Kaynak bağdaştırıcısı kuruluşunun doğrulanması” sayfa 470. WildFlyiçin varsayılan URL `http://localhost:8080/WMQ_IVT/'` dir.

IBM MQ ve WebSphere Application Server ' yi birlikte kullanma

WebSphere Application Serveriçindeki IBM MQ ileti alışverişi sağlayıcısı aracılığıyla Java Message Service (JMS) ileti sistemi uygulamaları, IBM MQ sisteminizi JMS ileti sistemi kaynaklarının dış sağlayıcısı olarak kullanabilir.

Bu görev hakkında

Java içinde yazılan ve WebSphere Application Server altında çalışan uygulamalar, ileti alışverişi gerçekleştirmek için Java Message Service (JMS) belirtimini kullanabilir. Bu ortamdaki ileti alışverişi bir IBM MQ kuyruk yöneticisi tarafından sağlanabilir.

IBM MQ kuyruk yöneticisini kullanmanın bir yararı, JMS uygulamalarının bir IBM MQ ağının işlevlerine tam olarak katılmasıdır; bu, uygulamaların çok sayıda platformda çalışan kuyruk yöneticileriyle ileti değiş tokuşu yapmalarına olanak sağlar.

Uygulamalar, kuyruk bağlantısı üreticisi nesnesi için *istemci iletimi* ya da *bağ tanımları iletimi* kullanabilir. Bağ tanımları iletimi için, kuyruk yöneticisinin bağlantı gerektiren uygulama için yerel olarak var olması gerekir.

Varsayılan olarak, IBM MQ kuyruklarında tutulan JMS iletileri, JMS ileti üstbilgisi bilgilerinden bazılarını tutmak için MQRFH2 üstbilgisini kullanır. Birçok kalıt IBM MQ uygulaması bu üstbilgilere sahip iletileri işleyemez ve kendi ayırıcı özellik üstbilgilerini (örneğin, MQCIH for CICS Bridge ya da MQWIH for IBM MQ Workflow uygulamaları) gerektirir. Bu özel noktalara ilişkin ek bilgi için JMS iletilerinin IBM MQ iletilerle eşlenmesibaşlıklı konuya bakın.

İlgili görevler

WebSphere Application Server içinde JMS kaynaklarının yapılandırılması

Uygulama sunucusunun en son kaynak bağdaştırıcısı bakım düzeyini kullanacak şekilde yapılandırılması

WebSphere Application Server ' yi IBM MQ ile kullanma

IBM MQ ve IBM MQ for z/OS , WebSphere Application Serverile birlikte gönderilen varsayılan ileti sistemi sağlayıcısıyla birlikte ya da bu sağlayıcıya alternatif olarak kullanılabilir.

IBM MQ ileti alışverişi sağlayıcısı, WebSphere Application Server' un bir parçası olarak kurulur. Bu, IBM MQ kaynak bağdaştırıcısının bir sürümünü ve kuyruk yöneticisinin uygulama sunucusu tarafından yönetilen XA hareketlerine katılmasını sağlayan IBM MQ Extended Transactional Client işlevini içerir. Kaynak bağdaştırıcısı kullanılarak, ileti odaklı Bean 'ler etkinleştirme belirtimlerini ya da dinleyici kapılarını kullanacak şekilde yapılandırılabilir.

Uygulama sunucusunun desteklenmesi için, IBM MQ kaynak bağdaştırıcısı kuruluş doğrulama sınaması programının uygulama sunucusuna konuşlandırılması ve başarıyla çalıştırılması gerekir. IBM MQ kaynak bağdaştırıcısı kuruluş doğrulama sınaması programı başarıyla çalıştırıldıktan sonra, IBM MQ kaynak bağdaştırıcısı desteklenen herhangi bir IBM MQ kuyruk yöneticisine bağlanabilir.

WebSphere Application Server ile IBM MQ arasındaki JMS bağlantıları

WebSphere Application Serverile kullanılabilecek IBM MQ düzeylerini düşünmeden önce, uygulama sunucusu içinde çalışan Java Message Service (JMS) uygulamalarının IBM MQ kuyruk yöneticilerine nasıl bağlanabileceğini anlamanız önemlidir.

Bir IBM MQ kuyruk yöneticisinin kaynaklarına erişmesi gereken JMS uygulamaları, aşağıdaki iletim tiplerinden birini kullanarak bunu yapabilir:

Bağ Tanımları

Bu iletim, uygulama sunucusu ve kuyruk yöneticisi aynı makineye ve işletim sistemi görüntüsüne kurulduğunda kullanılabilir. BINDINGS kipi kullanılırken, iki ürün arasındaki tüm iletişim IPC (Inter-Process Communication; İşlemler Arası İletişim) kullanılarak yapılır.

IBM MQ ileti alışverişi sağlayıcısı, BINDINGS kipinde bir IBM MQ kuyruk yöneticisine bağlanmak için gereken yerli kitaplıkları içermez. BINDINGS kipi bağlantısını kullanmak için, IBM MQ uygulama sunucusuyla aynı makineye kurulmalı ve kaynak bağdaştırıcısının yerel kitaplık yolu, bu kitaplıkların bulunduğu IBM MQ dizinini gösterecek şekilde yapılandırılmalıdır. Daha fazla bilgi için WebSphere Application Server ürün belgelerine bakın:

- WebSphere Application Server traditionaliçin bkz. IBM MQ ileti alışverişi sağlayıcısını yerli kitaplıklarla yapılandırma.
- WebSphere Libertyiçin bkz. IBM MQ ileti sistemi sağlayıcısını kullanmak için JMS uygulamalarını Liberty 'ye konuşlandırma.

z/OS z/OSişletim sistemlerinde, bir WebSphere Application Server bağlantı üreticisini bağ tanımlama kipinde bir IBM MQ kuyruk yönetimi kipine bağlamak istiyorsanız, WebSphere Application Server STEPLIB birleşiminde doğru IBM MQ kitaplıklarını belirtmeniz gerekir. Daha fazla bilgi için WebSphere Application Server ürün belgelerinde IBM MQ kitaplıkları ve z/OS STEPLIB için WebSphere Application Server başlıklı konuya bakın.

CLIENT

İstemci iletimi, WebSphere Application Server ile IBM MQarasında iletişim kurmak için TCP/IP ' yi kullanır. Uygulama sunucusu ve kuyruk yöneticisi farklı makinelerde bulunduğunda kullanılmanın yanı sıra, iki ürün aynı makineye ve işletim sistemi görüntüsüne kurulduğunda da CLIENT kipi kullanılabilir.

JMS uygulamaları bir BINDINGS_THEN_CLIENT iletim tipini de belirtebilir. Bu iletim tipi kullanıldığında, uygulama başlangıçta BINDINGS kipini kullanarak kuyruk yöneticisine bağlanmayı dener; bunu yapamazsa, CLIENT iletimini dener.

IBM MQ kaynak bağdaştırıcısının hangi sürümünün WebSphere Application Server içinde kurulu olduğunu bulma

IBM MQ kaynak bağdaştırıcısının WebSphere Application Serveriçinde hangi sürümünün kurulu olduğuna ilişkin bilgi için Hangi WebSphere MQ Kaynak Bağdaştırıcısı (RA) sürümü WebSphere Application Serverile birlikte gönderilir?başlıklı teknik nota bakın.

WebSphere Application Server ' in şu anda kullandığı kaynak bağdaştırıcısının düzeyini belirlemek için aşağıdaki Jython ve JACL komutlarını kullanabilirsiniz:

Jython

```
wmqInfoMBeansUnsplit = AdminControl.queryNames("WebSphere:type=WMQInfo,*")
wmqInfoMBeansSplit = AdminUtilities.convertToList(wmqInfoMBeansUnsplit)
for wmqInfoMBean in wmqInfoMBeansSplit: print wmqInfoMBean; print
AdminControl.invoke(wmqInfoMBean, 'getInfo', '')
```

Not: Bu komutu çalıştırmak için bu komutu girdikten sonra **Dön** düğmesini iki kez tıklatmanız gerekir.

JACL

```
set wmqInfoMBeans [$AdminControl queryNames WebSphere:type=WMQInfo,*]
foreach wmqInfoMBean $wmqInfoMBeans {
puts $wmqInfoMBean;
puts [$AdminControl invoke $wmqInfoMBean getInfo [] []]
}
```

Kaynak bağdaştırıcısının güncellenmesi

Uygulama sunucusuyla birlikte kurulan IBM MQ kaynak bağdaştırıcısına ilişkin güncellemeler WebSphere Application Server Düzeltme Paketleri 'nde yer alır. IBM MQ kaynak bağdaştırıcısının güncellenmesi için

Kaynak bağdaştırıcısını güncelle ... WebSphere Application Server Administrative Console olanağındaki olanak önerilmez; bunun yapılması, WebSphere Application Server Düzeltme Paketlerinde sağlanan güncellemelerin etkili olmayacağı anlamına gelir.

MQ_INSTALL_ROOT değişkeni

WebSphere Application Server 7.0dizininden MQ_INSTALL_ROOT yalnızca yerli kitaplıkları bulmak için kullanılır ve kaynak bağdaştırıcısında yapılandırılan herhangi bir yerli kitaplık yolu tarafından geçersiz kılınır.

WebSphere Application Server ile IBM MQ arasında bağlantı kurulması

Uyarı:

1. Desteklenen bir WebSphere Application Server sürümü, desteklenen herhangi bir IBM MQsürümüne bağlanmak için onunla birlikte paketlenen IBM MQ kaynak bağdaştırıcısını kullanabilir.
2. Bağ tanımlama kipi kullanılırsa, WebSphere Application Server içindeki bazı kitaplıkların, bağlandığı kuyruk yöneticisinin sürümüyle eşleşmesi gerekir:
   - WebSphere Application Server , IBM MQ 9.3ile birlikte sağlanan yerel kitaplıkları yükleyecek şekilde yapılandırılmalıdır. Ek bilgi için bkz. “Java Native Interface (JNI) kitaplıklarının yapılandırılması” sayfa 92 .
   - z/OSz/OSsistemlerinde, WebSphere Application Server STEPLIB birleşiminde doğru IBM MQ kitaplıklarını belirtmeniz gerekir.

     Gereksinim duyduğunuz IBM MQ kitaplıklarının ayrıntıları için bkz. IBM MQ kitaplıkları ve WebSphere Application Server for z/OS STEPLIB .

     LINKLIST (LINKLST) içinde bir IBM MQ sürümü için kitaplığınız varsa, kitaplıkları STEPLIB ile geçersiz kılarak farklı bir IBM MQ sürümüne bağlanabilirsiniz.
3. IBM MQ Resource Adapter sürümü, kuyruk yöneticisi kuruluşu tarafından sağlanan yerel (paylaşılan) kitaplık sürümlerinden bağımsızdır.

   Örneğin, bir IBM MQ 8.0 Kaynak Bağdaştırıcısı ile WebSphere Application Server 8.5, IBM MQ 9.0 yerel kitaplıklarını kullanarak bir IBM MQ 9.0 kuyruk yöneticisine bağ tanımlama bağlantısını yönetmeye devam edebilir.

Daha fazla bilgi için bkz “IBM MQ kaynak bağdaştırıcısı destek bildirimi” sayfa 421.

BINDINGS ve CLIENT iletim tipleri, WebSphere Application Server'un herhangi bir sürümünden IBM MQ ' e bağlanmak için kullanılabilir. BINDINGS iletim tipi için aşağıdaki kısıtlamalar geçerlidir:

- IBM MQ , uygulama sunucusuyla aynı makineye kurulmalıdır.
- WebSphere Application Server , IBM MQile birlikte sağlanan yerel kitaplıkları yükleyecek şekilde yapılandırılmalıdır.
- z/OSz/OSsistemlerinde, bir WebSphere Application Server bağlantı üreticisini bağ tanımlama kipinde bir IBM MQ kuyruk yönetimi kipine bağlamak istiyorsanız, WebSphere Application Server STEPLIB birleştirmesinde doğru IBM MQ kitaplıklarının belirtilmesi gerekir.

Aşağıdaki çizelge, IBM MQ kaynak bağdaştırıcısının her sürümünün çalıştırılmak üzere desteklendiği WebSphere Application Server sürümlerini göstermektedir.

Çizelge 70. WebSphere Application Server sürümleri IBM MQ kaynak bağdaştırıcısı sürümleriyle eşleniyor.	
IBM MQ kaynak bağdaştırıcısının sürümü	**Kaynak bağdaştırıcısının bu sürümü hangi WebSphere Application Server sürümünde çalışabilir?**
IBM MQ 9.0 ve daha sonra	Kaynak bağdaştırıcısı aşağıdaki yerde çalışabilir: • WebSphere Liberty' in herhangi bir Java EE 7 uyumlu sürümü. • WebSphere Application Server traditional 9.0
IBM MQ 8.0	Kaynak bağdaştırıcısı, WebSphere Liberty ' un herhangi bir Java EE 7 uyumlu sürümünde çalışabilir IBM MQ 8.0 kaynak bağdaştırıcısının WebSphere Application Server traditionaliçinde çalıştırılması desteklenmez. WebSphere Application Server traditional ' da kurulu olan kaynak bağdaştırıcısı, IBM MQ 8.0 kuyruk yöneticilerine bağlanmak için kullanılmalıdır.

İlgili kavramlar

“IBM MQ kaynak bağdaştırıcısı destek bildirimi” sayfa 421
Bir uygulama ile kuyruk yöneticisi arasındaki iletişim için kullanmanız gereken IBM MQ kaynak bağdaştırıcısı, Jakarta Messaging 3.0 API' yi mi, yoksa JMS 2.0 API ' yı mı kullandığınıza bağlıdır.

İlgili bilgiler

IBM MQ için Sistem Gereksinimleri

WebSphere Application Server ile IBM MQ arasında yaratılan TCP/IP bağlantılarının sayısının belirlenmesi

Paylaşım etkileşimleri özelliğini kullanarak birden çok etkileşim MQI kanalı eşgörünümlerini paylaşabilir; bu, TCP/IP bağlantısı olarak da bilinir.

Bu görev hakkında

IBM MQ ileti alışverişi sağlayıcısı normal kipini kullanan WebSphere Application Server 7 ve WebSphere Application Server 8içinde çalışan uygulamalar otomatik olarak bu özelliği kullanır. Bu, aynı IBM MQ kuyruk yöneticisine bağlanan aynı uygulama sunucusu yönetim ortamında çalışan birden çok uygulamanın aynı kanal yönetim ortamını paylaşabildiği anlamına gelir.

Tek bir kanal örneğinde paylaşılabilen etkileşimlerin sayısı, IBM MQ kanal özelliği **SHARECNV**tarafından belirlenir. Sunucu bağlantısı kanalları için bu özelliğin varsayılan değeri 10 'dur.

WebSphere Application Server 7 ve WebSphere Application Server 8tarafından oluşturulan etkileşimlerin sayısına bakarak, oluşturulan kanal örneklerinin sayısını belirleyebilirsiniz.

IBM MQ ileti alışverişi sağlayıcısı kipine ilişkin ek bilgi için PROVIDERVERSION normal kipibaşlıklı konuya bakın.

İlgili kavramlar

Paylaşım sohbetlerini kullanma
Sohbet paylaşımına izin verilen bir ortamda, sohbetler bir MQI kanalı eşgörünümünü paylaşabilir.

“IBM MQ classes for JMS içinde TCP/IP bağlantısının paylaşılması” sayfa 304
Tek bir TCP/IP bağlantısını paylaşmak için birden çok MQI kanalı eşgörünümü yapılabilir.

JMS bağlantı üreticileri

Bağlantılar ve oturumlar oluşturmak için IBM MQ ileti sistemi sağlayıcısı bağlantı üreticisini kullanan WebSphere Application Servericinde çalışan uygulamalar, bağlantı üreticisinden oluşturulan her JMS bağlantısı için ve bir JMS bağlantısından oluşturulan her JMS oturumu için etkin etkileşimler içerir.

Bağlantı üreticisinden oluşturulan her JMS bağlantısı için bir etkileşim

Her JMS bağlantı üreticisinin ilişkili bir bağlantı havuzu vardır; bu havuz serbest havuz ve etkin havuz olmak üzere iki bölüme ayrılmıştır. Başlangıçta her iki havuz da boştur.

Bir uygulama bir bağlantı üreticisinden JMS bağlantısı oluşturduğunda, WebSphere Application Server boş havuzda JMS bağlantısı olup olmadığını denetler. Varsa, etkin havuza taşınır ve uygulamaya verilir. Tersi durumda, yeni bir JMS bağlantısı yaratılır, etkin havuza konup uygulamaya döndürülür. Bir bağlantı üreticisinden yaratılabilecek bağlantı sayısı üst sınırı, **Maximum connections**bağlantı üreticisi bağlantı havuzu özelliğiyle belirlenir. Bu özelliğin varsayılan değeri 10 'dur.

Bir uygulama JMS bağlantısı ile bittikten ve kapatıldıktan sonra, bağlantı etkin havuzdan yeniden kullanılabilir olduğu serbest havuza taşınır. **Unused timeout** bağlantı havuzu özelliği, bir JMS bağlantısının bağlantısı kesilmeden önce boş havuzda ne kadar süre kalabileceğini tanımlar. Bu özelliğin varsayılan değeri 1800 saniyedir (30 dakika).

JMS bağlantısı ilk oluşturulduğunda, WebSphere Application Server ile IBM MQ arasında bir etkileşim başlar. Serbest havuz için **Unused timeout** özelliğinin değeri aşıldığında bağlantı kapatılıncaya kadar etkileşim etkin kalır.

JMS bağlantısından oluşturulan her JMS oturumu için bir etkileşim

Bir IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticisinden oluşturulan her JMS bağlantısının ilişkili bir JMS oturum havuzu vardır. Oturum havuzları, bağlantı havuzlarıyla aynı şekilde çalışır. JMS Tek bir JMS bağlantısından yaratılabilecek oturum sayısı üst sınırı, bağlantı üreticisi oturum havuzu özelliği tarafından belirlenir **Maximum connections**. Bu özelliğin varsayılan değeri 10 'dur.

Bir etkileşim JMS oturumu ilk oluşturulduğunda başlar, JMS oturumu kapatılıncaya kadar etkileşim etkin kalır; bu nedenle, oturum havuzu için **Unused timeout** özelliğinin değerinden daha uzun bir süre boş havuzda kalır.

SHARECNV özelliği için bir değer hesaplama

Aşağıdaki formülü kullanarak tek bir bağlantı üreticisinden IBM MQ ürününe etkileşim sayısı üst sınırını hesaplayabilirsiniz:

```
Maximum number of conversations =
    connection Pool Maximum Connections +
   (connection Pool Maximum Connections * Session Pool Maximum Connections)
```

Bu sayıda etkileşimlerin gerçekleşmesine izin vermek için oluşturulacak kanal örneklerinin sayısı, aşağıdaki hesaplama kullanılarak hesaplanabilir:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplamadan geriye kalan her şey yukarı yuvarlanır.

Bağlantı havuzu **Maximum connections** ve oturum havuzu **Maximum connections** özellikleri için varsayılan değeri kullanan basit bir bağlantı üreticisi için, bu bağlantı üreticisi için WebSphere Application Server ile IBM MQ arasında var olabilecek etkileşim sayısı üst sınırı şöyledir:

```
Maximum number of conversations =
   connection Pool Maximum Connections +
  (connection Pool Maximum Connections * Session Pool Maximum Connections)
```

Örneğin:

```
        = 10 + (10 * 10)
        = 10 + 100
        = 110
```

Bu bağlantı üreticisi, **SHARECNV** özelliği 10 olarak ayarlanmış bir kanal kullanarak IBM MQ ' e bağlanıyorsa, bu bağlantı üreticisi için yaratılacak kanal örneği sayısı üst sınırı şöyledir:

```
Maximum number of channel instances = Maximum number of conversations / SHARECNV for the
channel being used
```

Örneğin:

```
        = 110 / 10
        = 11 (rounded up to nearest connection)
```

Etkinleştirme belirtimleri

Etkinleştirme belirtimini kullanmak üzere yapılandırılan, iletiyle yönlendirilen bean uygulamalarının, bir JMS hedefini izlemek için etkinleştirme belirtimi için etkin etkileşimleri ve iletileri işlemek için ileti odaklı bean eşgörünümünü çalıştırmak için kullanılan her sunucu oturumu için etkileşimleri vardır.

Bir etkinleştirme belirtimini kullanacak şekilde yapılandırılmış, ileti odaklı bean uygulamaları için aşağıdaki etkileşimler etkindir:

- Uygun iletiler için bir JMS hedefini izlemek üzere etkinleştirme belirtimine ilişkin bir etkileşim. Bu etkileşim, etkinleştirme belirtimi başlar başlamaz başlar ve etkinleştirme belirtimi duruncaya kadar etkin kalır.
- İletileri işlemek için ileti odaklı bir Bean eşgörünümünü çalıştırmak üzere kullanılan her sunucu oturumu için bir etkileşim.

Etkinleştirme belirtimi gelişmiş özelliği **Maximum server sessions** , belirli bir etkinleştirme belirtimi için herhangi bir zamanda etkin olabilecek sunucu oturumu sayısı üst sınırını belirtir. Bu özellik varsayılan değer olan 10 değerine sahiptir. Sunucu oturumları gerektiğinde oluşturulur ve **Server session pool timeout**etkinleştirme belirtimi gelişmiş özelliği tarafından belirtilen süre boyunca boşta kaldıysa kapatılır. Bu özelliğin varsayılan değeri 300000 milisaniyedir (5 dakika).

Etkileşimler, bir sunucu oturumu oluşturulduğunda başlar ve etkinleştirme belirtimi durdurulduğunda ya da bir sunucu oturumu zamanaşımına uğradığında durdurulur.

Bu, tek bir etkinleştirme belirtiminden IBM MQ ' e kadar olan etkileşim sayısı üst sınırının aşağıdaki formül kullanılarak hesaplanabileceği anlamına gelir:

```
Maximum number of conversations = Maximum server sessions + 1
```

Bu sayıda etkileşimlerin gerçekleşmesine izin vermek için oluşturulan kanal örneklerinin sayısı, aşağıdaki hesaplama kullanılarak bulunabilir:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplamadan kalan herhangi bir kalanlar yukarı yuvarlanır.

Maximum server sessions özelliği için varsayılan değeri kullanan basit bir etkinleştirme belirtimi için, bu etkinleştirme belirtimi için WebSphere Application Server ile IBM MQ arasında var olabilecek etkileşim sayısı üst sınırı şu şekilde hesaplanır:

```
Maximum number of conversations = Maximum server sessions + 1
```

Örneğin:

```
    = 10 + 1
    = 11
```

Bu etkinleştirme belirtimi, **SHARECNV** özelliği 10 olarak ayarlanmış bir kanal kullanılarak IBM MQ ' e bağlanıyorsa, oluşturulan kanal eşgörünümlerinin sayısı şu şekilde hesaplanır:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Örneğin:

```
    = 11 / 10
    = 2 (rounded up to nearest connection)
```

Application Server Facilities (ASF) kipinde çalışan dinleyici kapıları

İletiyle yönlendirilen Bean uygulamaları tarafından kullanılan ASF kipinde çalışan dinleyici kapıları, her sunucu oturumu için etkileşimler yaratır. Biri uygun iletiler için bir hedefi izler ve diğeri iletileri işlemek için iletiyle yönlendirilen bir Bean eşgörünümünü çalıştırır. Her bir dinleyici kapısı için etkileşim sayısı, oturum sayısı üst sınırından hesaplanabilir.

Varsayılan olarak, dinleyici kapıları, iletileri algılamak ve işlemek üzere ileti odaklı Bean 'lere teslim etmek için hangi uygulama sunucularının kullanması gerektiğini tanımlayan 1.1 belirtiminin bir parçası olarak ASF kipinde çalışır. Bu varsayılan işlem kipinde dinleyici kapılarını kullanmak üzere ayarlanan ileti odaklı bean uygulamaları etkileşimler yaratır:

Dinleyici kapısının uygun iletilere ilişkin bir hedefi izlemesi için bir etkileşim
: Dinleyici kapıları bir JMS bağlantı üreticisini kullanacak şekilde yapılandırıldı. Bir dinleyici kapısı başlatıldığında, bağlantı üreticisi serbest havuzundan JMS bağlantısı için bir istek yapılır. Dinleyici kapısı durdurulduğunda, bağlantı boş havuza döndürülür. Bağlantı havuzunun nasıl kullanıldığına ve bunun IBM MQile olan etkileşim sayısını nasıl etkilediğine ilişkin daha fazla bilgi için bkz. “JMS bağlantı üreticileri” sayfa 483.

İletileri işlemek için ileti odaklı bean eşgörünümünü çalıştırmak üzere kullanılan her sunucu oturumu için bir etkileşim
: Dinleyici kapısı özelliği **Maximum sessions** , belirli bir dinleyici kapısı için herhangi bir zamanda etkin olabilecek sunucu oturumu sayısı üst sınırını belirtir. Bu özelliğin varsayılan değeri 10 'dur. Sunucu oturumları gerektiği şekilde yaratılır ve dinleyici kapısının kullandığı JMS bağlantısıyla ilişkili oturum havuzundan alınan JMS oturumlarını kullanır.

Bir sunucu oturumu İleti Dinleyici Hizmeti özel özelliği **SERVER.SESSION.POOL.UNUSED.TIMEOUT**tarafından belirtilen süre boyunca boşta durduysa, oturum kapanır ve kullanılan JMS oturumu oturum havuzu boş havuzuna döndürülür. JMS oturumu, gerekli oluncaya kadar oturum havuzunda serbest kalır ya da boş havuzda oturum havuzunun **Unused timeout** özelliğinden daha uzun süre boşta kaldığı için kapanır.

Oturum havuzunun nasıl kullanıldığına ve WebSphere Application Server ile IBM MQ arasındaki etkileşimlerin nasıl yönetildiğine ilişkin ek bilgi için bkz. “JMS bağlantı üreticileri” sayfa 483.

İleti Dinleyici Hizmeti özel özelliği **SERVER.SESSION.POOL.UNUSED.TIMEOUT**, WebSphere Application Server ürün belgelerinde Monitoring server session pools for listener ports başlıklı konuya bakın.

Tek bir dinleyici kapısından IBM MQ ' e etkileşim sayısı üst sınırının hesaplanması

Aşağıdaki formülü kullanarak tek bir dinleyici kapısından IBM MQ ' e etkileşim sayısı üst sınırını hesaplayabilirsiniz:

```
Maximum number of conversations = Maximum sessions + 1
```

Bu sayıda etkileşimlerin gerçekleşmesine izin vermek için oluşturulacak kanal örneklerinin sayısı, aşağıdaki hesaplama kullanılarak hesaplanabilir:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplamadan geriye kalan her şey yukarı yuvarlanır.

Maximum sessions özelliği için varsayılan değeri kullanan basit bir dinleyici kapısı için, bu dinleyici kapısı için WebSphere Application Server ile IBM MQ arasında var olabilecek etkileşim sayısı üst sınırı şu şekilde hesaplanır:

```
Maximum number of conversations = Maximum sessions + 1
```

Örneğin:

```
= 10 + 1
= 11
```

Bu dinleyici kapısı, **SHARECNV** özelliği 10 olarak ayarlanmış bir kanal kullanarak IBM MQ ' e bağlanıyorsa, yaratılacak kanal eşgörünümlerinin sayısı şu şekilde hesaplanır:

```
Maximum number of channel instances =
    Maximum number of conversations / SHARECNV for the channel being used
```

Örneğin:

```
= 11 / 10
= 2 (rounded up to nearest connection)
```

Uygulama Sunucusu Olmayan Tesisler (ASF dışı) kipinde çalışan dinleyici kapıları

ASF dışı kipte çalışan dinleyici kapılarının konfigürasyonu, sunucu oturumlarını kullanarak kuyruk hedefini ve konu hedefini izleyecek şekilde tanımlanabilir. Sunucu oturumlarının birden çok etkileşimleri olabilir; bu etkileşimlerin sayısı her durumda hesaplanabilir.

Dinleyici kapıları, dinleyici kapılarının JMS hedeflerini izleme şeklini değiştiren ASF dışı kipte çalışacak şekilde yapılandırılabilir. İletiyle yönlendirilen bean uygulamaları, ASF olmayan işlem kipinde dinleyici kapılarını kullanarak, iletileri işlemek için ileti odaklı bir Bean eşgörünümünü çalıştırmak üzere kullanılan her sunucu oturumu için bir etkileşim yaratır. Dinleyici kapısı özelliği **oturum sayısı üst sınırı** , belirli bir dinleyici kapısı için herhangi bir zamanda etkin olabilecek Sunucu Oturumu sayısı üst sınırını belirtir. Bu özelliğin varsayılan değeri 10 'dur.

ASF dışı kipte çalışırken, bir dinleyici kapısı izleme hedefi, dinleyici kapısı özelliği **Oturum sayısı üst sınırı**tarafından belirlenen Sunucu Oturumu sayısını otomatik olarak yaratır. Bu Sunucu Oturumlarının tümü, dinleyici kapısının kullandığı JMS Connection ile ilişkili oturum havuzundan alınan JMS Oturumlarını kullanır ve uygun iletiler için sürekli olarak bir JMS Hedef 'i izler.

Dinleyici kapısı bir konu hedefini izleyecek şekilde yapılandırıldıysa, **Oturum sayısı üst sınırı** değeri yoksayılır ve tek bir Sunucu Oturumu kullanılır.

ASF olmayan kipte çalışan bir dinleyici kapısı tarafından kullanılan Sunucu Oturumları, dinleyici kapısı durdurulduğa kadar etkin kalır; bu noktada, kullanılan JMS Oturumları, dinleyici kapısının kullandığı JMS Connection için oturum havuzu Serbest Havuzuna döndürülür.

Oturum havuzunun nasıl kullanıldığına ve WebSphere Application Server ile IBM MQ arasındaki etkileşimlerin nasıl yönetildiğine ilişkin ek bilgi için bkz. “JMS bağlantı üreticileri” sayfa 483.

WebSphere Application Serverile ASF ve ASF dışı işletim kipine ilişkin ek bilgi ve ASF dışı kipi kullanmak üzere Dinleyici Kapılarının nasıl yapılandırılacağına ilişkin bilgi için ASF kipinde ve ASF dışı kipte ileti işlemebaşlıklı konuya bakın.

Bir kuyruk hedefi izlenirken etkileşim sayısı üst sınırının hesaplanması

ASF dışı kipte çalışan ve IBM MQ için bir kuyruk hedefini izleyen tek bir dinleyici kapısından gelen etkileşim sayısı üst sınırı aşağıdaki formül kullanılarak hesaplanabilir:

```
Maximum number of conversations = Maximum sessions
```

Bu etkileşim sayısına izin vermek için oluşturulacak kanal örneklerinin sayısı, aşağıdaki hesaplama kullanılarak bulunabilir:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplamadan geriye kalan her şey yukarı yuvarlanır.

Oturum sayısı üst sınırı özelliği için varsayılan değeri kullanan ve bir kuyruk hedefini izleyen ASF dışı kipte çalışan basit bir dinleyici kapısı için, bu dinleyici kapısı için WebSphere Application Server ile IBM MQ arasında var olabilecek etkileşim sayısı üst sınırı:

```
Maximum number of conversations = Maximum sessions
```

Örneğin:

```
= 10
```

Bu dinleyici kapısı, **SHARECNV** özelliği 10 olarak ayarlanmış bir kanal kullanılarak IBM MQ ' e bağlanıyorsa, oluşturulan kanal eşgörünümlerinin sayısı şu şekilde hesaplanır:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Örneğin:

```
= 10 / 10
= 1
```

Bir konu hedefini izlerken etkileşim sayısı üst sınırını hesaplama

ASF dışı kipte çalışan ve bir konu hedefini izleyecek şekilde yapılandırılmış bir dinleyici kapısı için, dinleyici kapısından IBM MQ ' e kadar olan etkileşimlerin sayısı:

```
Maximum number of conversations = 1
```

Bu etkileşim sayısına izin vermek için oluşturulacak kanal örneklerinin sayısı, aşağıdaki hesaplama kullanılarak bulunabilir:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplamadan geriye kalan her şey yukarı yuvarlanır.

Oturum sayısı üst sınırı özelliği için varsayılan değeri kullanan ve bir konu hedefini izleyen ASF dışı kipte çalışan basit bir dinleyici kapısı için, bu dinleyici kapısı için WebSphere Application Server ile IBM MQ arasında var olabilecek etkileşim sayısı üst sınırı:

```
Maximum number of conversations = Maximum sessions
```

Örneğin:

```
= 10
```

Bu dinleyici kapısı, **SHARECNV** özelliği 10 olarak ayarlanmış bir kanal kullanılarak IBM MQ ' e bağlanıyorsa, oluşturulan kanal eşgörünümlerinin sayısı şu şekilde hesaplanır:

```
Maximum number of channel instances =
   Maximum number of conversations / SHARECNV for the channel being used
```

Örneğin:

```
= 10 / 10
= 1
```

IBM MQ ile WebSphere Application Server bağlantısını güvenli kılmak için kimlik doğrulama diğer adlarını yapılandırma

Kimlik doğrulama diğer adları, IBM MQile WebSphere Application Server bağlantısını güvenli kılmak için kullanılabilecek bir kullanıcı adı ve parola birleşimiyle eşler. Kimlik doğrulama diğer adıyla bir bağlantı üreticisi yapılandırabilirsiniz.

Kurumsal uygulamalarla kimlik doğrulama diğer adlarını kullanma

WebSphere Application Server içinde çalışan bir kurum uygulaması IBM MQile bir JMS bağlantısı oluşturmayı denediğinde, uygulama, uygulama sunucusunun Java Naming Directory Interface (JNDI) havuzundan IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticisi tanımlamasını arar.

IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticisi tanımlaması uygulama sunucusunun JNDI havuzundan bulunduğunda, aşağıdaki yöntemlerden biri çağrılır:

- `ConnectionFactory.createConnection()`
- `ConnectionFactory.createConnection(String username, String password)`

Bağlantı üreticisi tanımlı bir J2C kimlik doğrulama diğer adıyla yapılandırıldıysa, bağlantı oluşturmak için bağlantı üreticisi kullanıldığında kimlik doğrulama diğer adındaki kullanıcı adı ve parola IBM MQ ' e akıtılabilir.

Bağlantı üreticileri ve kimlik doğrulama diğer adları

IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticileri, IBM MQ kuyruk yöneticilerine nasıl bağlanılacağına ilişkin bilgileri içerir. WebSphere Application Server içinde çalışan kurumsal uygulamalar, IBM MQile JMS bağlantıları oluşturmak için bağlantı üreticilerini kullanabilir.

WebSphere Application Server , bağlantı üreticileri tanımlamalarını JNDIkullanılarak erişilebilen bir havuzda saklar. Bir bağlantı üreticisi yaratıldığında, bağlantı üreticisine, tanımlandığı uygulama sunucusu kapsamında (Hücre, Düğüm ya da Sunucu kapsamı) bu bağlantı üreticisini benzersiz olarak tanıması için bir JNDI adı verilir.

Örneğin, WebSphere Application Server Hücre kapsamında tanımlanan bir IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticisi, BINDINGS iletimi kullanılarak kuyruk yöneticisine (`myQM`) nasıl bağlanılacağına ilişkin bilgileri içerir. Bu bağlantı üreticisine, benzersiz olarak tanımlamak için JNDI ad `jms/myCF` verilir.

Bağlantı üreticileri, kimlik doğrulama diğer adını kullanacak şekilde de yapılandırılabilir. Kimlik doğrulama diğer adları, bir kullanıcı adı ve parola birleşimiyle eşler. Bağlantı üreticisinin nasıl kullanıldığına bağlı olarak, kimlik doğrulama diğer adındaki kullanıcı adı ve parola, JMS bağlantısı oluşturulduğunda IBM MQ ' e akıtılabilir ya da akmayabilir.

Önemli: IBM MQ 8.0'den önce, varsayılan IBM MQ Object Authority Manager (OAM) bir yetki denetimi gerçekleştirdi; yalnızca, bağlantı kurulduğunda IBM MQ' e iletilen kullanıcı adının kuyruk yöneticisine erişme yetkisine sahip olduğundan emin olmak için.

Belirtilen parolayı doğrulamak için herhangi bir denetim yapılmadı. Bir kimlik doğrulama denetimi gerçekleştirmek ve kullanıcı kimliği ile parolanın eşleştiğini doğrulamak için bir IBM MQ kanal güvenlik çıkışı yazmanız gerekir. Bunun nasıl yapılacağına ilişkin ayrıntılar için bkz. “Kanal güvenliği çıkış programları” sayfa 928.

Kuyruk yöneticisi, IBM MQ 8.0' den kullanıcı adının yanı sıra parolayı da denetler.

Bağlantı üreticisinin kullanılması

Aşağıdaki konularda, doğrudan ve dolaylı aramalar kullanılarak bağlantı üreticisinin kullanılmasıyla ilgili bilgiler yer alır:

- “Doğrudan arama yoluyla bağlantı üreticisinin kullanılması” sayfa 492
- “Dolaylı arama yoluyla bağlantı üreticisinin kullanılması” sayfa 493

CLIENT iletimini kullanma

CLIENT iletimini kullanmak üzere yapılandırılan bağlantı üreticileri, kuyruk yöneticisine bağlanmak için kullanacakları IBM MQ sunucu bağlantı kanalını (SVRCONN) belirtmelidir.

Bağlantı üreticisinin kullanmak üzere yapılandırıldığı kanal için IBM MQ kanal aracısı kullanıcı kimliği (MCAUSER) özelliği boş kalırsa, bağlantı üreticisi doğrudan aramayla ya da dolaylı aramayla kullanılabilir.

MCAUSER özelliği bir kullanıcı kimliğine ayarlanırsa, bu kullanıcı kimliği, kurumsal uygulamanın doğrudan ya da dolaylı bir aramanın kullanılıp kullanılmadığına bakılmaksızın, IBM MQile bağlantı oluşturmak için bağlantı üreticisi kullanıldığında IBM MQ ' e aktarılır.

Özet tabloları

Aşağıdaki çizelgelerde, sırasıyla BINDINGS iletimi ve CLIENT iletimi kullanıldığında IBM MQ ' e hangi kullanıcı tanıtıcılarının aktarılabileceği özetlenmektedir:

Çizelge 71. BINDINGS kipi

Yapılandırma	**Uygulama çağrıları `ConnectionFactory.createConnection()`**	**Uygulama çağrıları `ConnectionFactory.createConnection(String username, String password)`**
Uygulamanın konuşlandırma tanımlayıcısı bağlantı üreticisi için bir Kaynak Başvurusu içermiyor	Uygulama sunucusu işlemine ilişkin kullanıcı kimliği IBM MQadresine akıtılır.	`ConnectionFactory.createConnection(String username, String password)` yöntemine geçirilen kullanıcı kimliği ve parola IBM MQ' e aktarılır.
Uygulamanın konuşlandırma tanımlayıcısı bağlantı üreticisi için bir Kaynak Başvurusu içeriyor ve **`res-auth`** özelliği "Uygulama" olarak ayarlandı	Uygulama sunucusu işlemine ilişkin kullanıcı kimliği IBM MQadresine akıtılır.	`ConnectionFactory.createConnection(String username, String password)` yöntemine geçirilen kullanıcı kimliği ve parola IBM MQ' e aktarılır.

Çizelge 71. BINDINGS kipi (devamı var)

Yapılandırma	Uygulama çağrıları `ConnectionFactory.createConnection()`	Uygulama çağrıları `ConnectionFactory.createConnection(String username, String password)`
Uygulamanın konuşlandırma tanımlayıcısı bağlantı üreticisi için bir Kaynak Başvurusu içeriyor ve **res-auth** özelliği "Taşıyıcı" olarak ayarlandı	Bağlantı üreticisine ilişkin kimlik doğrulama diğer adında belirtilen kullanıcı kimliği ve parola IBM MQ' e akıtılır.	Bağlantı üreticisine ilişkin kimlik doğrulama diğer adında belirtilen kullanıcı kimliği ve parola IBM MQ' e akıtılır.
Uygulamanın konuşlandırma tanımlayıcısı, **res-auth** özelliği "Taşıyıcı" olarak ayarlanmış bağlantı üreticisine ilişkin bir Kaynak Başvurusu içeriyor ve uygulama bir kimlik doğrulama diğer adıyla yapılandırıldı	Uygulamanın kullanmak üzere yapılandırıldığı kimlik doğrulama diğer adında belirtilen kullanıcı kimliği ve parola IBM MQ' e akıtılır.	Uygulamanın kullanmak üzere yapılandırıldığı kimlik doğrulama diğer adında belirtilen kullanıcı kimliği ve parola IBM MQ' e akıtılır.

Çizelge 72. CLIENT kipi

Yapılandırma	Uygulama çağrıları `ConnectionFactory.createConnection()`	Uygulama çağrıları `ConnectionFactory.createConnection(String username, String password)`
Uygulamanın konuşlandırma tanımlayıcısı, bağlantı üreticisi için bir Kaynak Başvurusu içermiyor ve bağlantı üreticisi, MCAUSER özelliği ayarlanmamış bir IBM MQ kanalını kullanacak şekilde yapılandırıldı	Uygulama sunucusu işlemine ilişkin kullanıcı kimliği IBM MQadresine akıtılır.	`ConnectionFactory.createConnection(String username, String password)` yöntemine geçirilen kullanıcı kimliği ve parola IBM MQ' e aktarılır.
Uygulamanın konuşlandırma tanımlayıcısı, bağlantı üreticisi için bir Kaynak Başvurusu içermiyor ve bağlantı üreticisi, MCAUSER özelliği bir kullanıcı kimliğine ayarlanmış bir IBM MQ kanalını kullanacak şekilde yapılandırıldı	Bağlantı üreticisinin kullanmak üzere yapılandırıldığı IBM MQ kanalında MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQolarak aşağıya doğru akıtılır.	Bağlantı üreticisinin kullanmak üzere yapılandırıldığı IBM MQ kanalında MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQolarak aşağıya doğru akıtılır.
Uygulamanın konuşlandırma tanımlayıcısı, **res-auth** özelliği *Uygulama* olarak ayarlanmış bağlantı üreticisine ilişkin bir Kaynak Başvurusu içerir ve bağlantı üreticisi, MCAUSER özelliği ayarlanmamış bir IBM MQ kanalını kullanacak şekilde yapılandırılır.	Uygulama sunucusu işlemine ilişkin kullanıcı kimliği IBM MQadresine akıtılır.	`ConnectionFactory.createConnection(String username, String password)` yöntemine geçirilen kullanıcı kimliği ve parola IBM MQ' e aktarılır.

Çizelge 72. CLIENT kipi (devamı var)

Yapılandırma	Uygulama çağrıları **ConnectionFactory.createConnection()**	Uygulama çağrıları **ConnectionFactory.createConnection(String username, String password)**
Uygulamanın konuşlandırma tanımlayıcısı, **res-auth** özelliği *Uygulama* olarak ayarlanmış bağlantı üreticisine ilişkin bir Kaynak Başvurusu içerir ve bağlantı üreticisi, MCAUSER özelliği bir kullanıcı kimliğine ayarlanmış bir IBM MQ kanalını kullanacak şekilde yapılandırılır.	Bağlantı üreticisinin kullanmak üzere yapılandırıldığı IBM MQ kanalında MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' e akıtılır.	Bağlantı üreticisinin kullanmak üzere yapılandırıldığı IBM MQ kanalında MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' e akıtılır.
Uygulamanın konuşlandırma tanımlayıcısı, **res-auth** özelliği "*Taşıyıcı* olarak ayarlanmış bağlantı üreticisine ilişkin bir Kaynak Başvurusu içeriyor ve bağlantı üreticisi, MCAUSER özelliği ayarlanmamış bir IBM MQ kanalını kullanacak şekilde yapılandırıldı.	Bağlantı üreticisine ilişkin kimlik doğrulama diğer adında belirtilen kullanıcı kimliği ve parola IBM MQ' e akıtılır.	Bağlantı üreticisine ilişkin kimlik doğrulama diğer adında belirtilen kullanıcı kimliği ve parola IBM MQ' e akıtılır.
Uygulamanın konuşlandırma tanımlayıcısı, **res-auth** özelliği "*Taşıyıcı* olarak ayarlanmış bağlantı üreticisine ilişkin bir Kaynak Başvurusu içerir ve bağlantı üreticisi, MCAUSER özelliği bir kullanıcı kimliğine ayarlanmış bir IBM MQ kanalını kullanacak şekilde yapılandırılır.	Bağlantı üreticisinin kullanmak üzere yapılandırıldığı IBM MQ kanalında MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' e akıtılır.	Bağlantı üreticisinin kullanmak üzere yapılandırıldığı IBM MQ kanalında MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' e akıtılır.
Uygulamanın konuşlandırma tanımlayıcısı, **res-auth** özelliği "*Taşıyıcı* olarak ayarlanmış bağlantı üreticisine ilişkin bir Kaynak Başvurusu içeriyor ve uygulama bir kimlik doğrulama diğer adıyla yapılandırıldı ve bağlantı üreticisi, MCAUSER özelliği ayarlanmamış bir IBM MQ kanalını kullanacak şekilde yapılandırıldı.	Uygulamanın kullanmak üzere yapılandırıldığı kimlik doğrulama diğer adında belirtilen kullanıcı kimliği ve parola IBM MQ' e akıtılır.	Uygulamanın kullanmak üzere yapılandırıldığı kimlik doğrulama diğer adında belirtilen kullanıcı kimliği ve parola IBM MQ' e akıtılır.

Çizelge 72. CLIENT kipi (devamı var)		
Yapılandırma	**Uygulama çağrıları `ConnectionFactory.createConnection()`**	**Uygulama çağrıları `ConnectionFactory.createConnection(String username, String password)`**
Uygulamanın konuşlandırma tanımlayıcısı, **`res-auth`** özelliği *Taşıyıcı* olarak ayarlanmış bağlantı üreticisine ilişkin bir Kaynak Başvurusu içeriyor ve uygulama bir kimlik doğrulama diğer adıyla yapılandırıldı ve bağlantı üreticisi, MCAUSER kullanıcı kimliğine ayarlanmış bir IBM MQ kanalını kullanacak şekilde yapılandırıldı.	Bağlantı üreticisinin kullanmak üzere yapılandırıldığı IBM MQ kanalında MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' e akıtılır.	Bağlantı üreticisinin kullanmak üzere yapılandırıldığı IBM MQ kanalında MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' e akıtılır.

Doğrudan arama yoluyla bağlantı üreticisinin kullanılması

Bir IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticisi tanımlandıktan sonra, bir kurum uygulaması bağlantı üreticisi tanımını arayabilir ve IBM MQ kuyruk yöneticisine JMS bağlantısı yaratmak için bunu kullanabilir. Bu, doğrudan bir bakışla yapılabilir.

Doğrudan arama kullanmak için bir kurum uygulaması, aşağıdaki yöntem çağrısıyla uygulama sunucusunun JNDI havuzuna bağlanır:

```
InitialContext ctx = new InitialContext();
```

JNDI havuzuna bağlandıktan sonra, kurum uygulaması bağlantı üreticisinin JNDI adını kullanarak bağlantı üreticisi tanımlamasını aşağıdaki gibi tanımlar:

```
ConnectionFactory cf = (ConnectionFactory) ctx.lookup("jms/myCF");
```

Notlar:

- Kurum uygulaması geliştirilirken, uygulama geliştiricinizin gerekli bağlantı üreticisinin JNDI adını bilmesi gerekir. JNDI adı uygulamanın içinde sabit olarak kodlandığından, JNDI adı değişirse, uygulamayı yeniden yazmanız ve yeniden konuşlandırmanız gerekir.
- Bu şekilde bir bağlantı üreticisi tanımlaması kullanıldığında, kimlik doğrulama diğer adında belirtilen kullanıcı adı ve parola (bağlantı üreticisinin kullanmak üzere yapılandırıldığı) IBM MQ' e geçmez. Bu, yetkisiz uygulamaların bağlantı üreticisini tanımasını ve güvenli IBM MQ sistemlerine bağlanmak için bunu kullanabilmesini önlemektir.

IBM MQ ' e akan kullanıcı adı ve parola, bağlantı üreticisinden JMS bağlantısı oluşturmak için kullanılan yönteme bağlıdır.

Bir uygulama bu yöntemi kullanarak bir JMS bağlantısı oluşturursa:

```
ConnectionFactory.createConnection()
```

Varsayılan kullanıcı kimliği IBM MQ' e aktarılır. Bu, kurum uygulamasının çalıştığı uygulama sunucusunu başlatan kullanıcı adı ve paroladır.

Diğer bir seçenek olarak, bir uygulama yöntemi çağırarak JMS bağlantısı yaratabilir:

```
ConnectionFactory.createConnection(String username, String password)
```

Bir uygulama bir bağlantı üreticisini doğrudan aradıktan sonra bu yöntemi çağırdıysa, `createConnection()` yöntemine geçirilen kullanıcı adı ve parola IBM MQ' e aktarılır.

Önemli: IBM MQ 8.0öncesinde IBM MQ , bir yetki denetimini işledi; yalnızca, aşağı akıtılan kullanıcı adının kuyruk yöneticisine erişim yetkisi olduğundan emin olmak için.

Parolada herhangi bir denetim yapılmadı. Kimlik doğrulama denetimi gerçekleştirmek ve kullanıcı adı ve parolanın geçerli olduğunu doğrulamak için bir IBM MQ kanal güvenlik çıkışının yazılması gerekir. Bunun nasıl yapılacağına ilişkin ayrıntılar için bkz. “Kanal güvenliği çıkış programları” sayfa 928.

Kuyruk yöneticisi, IBM MQ 8.0' den kullanıcı adının yanı sıra parolayı da denetler.

Dolaylı arama yoluyla bağlantı üreticisinin kullanılması

When you are writing an enterprise application, if the JNDI name of the connection factory is unknown, or if the application is to be installed onto different application servers using a different connection factory, with a different JNDI name (depending on what application server it is installed onto), then the connection factory can be looked up using a resource reference. Bu, dolaylı arama yoluyla yapılabilir.

Örnek

Bir kurum uygulaması, `jms/myCF`kullanarak doğrudan bağlantı üreticisini aramak yerine, yerel JNDI adına sahip bir kaynak başvurusu içerir: `jms/myResourceReferenceCF`.

Bu JNDI adını kullanmak için uygulama, uygulama sunucusunun JNDI havuzuna, uygulamanın doğrudan bir aramada bulunmasıyla aynı şekilde bağlanır:

```
InitialContext ctx = new InitialContext();
```

Uygulama, `jms/myCF` öğesini doğrudan tanımlamak yerine, kaynak başvurusunun JNDI adını tanımlar:

```
ConnectionFactory cf = (ConnectionFactory) ctx.lookup("java:comp/env/jms/
myResourceReferenceCF");
```

Uygulama sunucusuna kurum uygulamasının dolaylı bir aramanın gerçekleştirildiğini söylemek için `java:comp/env` yerel JNDI adı öneki gerekir.

Uygulama konuşlandırıldığında, kullanıcı JNDI kaynak başvurusunun adını `jms/myResourceReferenceCF` uygulamanın önceden yaratmış olduğu bağlantı üreticisinin JNDI adıyla eşler: `jms/myCF`.

Uygulama çalıştırıldığında, uygulama sunucusunun eşlediği yerel JNDI adını kullanarak bir JMS bağlantı üreticisi arar: `jms/myCF`. Daha sonra bu bağlantı üreticisi, uygulama tarafından IBM MQile bağlantı oluşturmak için kullanılır.

Kimlik doğrulama diğer adları ve dolaylı aramaları

Kaynak başvurusu, sağlanan bağlantı üreticisinin davranışını değiştiren ek özelliklerin tanımlanmasına da olanak sağlar. Bir kaynak başvurusunun özelliklerinden biri **`res-auth`**' dir. Bu özelliğin değeri, kurum uygulamasının IBM MQ ile bağlantı yaratırken (bir kimlik doğrulama diğer adı tanımlandıysa) kaynak başvurusunun eşlendiği bağlantı üreticisinin kimlik doğrulama diğer adını mı kullanması gerektiğini, yoksa uygulamanın kendi kullanıcı adını ve parolasını belirtip belirtmediğini belirtir.

Bu özelliğin varsayılan değeri *Uygulama*' dır. Bu, bir JMS bağlantısı yaratıldığında kuyruk yöneticisine akan kullanıcı adı ve parolanın uygulamanın kendisi tarafından belirlendiği anlamına gelir. Kaynak başvurusunun eşlendiği bağlantı üreticisinin kimlik doğrulama diğer adı kullanılmaz.

Uygulamalar aşağıdaki yöntemlerden birini kullanarak JMS bağlantıları yaratabilir:

- `ConnectionFactory.createConnection()`
- `ConnectionFactory.createConnection(String username, String password)`

Bir uygulama `ConnectionFactory.createConnection()` kullanıyorsa ve **res-auth** , *Uygulama*olarak ayarlanmışsa, varsayılan kullanıcı kimliği IBM MQ' e doğru akacaktır. Bu, kurum uygulamasının çalıştığı uygulama sunucusunu başlatan kullanıcı adı ve paroladır.

Bir uygulama `ConnectionFactory.createConnection(String username, String password)` kullanıyorsa ve **res-auth** , *Uygulama*olarak ayarlanmışsa, yönteme geçirilen kullanıcı adı ve parola IBM MQadresine gönderilir.

Kaynak başvurusunun bağlantı yaratırken eşlediği bağlantı üreticisinde tanımlanan kimlik doğrulama diğer adını kullanmak için **res-auth** özelliğini *Taşıyıcı*değerine ayarlamanız gerekir. Bir uygulama JMS bağlantısı oluşturduğunda, `createConnection` çağrısı bir kullanıcı adı ve parola belirtse de kimlik doğrulama diğer adı ayrıntıları kullanılır.

Dolaylı arama kullanılırken kimlik doğrulama diğer adının geçersiz kılınması

Bir uygulama **res-auth** özelliği *Taşıyıcı*olarak ayarlanmış bir kaynak başvurusu kullanıyorsa, JMS bağlantıları yaratıldığında kullanılan kimlik doğrulama diğer adını geçersiz kılabilirsiniz.

Kimlik doğrulama diğer adını geçersiz kılmak için kaynak başvurusunun, uygulamanın konuşlandırılacağı uygulama sunucusu ortamında önceden yaratılmış olan bir kimlik doğrulama diğer adıyla eşlenen **authDataAlias**adlı ek bir özellik içermesi gerekir. Bu özelliği, IBMtarafından sağlanan Rational araçları kullanılarak yaratılan kaynak başvurularında belirtebilirsiniz.

Bu yöntemi kullanarak, dolaylı olarak aranmış bir JMS bağlantı üreticisini kullanırken farklı bir kimlik doğrulama diğer adı kullanabilirsiniz. Belirtilen kimlik doğrulama diğer adı yoksa, kurum uygulaması kurulduktan sonra yeni bir kimlik doğrulama diğer adı belirtilebilir. Daha fazla bilgi için WebSphere Application Server ürün belgelerinde *Kaynak başvuruları* konusuna bakın.

WebSphere Application Server 8.5.5 ile ilgili bilgiler

Kaynak başvuruları

WebSphere Application Server 8.0 ile ilgili bilgiler

Kaynak başvuruları

WebSphere Application Server 7.0 ile ilgili bilgiler

Kaynak başvuruları

WebSphere Application Server kümelerini kullanırken iletiyle yönlendirilen Bean 'ler için iş yükü dengelemesi

Bir WebSphere Application Server 7.0 ve WebSphere Application Server 8.0 kümesinde konuşlandırılan ve IBM MQ ileti alışverişi sağlayıcısı normal kipinde çalışacak şekilde yapılandırılan ileti odaklı bean uygulamalarını kullanırken, küme üyelerinden biri iletilerin çoğunu işler. İletilerin işlenmesini birden çok küme üyesine dağıtmak için küme üyelerinin iş yükünü dengeleyebilirsiniz.

IBM MQ , uygulamaların **MQCB** ve **MQCTL**adlı API ' leri kullanarak kuyruktaki iletileri zamanuyumsuz olarak kullanmasını sağlayan **Asynchronous consume**adlı bir özellik içerir.

WebSphere Application Server 7.0 ve WebSphere Application Server 8.0içinde çalışan, IBM MQ ileti alışverişi sağlayıcısı normal kipini kullanan, ileti odaklı bean uygulamaları bu özelliği otomatik olarak kullanır. Uygulamalar başladığında, **MQCB**çağrılarak izlenecek şekilde yapılandırıldıkları JMS hedefinde zamanuyumsuz bir tüketici ayarlar. Daha sonra, uygulamanın JMS hedefinden ileti almaya hazır olduğunu belirtmek için **MQCTL** API çağrılır.

İletiyle yönlendirilen Bean uygulamaları bir WebSphere Application Server kümesine konuşlandırıldığında, her küme üyesi, iletiyle yönlendirilen Bean 'in iletileri izlediği JMS hedefi için zamanuyumsuz bir tüketici ayarlar. JMS hedefini barındıran IBM MQ kuyruk yöneticisi, JMS hedefinde işlenecek uygun bir ileti olduğunda küme üyesine bildirimde bulunulmasından sorumludur.

WebSphere Application Server bir IBM MQ kuyruk yöneticisine bağlanırken, JMS hedefine ulaşan iletiler, o JMS hedefine kaydedilen zamanuyumsuz tüketicilerin tümüne daha eşit olarak dağıtılır. Bir WebSphere Application Server 7.0 ve WebSphere Application Server 8.0 kümesi içinde konuşlandırılan ileti odaklı Bean uygulamaları için bu, iletilerin küme üyeleri arasında daha eşit olarak dağıtılacağı anlamına gelir.

İlgili görevler
JMS **PROVIDERVERSION** özelliğinin yapılandırılması

IBM MQ Headers paketinin kullanılması

IBM MQ Headers paketi, bir iletinin IBM MQ üstbilgilerini değiştirmek için kullanabileceğiniz bir yardımcı arabirim ve sınıf kümesi sağlar. Genellikle, komut sunucusunu kullanarak (PCF (Programlanır Komut Biçimi) iletilerini kullanarak) denetim hizmetleri gerçekleştirmek istediğiniz için IBM MQ Headers paketini kullanırsınız.

Bu görev hakkında

IBM MQ Headers paketi, `com.ibm.mq.headers` ve `com.ibm.mq.headers.pcf` paketlerinde bulunur. Bu olanağı, IBM MQ ' in Java uygulamalarında kullanılmak üzere sağladığı iki alternatif API için de kullanabilirsiniz:

- IBM MQ classes for Java (IBM MQ Temel Javaolarak da adlandırılır).
- IBM MQ classes for Java Message Service (IBM MQ classes for JMS, IBM MQ JMSolarak da adlandırılır).

IBM MQ Temel Java uygulamalar genellikle MQMessage nesnelerini kullanır ve Üstbilgiler destek sınıfları IBM MQ Temel Java arabirimlerini yerel olarak anladıkları için bu nesnelerle doğrudan etkileşimde bulunabilir.

IBM MQ JMSiçinde, bir iletiye ilişkin bilgi yükü genellikle DataInput ve DataOutput akışlarıyla işlenebilecek bir String ya da byte dizisi nesnesidir. IBM MQ Üstbilgiler paketi, bu veri akışlarıyla etkileşim kurmak için kullanılabilir ve IBM MQ JMS uygulamaları tarafından gönderilen ve alınan MQ iletilerini işlemek için uygundur.

Bu nedenle, IBM MQ Üstbilgiler paketi IBM MQ Temel Java paketine başvurular içerse de, bu paket IBM MQ JMS uygulamalarında da kullanılmak üzere tasarlanmıştır ve Java Platform, Enterprise Edition (Java EE) ortamlarında kullanıma uygundur.

IBM MQ Headers paketini kullanmanın tipik bir yolu, denetim iletilerini Programlanabilir Komut Biçimi 'nde (PCF) değiştirmektir; örneğin, aşağıdaki nedenlerden biri olabilir:

- Bir IBM MQ kaynağına ilişkin ayrıntılara erişmek için.
- Kuyruğun derinliğini izlemek için.
- Bir kuyruğa erişimi engellemek için.

IBM MQ JMS API ile PCF iletileri kullanılarak, uygulama odaklı kaynakların bu tür yönetimi, IBM MQ Temel Java API ' yı kullanmak zorunda kalmadan Java EE uygulamalarından gerçekleştirilebilir.

Yordam

- IBM MQ classes for Javaile ilgili ileti üstbilgilerini değiştirmek üzere IBM MQ Üstbilgiler paketini kullanmak için bkz. “IBM MQ classes for Java ile kullanma” sayfa 495.
- IBM MQ classes for JMSile ilgili ileti üstbilgilerini değiştirmek üzere IBM MQ Üstbilgiler paketini kullanmak için bkz. “IBM MQ classes for JMS ile kullanma” sayfa 496.

IBM MQ classes for Java ile kullanma

IBM MQ classes for Java uygulamaları genellikle MQMessage nesnelerini kullanır ve Üstbilgiler destek sınıfları, IBM MQ classes for Java arabirimlerini yerel olarak anladıkları için bu nesnelerle doğrudan etkileşimde bulunabilir.

Bu görev hakkında

IBM MQ , IBM MQ Headers paketinin IBM MQ Base Java API (IBM MQ classes for Java) ile nasıl kullanılacağını gösteren bazı örnek uygulamalar sağlar.

Örneklerde iki şey var:

- Bir denetim işlemi gerçekleştirmek ve yanıt iletisini ayrıştırmak için bir PCF iletisi yaratılması.
- Bu PCF iletisinin IBM MQ classes for Javakullanılarak gönderilmesi.

Hangi altyapıyı kullandığınıza bağlı olarak, bu örnekler IBM MQ kuruluşunuzun `samples` ya da `tools` dizinindeki `pcf` dizini altına kurulur (bkz. “IBM MQ classes for Java için kuruluş dizinleri” sayfa 342).

Yordam

1. Bir denetim işlemi gerçekleştirmek ve yanıt iletisini ayrıştırmak için bir PCF iletisi yaratın.
2. Bu PCF iletisini IBM MQ classes for Javakomutunu kullanarak gönderin.

İlgili kavramlar

“IBM MQ ileti üstbilgilerinin IBM MQ classes for Java ile işlenmesi” sayfa 368
Farklı ileti üstbilgisi tiplerini gösteren Java sınıfları sağlanır. İki yardımcı sınıf da sağlanır.

“IBM MQ classes for Java ile PCF iletilerinin işlenmesi” sayfa 373
Java sınıfları, PCF yapılandırılmış iletileri yaratmak ve ayrıştırmak, PCF isteklerinin gönderilmesini ve PCF yanıtlarının toplanmasını kolaylaştırmak için sağlanır.

IBM MQ classes for JMS ile kullanma

IBM MQ Üstbilgileri IBM MQ classes for JMSile kullanmak için, IBM MQ classes for Javaile aynı temel adımları gerçekleştirmeniz gerekir. IBM MQ Headers paketi ve IBM MQ classes for Javaile aynı örnek kod kullanılarak PCF iletisi oluşturulabilir ve yanıt tam olarak aynı şekilde ayrıştırılabilir.

Bu görev hakkında

IBM MQ API kullanılarak bir PCF iletisi göndermek için, ileti bilgi yükü bir JMS Bytes Message içine yazılmalı ve standart JMS API ' leri kullanılarak gönderilmelidir. Dikkate alınması gereken tek şey, iletinin bir JMS RFH2 ya da MQMD ' de belirli değerleri olan başka bir üstbilgi içermemesi gerekdiğidir.

Bir PCF iletisi göndermek için aşağıdaki adımları tamamlayın. PCF iletisinin yaratılma ve yanıt iletisinden bilgi ayıklma şekli IBM MQ classes for Java iletisiyle aynıdır (bkz. “IBM MQ classes for Java ile kullanma” sayfa 495).

Yordam

1. SYSTEM.ADMIN.COMMAND.QUEUEkuyruğunu gösteren bir JMS Kuyruk Hedefi yaratın.

   IBM MQ JMS uygulamaları, PCF iletilerini SYSTEM.ADMIN.COMMAND.QUEUEve bu kuyruğu temsil eden bir JMS Hedef nesnesine erişilmesi gerekir. Hedef için aşağıdaki özellikler ayarlanmış olmalıdır:

   ```
   WMQ_MQMD_WRITE_ENABLED = YES
   WMQ_MESSAGE_BODY = MQ
   ```

   WebSphere Application Serverkullanıyorsanız, bu özellikleri Hedef 'de özel özellikler olarak tanımlamanız gerekir.

   Hedefi programlı olarak bir uygulama içinden yaratmak için aşağıdaki kodu kullanın:

   ```
   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. Bir PCF iletisini, doğru MQMD değerlerini içeren bir JMS Byte iletisine dönüştürün.

   Bir JMS Bytes iletisinin yaratılması ve üzerine PCF iletisinin yazılması gerekir. Bir yanıt kuyruğunun yaratılması gerekiyor, ancak bunun belirli bir ayarı olmaması gerekiyor.

   Aşağıdaki örnek kod parçacığı, JMS Bytes Message 'ın nasıl yaratılacağını ve buna bir com.ibm.mq.headers,pcf.PCFMessage nesnesinin nasıl yazılacağını gösterir. PCFMessage nesnesi

(pcfCmd) daha önce IBM MQ Headers paketi kullanılarak oluşturulmuştur. (PCFMessage yükleme paketinin com.ibm.mq.headers.pcf.PCFMessageolduğunu unutmayın).

```
 // 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. İletiyi gönderin ve standart JMS API ' lerini kullanarak yanıtı alın.
4. Yanıt iletisini işlenmek üzere bir PCF iletisine dönüştürün.

   Yanıt iletisini almak ve PCF iletisi olarak işlemek için aşağıdaki kodu kullanın:

```
 // 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);
```

İlgili kavramlar

“JMS ileti” sayfa 138

JMS iletileri, bir üstbilgiden, özelliklerden ve bir gövdeden oluşur. JMS , beş tip ileti gövdesini tanımlar.

IBM i IBM MQ ' ın Java ve JMS ile IBM i üzerinde ayarlanması

Bu konu derlemi, CL komutlarını ya da qshell ortamını kullanarak IBM i üzerinde IBM MQ with Java ve JMS ürününü nasıl kuracağınıza ve sınacağınıza ilişkin bir genel bakış sağlar.

Not:

- IBM MQ 8.0, `ldap.jar`, `jndi.jar` ve `jta.jar` 'den JDK' nın bir parçasıdır.
- V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0' dan Jakarta Messaging 3.0 , yeni uygulamalar geliştirmek için desteklenir. IBM MQ 9.3.0 , var olan uygulamalar için JMS 2.0 ' e destek vermeye devam eder. Aynı uygulamada hem Jakarta Messaging 3.0 API hem de JMS 2.0 API ' nin kullanılması desteklenmez. Daha fazla bilgi için, bkz. JMS/Jakarta İleti Sistemi için IBM MQ sınıflarının kullanılması.

CL komutlarının kullanılması

Ayarladığınız CLASSPATH, JNDI olmadan MQ tabanı Java, JMS ve JMS ile sınanmak içindir.

/home/Userprofile dizininiz altında bir .profile dosyası kullanmıyorsanız, aşağıdaki sistem düzeyi ortam değişkenlerini ayarlamanız gerekir. **WRKENVVAR** komutunu kullanarak bunların ayarlı olup olmadığını denetleyebilirsiniz.

1. Tüm sisteme ilişkin ortam değişkenlerini görüntülemek için şu komutu verin: **WRKENVVAR LEVEL (*SYS)**
2. İşinize özgü ortam değişkenlerini görüntülemek için şu komutu verin: **WRKENVVAR LEVEL (*JOB)**
3. CLASSPATH ayarlanmamışsa şu komutu verin:

JM 3.0

```
ADDENVVAR ENVVAR(CLASSPATH)
       VALUE('.:/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar
       :/QIBM/ProdData/mqm/java/lib/connector.jar:/QIBM/ProdData/mqm/java/lib
       :/QIBM/ProdData/mqm/java/samples/base
       :/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jakarta.client.jar
       :/QIBM/ProdData/mqm/java/lib/jms.jar
       :/QIBM/ProdData/mqm/java/lib/providerutil.jar
       :/QIBM/ProdData/mqm/java/lib/fscontext.jar:') LEVEL(*SYS)
```

JMS 2.0

```
ADDENVVAR ENVVAR(CLASSPATH)
       VALUE('.:/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar
       :/QIBM/ProdData/mqm/java/lib/connector.jar:/QIBM/ProdData/mqm/java/lib
       :/QIBM/ProdData/mqm/java/samples/base
       :/QIBM/ProdData/mqm/java/lib/com.ibm.mq.allclient.jar
       :/QIBM/ProdData/mqm/java/lib/jms.jar
       :/QIBM/ProdData/mqm/java/lib/providerutil.jar
       :/QIBM/ProdData/mqm/java/lib/fscontext.jar:') LEVEL(*SYS)
```

4. QIBM_MULTI_THREADED ayarlanmamışsa şu komutu verin:

```
ADDENVVAR ENVVAR(QIBM_MULTI_THREADED) VALUE('Y') LEVEL(*SYS)
```

5. QIBM_USE_DESCRIPTOR_STDIO ayarlanmazsa, şu komutu verin:

```
ADDENVVAR ENVVAR(QIBM_USE_DESCRIPTOR_STDIO) VALUE('I') LEVEL(*SYS)
```

6. QSH_REDIRECTION_TEXTDATA ayarlanmamışsa şu komutu verin:

```
ADDENVVAR ENVVAR(QSH_REDIRECTION_TEXTDATA) VALUE('Y') LEVEL(*SYS)
```

qshell ortamını kullanma

QSHELL ortamını kullanıyorsanız, /home/Userprofile dizininizde bir .profile ayarlayabilirsiniz. Daha fazla bilgi için Qshell Interpreter (qsh) belgelerine bakın.

.profileiçinde aşağıdakileri belirtin. CLASSPATH deyiminin tek bir satırda olması gerektiğini ya da \ karakteri kullanılarak farklı satırlara ayrılması gerektiğini unutmayın.

JM 3.0

```
CLASSPATH=.:/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar: \
/QIBM/ProdData/mqm/java/lib/connector.jar: \
/QIBM/ProdData/mqm/java/lib: \
/QIBM/ProdData/mqm/java/samples/base: \
/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jakarta.client.jar: \
/QIBM/ProdData/mqm/java/lib/jms.jar: \
/QIBM/ProdData/mqm/java/lib/providerutil.jar: \
/QIBM/ProdData/mqm/java/lib/fscontext.jar:
HOME=/home/XXXXX
LOGNAME=XXXXX
PATH=/usr/bin:
QIBM_MULTI_THREADED=Y QIBM_USE_DESCRIPTOR_STDIO=I
QSH_REDIRECTION_TEXTDATA=Y
TERMINAL_TYPE=5250
```

JMS 2.0

```
CLASSPATH=.:/QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar: \
/QIBM/ProdData/mqm/java/lib/connector.jar: \
/QIBM/ProdData/mqm/java/lib: \
/QIBM/ProdData/mqm/java/samples/base: \
/QIBM/ProdData/mqm/java/lib/com.ibm.mq.allclient.jar: \
/QIBM/ProdData/mqm/java/lib/jms.jar: \
/QIBM/ProdData/mqm/java/lib/providerutil.jar: \
/QIBM/ProdData/mqm/java/lib/fscontext.jar:
HOME=/home/XXXXX
LOGNAME=XXXXX
PATH=/usr/bin:
QIBM_MULTI_THREADED=Y QIBM_USE_DESCRIPTOR_STDIO=I
QSH_REDIRECTION_TEXTDATA=Y
TERMINAL_TYPE=5250
```

DSPLIBLkomutunu vererek, QMQMJAVA kitaplığının kitaplık listesinde bulunduğundan emin olun.

QMQMJAVA kitaplığı listede yoksa, şu komutu kullanarak ekleyin: **ADDLIBLE LIB (QMQMJAVA)**

IBM i Java ile IBM i üzerinde IBM MQ testi

MQIVP örnek programını kullanarak IBM MQ ' i Java ile test etme.

IBM MQ Temel Java test ediliyor

Aşağıdaki yordamı gerçekleştirin:

1. Aşağıdaki komutu vererek kuyruk yöneticisinin başlatıldığını ve kuyruk yöneticisinin durumunun ACTIVE olduğunu doğrulayın:

   ```
   WRKMQM MQMNAME(QMGRNAME)
   ```

2. JAVA.CHANNEL sunucu bağlantısı kanalı aşağıdaki komut verilerek yaratıldı:

   ```
   WRKMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN) MQMNAME(QMGRNAME)
   ```

   a. JAVA.CHANNEL yok, aşağıdaki komutu verin:

   ```
   CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN) MQMNAME(QMGRNAME)
   ```

3. **WRKMQMLSR** komutunu kullanarak, kuyruk yöneticisi dinleyicisinin 1414 kapısı ya da kullandığınız kapı için çalıştığını doğrulayın.

   a. Kuyruk yöneticisi için herhangi bir dinleyici başlatılmazsa, şu komutu verin:

   ```
   STRMQMLSR PORT(xxxx) MQMNAME(QMGRNAME)
   ```

MQIVP örnek sınama programının çalıştırılması

1. STRQSH komutunu girerek komut satırından qshell 'i başlatın.
2. **export** komutunu vererek doğru CLASSPATH değerinin ayarlandığını doğrulayın ve **cd** komutunu aşağıdaki gibi verin:

   ```
   cd /qibm/proddata/mqm/java/samples/wmqjava/samples
   ```

3. Aşağıdaki komutu girerek **java** programını çalıştırın:

   ```
   java MQIVP
   ```

Sizden istendiğinde ENTER tuşuna basabilirsiniz:

- Bağlantı tipi
- IP adresi
- Kuyruk yöneticisi adı

varsayılan değerleri kullanmak için. Bu, QMQMJAVA kitaplığında bulunabilecek ürün bağ tanımlarını doğrular.

Aşağıdaki örneğe benzer bir çıkış alırsınız. Telif hakkı bildiriminin, kullandığınız ürünün sürümüne bağlı olduğunu unutmayın.

```
> java MQIVP
MQSeries for Java Installation Verification Program
5724-H72 (C) Copyright IBM Corp. 2011, 2024. All Rights Reserved.
============================================================

Please enter the IP address of the MQ server :>
Please enter the queue manager name :>
Attaching Java program to QIBM/ProdData/mqm/java/lib/connector.JAR.
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...>
$
```

IBM MQ Java istemci bağlantısının sınanması

Aşağıdakileri belirtmeniz gerekir:

- Bağlantı tipi
- IP adresi
- Kapı
- Sunucu bağlantı kanalı
- Kuyruk yöneticisi

Aşağıdaki örneğe benzer bir çıkış alırsınız. Telif hakkı bildiriminin, kullandığınız ürünün sürümüne bağlı olduğunu unutmayın.

```
> java MQIVP
MQSeries for Java Installation Verification Program
5724-H72 (C) Copyright IBM Corp. 2011, 2024. All Rights Reserved.
============================================================

Please enter the IP address of the MQ server :> x.xx.xx.xx
Please enter the port to connect to : (1414)> 1470
Please enter the server connection channel name :> JAVA.CHANNEL
Please enter the queue manager name :> KAREN01
Success: Connected to queue manager.
Success: Opened SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Put a message to SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Got a message from SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Closed SYSTEM.DEFAULT.LOCAL.QUEUE
Success: Disconnected from queue manager


Tests complete -
SUCCESS: This MQ Transport is functioning correctly.
Press Enter to continue ...>
$
```

IBM i JMS ile IBM i üzerinde IBM MQ testi

JNDI ile ya da JNDI olmadan IBM MQ ' ni JMS ile nasıl test edeceğinizi

IVTRun örneğini kullanarak JNDI olmadan JMS sınanıyor

Aşağıdaki yordamı gerçekleştirin:

1. Aşağıdaki komutu vererek kuyruk yöneticisinin başlatıldığını ve kuyruk yöneticisinin durumunun ACTIVE olduğunu doğrulayın:

   ```
   WRKMQM MQMNAME(QMGRNAME)
   ```

2. **STRQSH** komutunu vererek komut satırından qshell öğesini başlatın.
3. Dizini aşağıdaki gibi değiştirmek için **cd** komutunu kullanın:

   ```
   cd /qibm/proddata/mqm/java/bin
   ```

4. Komut dosyasını çalıştır:

   ```
   IVTRun -nojndi [-m qmgrname]
   ```

Aşağıdaki örneğe benzer bir çıkış alırsınız. Telif hakkı bildirimlerinin, kullandığınız ürünlerin sürümlerine bağlı olduğunu unutmayın:

```
 IVTRun -nojndi -m ELCRTP19

Attaching Java program to
/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.JAR.
Attaching Java program to
/QIBM/ProdData/mqm/java/lib/jms.JAR.

5724-H72, 5724-B41, 5655-F10 (c) Copyright IBM Corp. 2011, 2024.
All Rights Reserved.
WebSphere MQ classes for Java Message Service 5.300
Installation Verification Test


Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
JMSType: null
JMSDeliveryMode: 2
JMSExpiration: 0
JMSPriority: 4
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f9404040403ccf041f0000c012
JMSTimestamp: 1020273404500
JMSCorrelationID:null
JMSDestination: queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
JMSReplyTo: null
JMSRedelivered: false
JMS_IBM_PutDate:20040326
JMSXAppID:QP0ZSPWT STANLEY 170302
JMS_IBM_Format:MQSTR
JMS_IBM_PutApplType:8
JMS_IBM_MsgType:8
JMSXUserID:STANLEY
JMS_IBM_PutTime:13441354
JMSXDeliveryCount:1
A simple text message from the MQJMSIVT program
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
```

```
$>
$
```

IBM MQ JMS istemci kipinin JNDI olmadan sınanması

Aşağıdaki yordamı gerçekleştirin:

1. Aşağıdaki komutu vererek kuyruk yöneticisinin başlatıldığını ve kuyruk yöneticisinin durumunun ACTIVE olduğunu doğrulayın:

   ```
   WRKMQM MQMNAME(QMGRNAME)
   ```

2. Aşağıdaki komutu vererek sunucu bağlantı kanalının oluşturulduğunu doğrulayın:

   ```
   WRKMQMCHL CHLNAME( SYSTEM.DEF.SVRCONN ) CHLTYPE(*SVRCN)
   MQMNAME(QMGRNAME)
   ```

3. **WRKMQMLSR** komutunu vererek, dinleyicinin doğru kapı için başlatıldığını doğrulayın.
4. **STRQSH** komutunu vererek komut satırından qshell öğesini başlatın.
5. **export** komutunu vererek CLASSPATH değişkeninin doğru olduğunu doğrulayın.
6. Dizini aşağıdaki gibi değiştirmek için **cd** komutunu kullanın:

   ```
   cd /qibm/proddata/mqm/java/bin
   ```

7. Komut dosyasını çalıştır:

   ```
   IVTRun -nojndi -client -m QMgrName -host hostname [-port port] [-channel channel]
   ```

Aşağıdaki örneğe benzer bir çıkış alırsınız. Telif hakkı bildirimlerinin, kullandığınız ürünlerin sürümlerine bağlı olduğunu unutmayın.

```
> IVTRun -nojndi -client -m ELCRTP19 -host ELCRTP19 -port 1414 -channel SYSTEM.DEF.SVRCONN

5724-H72, 5724-B41, 5655-F10 (c) Copyright IBM Corp. 2011, 2024.
All Rights Reserved.
WebSphere MQ classes for Java Message Service 5.300
Installation Verification Test


Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
JMSType: null
JMSDeliveryMode: 2
JMSExpiration: 0
JMSPriority: 4
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f9404040403ccf041f0000d012
JMSTimestamp: 1020274009970
JMSCorrelationID:null
JMSDestination: queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
JMSReplyTo: null
JMSRedelivered: false
JMS_IBM_PutDate:20040326
JMSXAppID:MQSeries Client for Java
JMS_IBM_Format:MQSTR
JMS_IBM_PutApplType:28
JMS_IBM_MsgType:8
JMSXUserID:QMQM
JMS_IBM_PutTime:14085237
JMSXDeliveryCount:1
```

```
A simple text message from the MQJMSIVT program
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
$
```

JNDI ile IBM MQ JMS sınanıyor

Aşağıdaki komutu vererek kuyruk yöneticisinin başlatıldığını ve kuyruk yöneticisinin durumunun ACTIVE olduğunu doğrulayın:

```
WRKMQM MQMNAME(QMGRNAME)
```

IVTRun örnek test komut dosyasının kullanılması

Aşağıdaki yordamı gerçekleştirin:

1. JMSAdmin.config dosyasında uygun değişiklikleri yapın. Bu dosyayı düzenlemek için IBM i komut satırından **EDTF** (Dosyayı Düzenle) komutunu kullanın

   ```
   EDTF '/qibm/proddata/mqm/java/bin/JMSAdmin.config'
   ```

   .

   a. Weblogic için LDAP ' ı kullanmak üzere şu yorumdan yorumu kaldırın:

      ```
      INITIAL_CONTEXT_FACTORY=com.sun.jndi.ldap.LdapCtxFactory
      ```

   b. WebSphere Application Serveriçin LDAP kullanmak üzere aşağıdaki açıklamayı kaldırın:

      ```
      INITIAL_CONTEXT_FACTORY=com.ibm.ejs.ns.jndi.CNInitialContextFactory
      ```

   c. Dosya sistemini sınamak için aşağıdaki açıklamayı kaldırın:

      ```
      INITIAL_CONTEXT_FACTORY=com.sun.jndi.fscontext.RefFSContextFactory
      ```

   d. Açıklamayı uygun satırdan kaldırarak doğru PROVIDER_URL adresini seçtiğinizden emin olun.

   e. # simgesini kullanarak diğer tüm satırları açıklama satırı yapın.

   f. Tüm değişikliklerinizi tamamladığınızda, **F2=Save** ve **F3=Exit**tuşlarına basın.

2. **STRQSH** komutunu vererek komut satırından qshell öğesini başlatın.
3. **export** komutunu vererek CLASSPATH değişkeninin doğru olduğunu doğrulayın.
4. Dizini aşağıdaki gibi değiştirmek için **cd** komutunu kullanın:

   ```
   cd /qibm/proddata/mqm/java/bin
   ```

5. **IVTSetup** komutunu vererek, yönetilen nesneleri (*MQQueueConnectionFactory* ve *MQQueue*) yaratmak için **IVTSetup** komut dosyasını başlatın.
6. Aşağıdaki komutu vererek IVTRun komut dosyasını çalıştırın:

   ```
   IVTRun -url providerURL [-icf initCtxFact]
   ```

Aşağıdaki örneğe benzer bir çıkış alırsınız. Telif hakkı bildirimlerinin, kullandığınız ürünlerin sürümlerine bağlı olduğunu unutmayın.

```
> IVTSetup
+ Creating script for object creation within JMSAdmin
+ Calling JMSAdmin in batch mode to create objects
Ignoring unknown flag: -i

5724-H72 (c) Copyright IBM Corp. 2011, 2024. All Rights Reserved.
Starting WebSphere MQ classes for Java Message Service Administration

InitCtx>
InitCtx>
InitCtx>
InitCtx>
InitCtx>
Stopping MQSeries classes for Java Message Service Administration

+ Administration done; tidying up files
+ Done!
$
> IVTRun -url file:////tmp/mqjms -icf com.sun.jndi.fscontext.RefFSContextFactory

5724-H72 (c) Copyright IBM Corp. 2011, 2024. All Rights Reserved.
MQSeries classes for Java Message Service
Installation Verification Test

Using administered objects, please ensure that these are available

Retrieving a QueueConnectionFactory from JNDI
Creating a Connection
Creating a Session
Retrieving a Queue from JNDI
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
JMSType: null
JMSDeliveryMode: 2
JMSExpiration: 0
JMSPriority: 4
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f9404040403ccf041f0000e012
JMSTimestamp: 1020274903770
JMSCorrelationID:null
JMSDestination: queue:///SYSTEM.DEFAULT.LOCAL.QUEUE
JMSReplyTo: null
JMSRedelivered: false
JMS_IBM_Format:MQSTR
JMS_IBM_PutApplType:8
JMSXDeliveryCount:1
JMS_IBM_MsgType:8
JMSXUserID:STANLEY
JMSXAppID:QP0ZSPWT STANLEY 170308
A simple text message from the MQJMSIVT program
Reply string equals original string
Closing QueueReceiver
Closing QueueSender
Closing Session
Closing Connection
IVT completed OK
IVT finished
$
```

Maven havuzu kullanılarak Java uygulama geliştirme

Bağımlılıkları otomatik olarak kurmak için Maven havuzunu kullanarak IBM MQiçin bir Java uygulaması geliştirirken, IBM MQ arabirimlerini kullanmadan önce hiçbir şeyi belirtik olarak kurmanız gerekmez.

Maven Merkezi Havuzu

Maven, uygulama oluşturmak için bir araçtır ve uygulamanızın erişmek isteyebileceği yapay nesneleri tutmak için bir havuz sağlar.

Maven Havuzu (ya da Merkezi Havuz), JAR dosyaları gibi dosyaların daha sonra bilinen bir adlandırma düzeneğiyle kolayca keşfedilen ayrı sürümlerine sahip olmalarını sağlayan bir yapıya sahiptir. Oluşturma

araçları, uygulamanızın bağımlılıklarını dinamik olarak çekmek için bu adları kullanabilir. Maven 'i bir oluşturma aracı olarak kullanırken POM dosyası olarak adlandırılan uygulamanızın tanımlamasında bağımlılıkları adlandırır ve oluşturma işlemi bundan sonra ne yapacağını bilir.

IBM MQ istemci dosyaları

IBM MQ Java istemci arabirimlerinin kopyaları `com.ibm.mq GroupId`altındaki Merkezi Havuzda bulunur. `com.ibm.mq.jakarta.client.jar` dosyasını (Jakarta Messaging 3.0) ve `com.ibm.mq.allclient.jar` dosyasını (JMS 2.0) bulabilirsiniz. Bu dosyalar genellikle bağımsız programlar için kullanılır. Ayrıca, Java EE uygulama sunucularında kullanılacak `wmq.jakarta.jmsra.rar` dosyasını (Jakarta Messaging 3.0) ve `wmq.jmsra.rar` dosyasını (JMS 2.0) da bulabilirsiniz. `jakarta.client.jar` ve `allclient.jar` , IBM MQ classes for JMS ve IBM MQ classes for Javaöğelerini içerir.

Önemli: IBM MQ relocatable JAR dosyasını içeren bir uygulama oluşturmak için Apache Maven Assembly Plugin *jar-with-bağımlılıkları* biçiminin kullanılması desteklenmez.

Maven komutu tarafından işlenen bir `pom.xml` dosyasında, aşağıdaki örneklerde gösterildiği gibi bu JAR dosyaları için bağımlılıklar eklersiniz:

- V 9.3.0 V 9.3.0 JM 3.0 Uygulama kodunuz ile `com.ibm.mq.jakarta.client.jar`arasındaki ilişkiyi göstermek için:

```
<dependency>
        <groupId>com.ibm.mq</groupId>
        <artifactId>com.ibm.mq.jakarta.client</artifactId>
        <version>9.3.0.0</version>
    </dependency>
```

- JMS 2.0 Uygulama kodunuz ile `com.ibm.mq.allclient.jar`arasındaki ilişkiyi göstermek için:

```
<dependency>
        <groupId>com.ibm.mq</groupId>
        <artifactId>com.ibm.mq.allclient</artifactId>
        <version>9.2.2.0</version>
    </dependency>
```

- V 9.3.0 V 9.3.0 JM 3.0 Jakarta EE kaynak bağdaştırıcısını kullanmak için:

```
<dependency>
     <groupId>com.ibm.mq</groupId>
     <artifactId>wmq.jakarta.jmsra</artifactId>
     <version>9.3.0.0</version>
 </dependency>
```

- JMS 2.0 JMS 2.0 Java EE kaynak bağdaştırıcısını kullanmak için:

```
<dependency>
     <groupId>com.ibm.mq</groupId>
     <artifactId>wmq.jmsra</artifactId>
     <version>9.2.2.0</version>
 </dependency>
```

Bir JMS projesini çalıştırmak için Eclipse içindeki basit bir proje örneği için, IBM Developer Developing Java applications for MQ with Mavenbaşlıklı makaleye bakın.

C++ uygulamaları geliştirilmesi

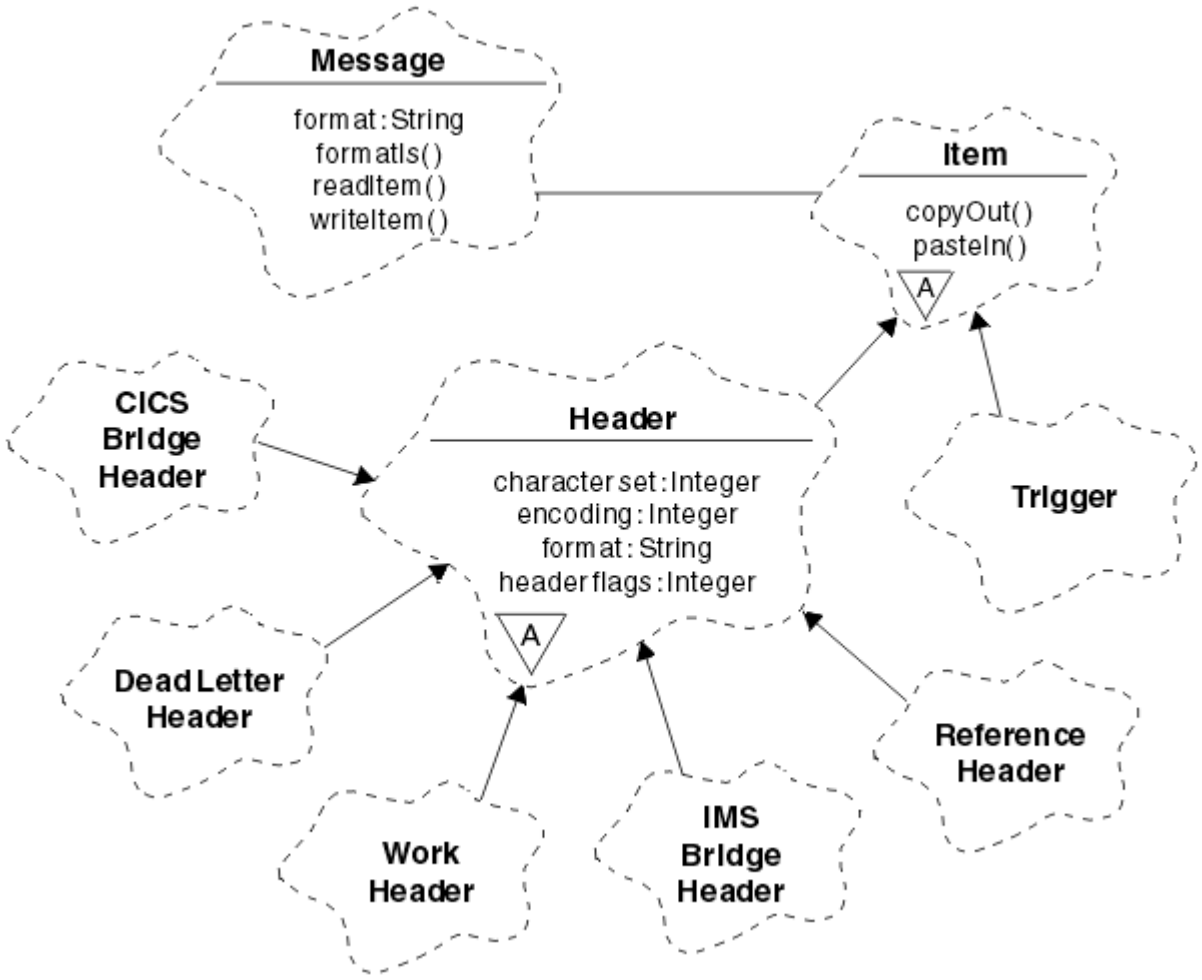
IBM MQ , IBM MQ nesnelerine eşdeğer C++ sınıflarını ve dizi veri tiplerine eşdeğer bazı ek sınıfları sağlar. MQI aracılığıyla kullanılamayan bazı özellikler sağlar.

IBM WebSphere MQ 7.0, IBM MQ programlama arabirimlerinde yapılan geliştirmeler C++ sınıflarına uygulanmaz.

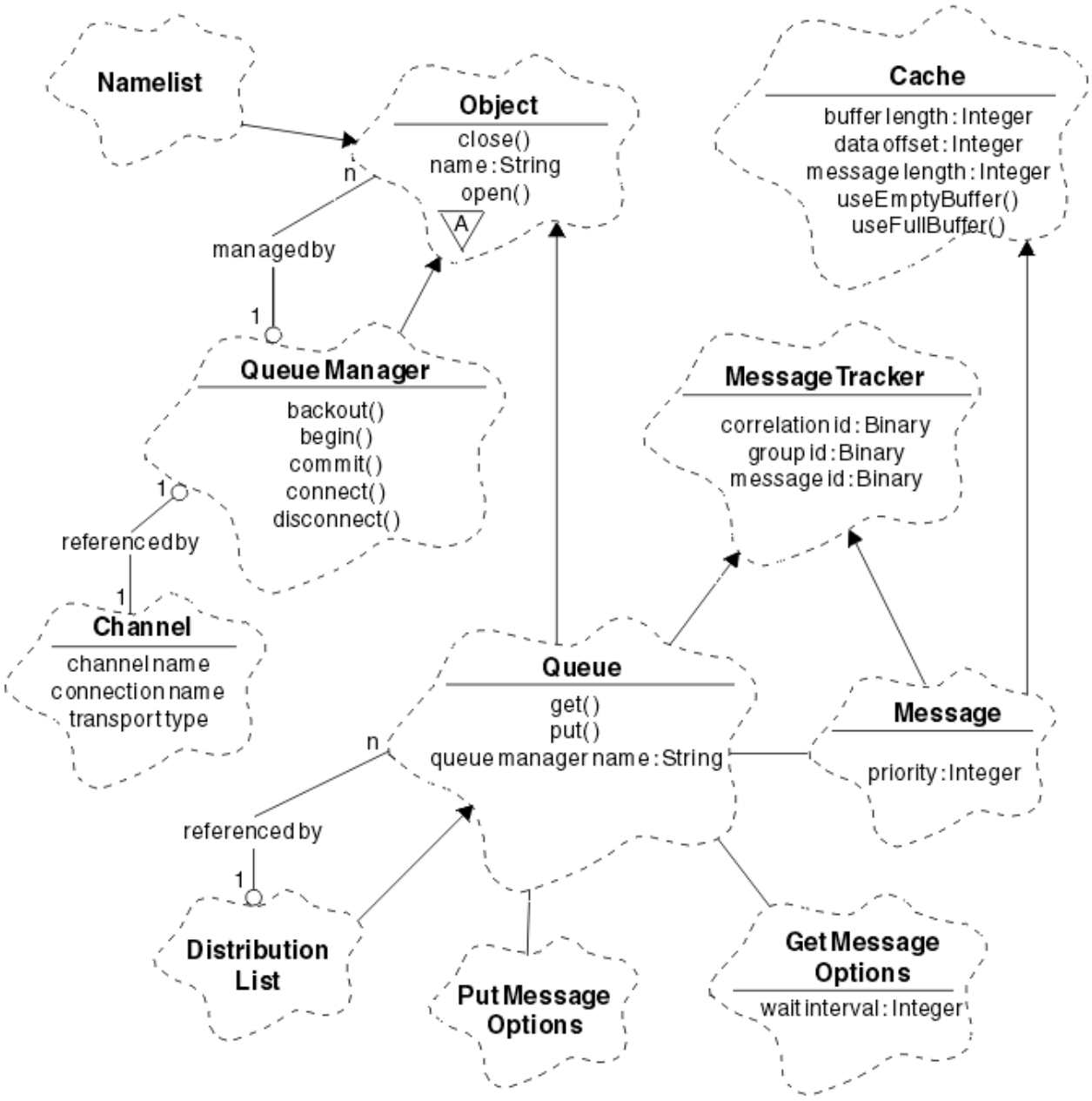
IBM MQ C++ aşağıdaki özellikleri sağlar:

- IBM MQ veri yapılarının otomatik olarak başlatılması.
- Tam zamanında kuyruk yöneticisi bağlantısı ve kuyruk açma.
- Örtük kuyruk kapatma ve kuyruk yöneticisi bağlantısı kesildi.
- Ölü harfli üstbilgi iletimi ve makbuzu.
- IMS köprü üstbilgisi iletimi ve alındı bilgisi.
- Referans ileti üstbilgisi iletimi ve makbuzu.
- İleti girişinin tetiklenmesi.
- CICS bridge üstbilgi iletimi ve makbuzu.
- İş başlığı iletimi ve makbuzu.
- İstemci kanalı tanımı.

Aşağıdaki Booch sınıfı şemaları, tüm sınıfların, tanıtıcı ya da veri yapısı olan yordamsal MQI 'daki (örneğin, C kullanılarak) IBM MQ varlıklarına geniş ölçüde paralel olduğunu gösterir. Tüm sınıflar ImqError sınıfından edinilir (bkz. ImqError C++ sınıfı). Bu sınıf, her nesneyle bir hata koşulunun ilişkilendirilmesine olanak sağlar.



Şekil 50. IBM MQ C++ sınıfları (öğe işleme)



Şekil 51. IBM MQ C++ sınıfları (kuyruk yönetimi)

Booch sınıfı çizgelerini doğru şekilde yorumlamak için aşağıdaki kuralları unutmayın:

- Yöntemler ve dikkate değer öznitelikler *sınıf* adının altında gösterilir.
- Bulut içindeki küçük bir üçgen, bir *soyut sınıfı*belirtir.
- *Kalıtım* üst sınıfa bir ok ile gösterilir.
- Bulutlar arasındaki özel simge olmayan bir çizgi, sınıflar arasındaki *işbirliği ilişkisini* gösterir.
- Sayı ile süslenmiş bir satır, iki sınıf arasındaki *gönderisel ilişkiyi* gösterir. Sayı, herhangi bir zamanda belirli bir ilişkiye katılabilecek nesnelerin sayısını gösterir.

Aşağıdaki sınıflar ve veri tipleri, kuyruk yönetimi sınıflarının C++ yöntem imzalarında kullanılır (bkz. Şekil 51 sayfa 507) ve öğe işleme sınıfları (bkz. Şekil 50 sayfa 506):

- MQBYTE24gibi bayt dizilerini saran ImqBinary sınıfı (bkz. ImqBinary C++ sınıfı).
- **typedef unsigned char ImqBoolean**olarak tanımlanan ImqBoolean veri tipi.
- MQCHAR64gibi karakter dizilerini saran ImqString sınıfı (bkz. ImqString C++ sınıfı).

Veri yapıları olan varlıklar, uygun nesne sınıfları içinde alt üst olur. Tek tek veri yapısı alanları (bkz. C++ ve MQI çapraz başvurusu) yöntemlerle erişilir.

Tutamaçları olan varlıklar ImqObject sınıf sıradüzeni altında gelir (bkz. ImqObject C++ sınıfı) ve MQI için kapsüllenmiş arabirimler sağlayın. Bu sınıfların nesneleri, yordamsal MQI ile göreli olarak gereken yöntem çağrılarının sayısını azaltabilen akıllı davranış sergiler. Örneğin, gerektiği şekilde kuyruk yöneticisi bağlantıları kurabilir ve atabilir ya da uygun seçeneklerle bir kuyruk açıp kapatabilirsiniz.

ImqMessage sınıfı (bkz. ImqMessage C++ sınıfı) MQMD veri yapısını kapsüller ve kullanıcı verileri ve *öğeler* için bir tutma noktası görevi görür (bkz. “C++ dilinde ileti okunması” sayfa 517) önbelleğe alınan arabellek olanaklarını sağlayarak. Kullanıcı verileri için sabit uzunluklu arabellekler sağlayabilir ve arabelleği birçok kez kullanabilirsiniz. Arabellekte bulunan veri miktarı, bir kullanımdan diğerine değişebilir. Alternatif olarak, sistem esnek uzunluklu bir arabellek sağlayabilir ve yönetebilir. Arabelleğin büyüklüğü (iletilerin alınması için kullanılabilir miktar) ve gerçekte kullanılan miktar (iletim için bayt sayısı ya da alınan bayt sayısı) önemli noktalar haline gelir.

İlgili kavramlar

Teknik genel bakış

“C++ örnek programları” sayfa 508
İletileri almayı ve yerleştirmeyi göstermek için dört örnek program sağlanır.

“C++ diliyle ilgili dikkat edilmesi gereken noktalar” sayfa 512
Bu konu derlemi, İleti Kuyruğu Arabirimi 'ni (MQI) kullanan uygulama programları yazarken dikkate almanız gereken C++ dili kullanımı ve kuralları ile ilgili ayrıntıları içerir.

“C++ dilinde ileti verileri hazırlanıyor” sayfa 516
İleti verileri, sistem ya da uygulama tarafından sağlanabilen bir arabellekte hazırlanır. Her iki yöntemin de avantajları vardır. Arabellek kullanma örnekleri verilmiştir.

“IBM MQ için uygulama geliştirilmesi” sayfa 5
İleti göndermek ve almak için uygulamalar geliştirebilir ve kuyruk yöneticilerinizi ve ilgili kaynakları yönetebilirsiniz. IBM MQ , birçok farklı dilde ve çerçevede yazılmış uygulamaları destekler.

İlgili başvurular

“IBM MQ C++ programları oluşturuluyor” sayfa 522
Desteklenen derleyicilerin URL , IBM MQ altyapılarında C++ programlarını ve örneklerini derlemek, bağlamak ve çalıştırmak için kullanılacak komutlarla birlikte listelenir.

C++ ve MQI çapraz başvurusu

IBM MQ C++ sınıfları

C++ örnek programları

İletileri almayı ve yerleştirmeyi göstermek için dört örnek program sağlanır.

Örnek programlar şunlardır:

- MERHABA DÜNYA (imqwrld.cpp)
- SPUT (imqsput.cpp)
- SGET (imqsget.cpp)
- DPUT (imqdput.cpp)

Örnek programlar, Çizelge 73 sayfa 509içinde gösterilen dizinlerde bulunur.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Çizelge 73. Örnek programların yeri		
Çevre	**Kaynağı içeren dizin**	**Oluşturulan dizini içeren dizin programlar**
AIX AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/ia
V 9.3.5 AIX AIX	*MQ_INSTALLATION_PATH*/samp	*MQ_INSTALLATION_PATH*/samp/bin/ca (bkz. not “1” sayfa 509)
IBM i IBM i	`/QIBM/ProdData/mqm/samp/`	(bkz. not “2” sayfa 509)
Linux Linux	*MQ_INSTALLATION_PATH*/samp	Yok
Windows Windows	*MQ_INSTALLATION_PATH*\tools\cplus\samples	*MQ_INSTALLATION_PATH*\tools\cplus\örnekler \bin\vn (bkz. not “3” sayfa 509)
z/OS z/OS	thlqual.SCSQCPPS	

Notlar:

1. V 9.3.5 AIX XLC 17 derleyicisi kullanılarak oluşturulan programlar "`ca`" klasöründe bulunurken, XLC 16 derleyicisi kullanılarak oluşturulan programlar "`ia`" klasöründe bulunur.
2. IBM i IBM i için ILE C++ derleyicisi kullanılarak oluşturulan programlar QMQM kitaplığında yer alır. Kaynak dosyalar `/QIBM/ProdData/mqm/samp`içinde yer almaktadır.
3. Windows Microsoft Visual Studio Visual Studio kullanılarak oluşturulan programlar *MQ_INSTALLATION_PATH*`\tools\cplus\samples\bin\vn`içinde bulunur. Bu derleyicilerle ilgili daha fazla bilgi için bkz. “Windows üzerinde C++ programları oluşturma” sayfa 528.

Örnek program HELLO WORLD (imqwrld.cpp)

Bu C++ örnek programı, ImqMessage sınıfını kullanarak düzenli bir veri paketinin (C yapısı) nasıl konacağını ve elde edeceğini gösterir.

Bu program, ImqMessage sınıfını kullanarak düzenli bir veri paketinin (C yapısı) nasıl konacağını ve elde edeceğini gösterir. Bu örnek, **open**, **close**ve **disconnect**gibi örtük yöntem çağrılarından yararlanarak birkaç yöntem çağrısı kullanır.

z/OS dışındaki tüm platformlarda

IBM MQile sunucu bağlantısı kullanıyorsanız, aşağıdaki yordamlardan birini izleyin:

- Var olan varsayılan kuyruğu kullanmak için, SYSTEM.DEFAULT.LOCAL.QUEUE, herhangi bir parametre geçirmeden **`imqwrlds`** programını çalıştırın
- Devingen olarak atanmış geçici bir kuyruk kullanmak için, **`imqwrlds`** komutunu varsayılan model kuyruğunun adını (SYSTEM.DEFAULT.MODEL.QUEUE.

IBM MQile istemci bağlantısı kullanıyorsanız, aşağıdaki yordamlardan birini izleyin:

- MQSERVER ortam değişkeninin ayarlanması (ek bilgi için *MQSERVER* kısmına bakın) ve **`imqwrldc`**komutunu çalıştırın ya da

- Tipik bir **channel-definition** ' in SYSTEM.DEF.SVRCONN/TCP/*anasistem adı* (1414) olabileceği **queue-name**, **queue-manager-name**ve **channel-definition**parametrelerini parametre olarak geçirme **imqwrldc** komutunu çalıştırın.

Açıkz/OS

z/OS

Örnek JCL **imqwrldr**kullanarak bir toplu iş oluşturun ve çalıştırın.

Daha fazla bilgi için bkz. z/OS Toplu İş, RRS Toplu İş ve CICS .

Örnek kod

```
extern "C" {
#include <stdio.h>
}

#include <imqi.hpp> // IBM MQ C++

#define EXISTING_QUEUE "SYSTEM.DEFAULT.LOCAL.QUEUE"

#define BUFFER_SIZE 12

static char gpszHello[ BUFFER_SIZE ] = "Hello world" ;
int main ( int argc, char * * argv ) {
  ImqQueueManager manager ;
  int iReturnCode = 0 ;

  // Connect to the queue manager.
  if ( argc > 2 ) {
    manager.setName( argv[ 2 ] );
  }
  if ( manager.connect( ) ) {
    ImqQueue * pqueue = new ImqQueue ;
    ImqMessage * pmsg = new ImqMessage ;

    // Identify the queue which will hold the message.
    pqueue -> setConnectionReference( manager );
    if ( argc > 1 ) {
      pqueue -> setName( argv[ 1 ] );


      // The named queue can be a model queue, which will result in
      // the creation of a temporary dynamic queue, which will be
      // destroyed as soon as it is closed. Therefore we must ensure
      // that such a queue is not automatically closed and reopened.
      // We do this by setting open options which will avoid the need
      // for closure and reopening.
      pqueue -> setOpenOptions( MQOO_OUTPUT | MQOO_INPUT_SHARED |
                                MQOO_INQUIRE );
    } else {
      pqueue -> setName( EXISTING_QUEUE );

      // The existing queue is not a model queue, and will not be
      // destroyed by automatic closure and reopening. Therefore we
      // will let the open options be selected on an as-needed basis.
      // The queue will be opened implicitly with an output option
      // during the "put", and then implicitly closed and reopened
      // with the addition of an input option during the "get".
    }

    // Prepare a message containing the text "Hello world".
    pmsg -> useFullBuffer( gpszHello , BUFFER_SIZE );
    pmsg -> setFormat( MQFMT_STRING );

    // Place the message on the queue, using default put message
    // Options.
    // The queue will be automatically opened with an output option.
    if ( pqueue -> put( * pmsg ) ) {
      ImqString strQueue( pqueue -> name( ) );

      // Discover the name of the queue manager.
      ImqString strQueueManagerName( manager.name( ) );
      printf( "The queue manager name is %s.\n",
              (char *)strQueueManagerName );
```

```
      // Show the name of the queue.
      printf( "Message sent to %s.\n", (char *)strQueue );

      // Retrieve the data message just sent ("Hello world" expected)
      // from the queue, using default get message options. The queue
      // is automatically closed and reopened with an input option
      // if it is not already open with an input option. We get the
      // message just sent, rather than any other message on the
      // queue, because the "put" will have set the ID of the message
      // so, as we are using the same message object, the message ID
      // acts as in the message object, a filter which says that we
      // are interested in a message only if it has this
      // particular ID.

      if ( pqueue -> get( * pmsg ) ) {
        int iDataLength = pmsg -> dataLength( );

        // Show the text of the received message.
        printf( "Message of length %d received, ", iDataLength );

        if ( pmsg -> formatIs( MQFMT_STRING ) ) {
          char * pszText = pmsg -> bufferPointer( );

          // If the last character of data is a null, then we can
          // assume that the data can be interpreted as a text
          // string.
          if ( ! pszText[ iDataLength - 1 ] ) {
            printf( "text is \"%s\".\n", pszText );
          } else {
            printf( "no text.\n" );
          }

        } else {
          printf( "non-text message.\n" );
        }
      } else {
        printf( "ImqQueue::get failed with reason code %ld\n",
                pqueue -> reasonCode( ) );
        iReturnCode = (int)pqueue -> reasonCode( );
      }

    } else {
      printf( "ImqQueue::open/put failed with reason code %ld\n",
              pqueue -> reasonCode( ) );
      iReturnCode = (int)pqueue -> reasonCode( );
    }

    // Deletion of the queue will ensure that it is closed.
    // If the queue is dynamic then it will also be destroyed.
    delete pqueue ;
    delete pmsg ;

  } else {
    printf( "ImqQueueManager::connect failed with reason code %ld\n"
            manager.reasonCode( ) );
    iReturnCode = (int)manager.reasonCode( );
  }

  // Destruction of the queue manager ensures that it is
  // disconnected.  If the queue object were still available
  // and open (which it is not), the queue would be closed
  // prior to disconnection.

  return iReturnCode ;
}
```

Örnek programlar SPUT (imqsput.cpp) ve SGET (imqsget.cpp)

Bu C++ programları, iletileri adlandırılmış bir kuyruğa yerleştirir ve bu kuyruktan ileti alır.

Bu örnekler aşağıdaki sınıfların kullanımını gösterir:

- ImqError (bkz. ImqError C++ sınıfı)
- ImqMessage (bkz. ImqMessage C++ sınıfı)
- ImqObject (bkz. ImqObject C++ sınıfı)
- ImqQueue (bkz. ImqQueue C++ sınıfı)

- ImqQueueManager (bkz. ImqQueueManager C++ sınıfı)

Programları çalıştırmak için uygun yönergeleri izleyin.

z/OS dışındaki tüm platformlarda

1. **imqsputs** *kuyruk-adı*komutunu çalıştırın.
2. Konsola metin satırlarını yazın. Bu hatlar, belirlenen kuyruğa ileti olarak yerleştirilir.
3. Girişi sonlandırmak için boş bir satır girin.
4. Tüm satırları almak ve konsolda görüntülemek için **imqsretrieve** *kuyruk-adı* komutunu çalıştırın.

z/OS Ek bilgi için bkz. “z/OS Batch, RRS Batch ve CICS üzerinde C++ programları oluşturma” sayfa 530 .

Açıkz/OS

z/OS

1. Örnek JCL **imqsputr**kullanarak bir toplu iş oluşturun ve çalıştırın. İletiler SYSIN veri kümesinden okunur.
2. Örnek JCL **imqsgetr**kullanarak bir toplu iş oluşturun ve çalıştırın. İletiler kuyruktan alınır ve SYSPRINT veri kümesine gönderilir.

Örnek program DPUT (imqdput.cpp)

Bu C++ örnek programı, iletileri iki kuyruklardan oluşan bir dağıtım listesine koyar.

DPUT, ImqDistributionList sınıfının kullanımını gösterir (bkz. ImqDistributionList C++ sınıfı). Bu örnek z/OSüzerinde desteklenmez.

1. İletileri adlandırılmış iki kuyruğa yerleştirmek için **imqdputs** *queue-name-1 queue-name-2* komutunu çalıştırın.
2. Bu kuyruklardan iletileri almak için **imqsretrieve** *queue-name-1* ve **imqsretrieve** *queue-name-2* komutunu çalıştırın.

C++ diliyle ilgili dikkat edilmesi gereken noktalar

Bu konu derlemi, İleti Kuyruğu Arabirimi 'ni (MQI) kullanan uygulama programları yazarken dikkate almanız gereken C++ dili kullanımı ve kuralları ile ilgili ayrıntıları içerir.

C++ Üstbilgi dosyaları

Üstbilgi dosyaları, IBM MQ uygulama programlarını C++ dilinde yazmanıza yardımcı olmak için MQI tanımlamasının bir parçası olarak sağlanır.

Bu üstbilgi dosyaları aşağıdaki tabloda özetlenmiştir.

Çizelge 74. C/C++ üstbilgi dosyaları

Dosya Adı	**İçerik**
IMQI.HPP	C++ MQI Sınıfları (CMQC.H ve IMQTYPE.H)
IMQTYPE.H	**ImqBoolean** veri tipini tanımlar
CMQC.H	MQI veri yapıları ve bildirge değişmezleri

Uygulamaların taşınabilirliğini artırmak için, **#include** ön işlemci yönergesinde üstbilgi dosyasının adını küçük harfle kodlayın:

```
#include <imqi.hpp> // C++ classes
```

C++ yöntemleri ve öznitelikleri

Yöntem adları büyük ve küçük harf karışık. Parametreler ve dönüş değerleri için çeşitli noktalar geçerlidir. Özniteliklere küme (set) ve get (get) yöntemleri kullanılarak erişilir.

const olan yöntemlerin değiştirgeleri yalnızca giriş içindir. İşaretçi (*) ya da başvuru (&) içeren imzalı parametreler başvuruya göre iletilir. Bir işaretçi ya da başvuru içermeyen dönüş değerleri değere geçirilir; döndürülen nesneler söz konusu olduğunda, bunlar çağıranın sorumluluğu haline gelen yeni varlıklardır.

Bazı yöntem imzaları, belirtilmezse varsayılan değeri alacak öğeleri içerir. Bu tür öğeler her zaman imzaların sonunda bulunur ve bir eşittir işaretiyle (=) gösterilir; eşittir işaretinden sonraki değer, öğe atlanırsa geçerli olan varsayılan değeri gösterir.

Bu sınıflardaki tüm yöntem adları, küçük harfle başlayarak büyük ve küçük harfe karıştırılır. Yöntem adı dışındaki her sözcük büyük harfle başlar. Kısaltmalar, anlamları yaygın olarak anlaşılmadıkça kullanılmaz. Kullanılan kısaltmalar arasında *id* (kimlik için) ve *sync* (eşitleme için) yer alır.

Nesne öznitelikl erine set ve get yöntemleri kullanılarak erişilir. Bir set yöntemi *set* sözcüğüyle başlar; get yönteminin öneki yok. Bir öznitelik *salt okunur*ise, ayarlama yöntemi yoktur.

Öznitelikler, nesne oluşturma sırasında geçerli durumlarla kullanıma hazırlanır ve bir nesnenin durumu her zaman tutarlıdır.

C++ dilinde veri tipleri

Tüm veri tipleri C **typedef** deyimiyle tanımlanır.

ImqBoolean tipi IMQTYPE.H içinde **imzasız karakter** olarak tanımlanır ve TRUE ve FALSE değerlerini içerebilir. **MQBYTE** dizileri yerine **ImqBinary** sınıf nesnelerini ve **char ***yerine **ImqString** sınıf nesnelerini kullanabilirsiniz. Birçok yöntem, depolama yönetimini kolaylaştırmak için **char** ya da **MQBYTE** işaretçileri yerine nesneleri döndürür. Tüm dönüş değerleri çağıranın sorumluluğu olur ve döndürülen bir nesne söz konusu olduğunda, saklama alanı silme işlemi kullanılarak atılabilir.

C++ dilinde ikili dizgiler işleniyor

İkili veri dizgileri, **ImqBinary** sınıfının nesneleri olarak bildirilir. Bu sınıftaki nesneler, bilinen C işleçleri kullanılarak kopyalanabilir, karşılaştırılabilir ve ayarlanabilir. Örnek kod sağlanmıştır.

Aşağıdaki kod örneği, ikili dizilim üzerindeki işlemleri gösterir:

```
#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.
...
```

C++ dilinde karakter dizgilerini işleme

Karakter verileri genellikle, bir dönüştürme işleci kullanılarak **char *** karakterine dönüştürülebilir **ImqString** sınıf nesnelerinde döndürülür. ImqString sınıfı, karakter dizgilerinin işlenmesine yardımcı olacak yöntemler içerir.

Karakter verileri MQI C++ yöntemleri kullanılarak kabul edildiğinde ya da döndürüldüğünde, karakter verileri her zaman boş sonlandırılır ve herhangi bir uzunlukta olabilir. Ancak IBM MQ , bilgilerin

kesilmesine neden olabilecek bazı sınırlar koymaktadır. Depolama yönetimini kolaylaştırmak için karakter verileri genellikle **ImqString** sınıf nesnelerinde döndürülür. Bu nesneler, sağlanan dönüştürme işleci kullanılarak **char *** biçimine dönüştürülebilir ve **char *** gerekli olduğu birçok durumda *salt okunur* amaçlarla kullanılabilir.

Not: Bir **ImqString** sınıf nesnesinden **char *** dönüştürme sonucu boş değerli olabilir.

C işlevleri **char ***üzerinde kullanılabilse de, **ImqString** sınıfının tercih edilen özel yöntemleri vardır; **işleç uzunluğu** () **strlen** ve **storage** () ile eşdeğerdir karakter verileri için ayrılan belleği gösterir.

C++ dilinde nesnelerin ilk durumu

Tüm nesneler, öznitelikleri tarafından yansıtılan tutarlı bir başlangıç durumuna sahiptir. İlk değerler sınıf tanımlarında tanımlanır.

C++ dilinde C kullanılması

C++ programından C işlevlerini kullandığınızda, uygun üstbilgileri ekleyin.

Aşağıdaki örnek, C++ programında bulunan `string.h` öğesini göstermektedir:

```
extern "C" {
#include <string.h>
}
```

C++ notasyonel kuralları

Bu örnek, yöntemlerin nasıl çağrılacağını ve parametrelerin nasıl bildirileceğini gösterir.

Bu kod örneği, **ImqBoolean ImqQueue**:: **get** (**ImqMessage &** *msg*)

Değiştirgeleri aşağıdaki gibi bildirin ve kullanın:

```
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( );

...
}
```

C++ dilinde örtük işlemler

Bir yöntemin başarıyla yürütülmesine ilişkin önkoşul koşullarını karşılamak için *tam zamanında*örtük olarak birden çok işlem gerçekleştirilebilir. Bu örtük işlemler bağlanma, açma, yeniden açma, kapatma ve bağlantı kesme işlemleridir. Sınıf özniteliklerini kullanarak, örtük davranışı denetleyebilir ve açabilirsiniz.

Bizimle

MQI çağrısıyla sonuçlanan herhangi bir yöntem için ImqQueueManager nesnesi otomatik olarak bağlanır (bkz. C++ ve MQI çapraz başvurusu).

Aç

Bir MQGET, MQINQ, MQPUT ya da MQSET çağrısıyla sonuçlanan herhangi bir yöntem için ImqObject nesnesi otomatik olarak açılır. Bir ya da daha çok ilgili **open option** değeri belirtmek için **openFor** yöntemini kullanın.

Yeniden aç

Bir ImqObject , nesnenin zaten açık olduğu, ancak var olan **açma seçeneklerinin** MQI çağrısına izin vermek için yeterli olmadığı bir MQGET, MQINQ, MQPUT ya da MQSET çağrısıyla sonuçlanan herhangi bir yöntem için otomatik olarak yeniden açılır. Nesne, MQCO_NONE geçici **kapatma seçenekleri** değeri kullanılarak geçici olarak kapatılır. İlgili bir öğe eklemek için **openFor** yöntemini kullanın**açma seçeneği**.

Yeniden açma belirli koşullarda sorunlara neden olabilir:

- Geçici bir dinamik kuyruk kapatıldığında yok edilir ve hiçbir zaman yeniden açılamaz.
- Dışlayıcı giriş için açılan bir kuyruğa (açık ya da varsayılan olarak), kapanış ve yeniden açma sırasında fırsat penceresinde başkaları erişebilir.
- Bir kuyruk kapatıldığında göz atma imleci konumu kaybolur. Bu durum, kapanmayı ve yeniden açılmasını önlemez, ancak MQGMO_BROWSE_FIRST yeniden kullanılıncaya kadar imlecin sonraki kullanımını önler.
- Bir kuyruk kapatıldığında, alınan son iletinin bağlamı kaybolur.

Bu durumlardan herhangi biri ortaya çıkarsa ya da öngörülürse, bir nesne açılmadan önce (belirtik ya da örtük olarak) yeterli **açma seçeneklerini** belirtik olarak ayarlayarak yeniden açılmaktan kaçının.

Karmaşık kuyruk işleme durumlarına ilişkin **açma seçeneklerinin** belirtik olarak ayarlanması daha iyi performansa neden olur ve yeniden açma kullanımıyla ilişkili sorunları önler.

Kapat

ImqObject , nesne durumunun artık geçerli olmadığı herhangi bir noktada otomatik olarak kapatılır; örneğin, bir ImqObject bağlantı başvurusu kesilirse ya da bir ImqObject nesnesi yok edilirse.

Bağlantıyı kes

Bağlantının artık geçerli olmadığı herhangi bir noktada (örneğin, bir ImqObject bağlantı başvurusu kesildiyse ya da bir ImqQueueManager nesnesi yok edildiyse) ImqQueueManager 'ın bağlantısı otomatik olarak kesilir.

C++ dilinde ikili ve karakter dizgileri

ImqString sınıfı, geleneksel *char ** veri biçimini kapsıyor. ImqBinary sınıfı ikili byte dizisini kapsıyor. Karakter verilerini ayarlanan bazı yöntemler verileri kesebilir.

Karakter ayarlanan yöntemler (**char ***) veri her zaman verilerin bir kopyasını alır, ancak belirli sınırlar IBM MQtarafından zorunlu kılındığı için bazı yöntemler kopyanın kesilmesine neden olabilir.

ImqString sınıfı (bkz. ImqString C++ sınıfı) Geleneksel **char *** ürününü kapsüller ve aşağıdakiler için destek sağlar:

- Karşılaştırma
- Bitiştirme
- Kopyalama
- Tamsayı-metin ve metinden tamsayıya dönüştürme
- Simge (sözcük) alma
- Büyük harf çevirisi

ImqBinary sınıfı (bkz. ImqBinary C++ sınıfı) İsteğe bağlı büyüklükteki ikili byte dizilerini kapsüller. Özellikle aşağıdaki öznitelikleri tutmak için kullanılır:

- **muhasebe simgesi** (MQBYTE32)
- **bağlantı etiketi** (MQBYTE128)
- **ilinti tanıtıcısı** (MQBYTE24)
- **tesis simgesi** (MQBYTE8)
- **grup tanıtıcısı** (MQBYTE24)
- **yönetim ortamı tanıtıcısı** (MQBYTE24)
- **ileti tanıtıcısı** (MQBYTE24)
- **ileti simgesi** (MQBYTE16)
- **hareket eşgörünümü tanıtıcısı** (MQBYTE16)

Bu özniteliklerin aşağıdaki sınıfların nesnelerine ait olduğu yer:

- ImqCICSBridgeHeader (bkz. ImqCICSBridgeHeader C++ sınıfı)
- ImqGetMessageOptions (bkz. ImqGetMessageOptions C++ sınıfı)
- ImqIMSBridgeHeader (bkz. ImqIMSBridgeHeader C++ sınıfı)
- ImqMessageTracker (bkz. ImqMessageTracker C++ sınıfı)
- ImqQueueManager (bkz. ImqQueueManager C++ sınıfı)
- ImqReferenceÜstbilgisi (bkz. ImqReferenceHeader C++ class)
- ImqWorkÜstbilgisi (bkz. ImqWorkÜstbilgi C++ sınıfı)

ImqBinary sınıfı, karşılaştırma ve kopyalama için de destek sağlar.

C++ dilinde desteklenmeyen işlevler

IBM MQ C++ sınıfları ve yöntemleri IBM MQ platformundan bağımsızdır. Bu nedenle, belirli platformlarda desteklenmeyen bazı işlevler sunabilir.

Desteklenmeyen bir altyapıda işlev kullanmayı denerseniz, işlev IBM MQ tarafından saptanır, ancak C++ dil bağ tanımları tarafından saptanmaz. IBM MQ , diğer MQI hataları gibi, hatayı programınıza bildirir.

C++ dilinde ileti alışverişi

Bu konu derlemi, C + + ' da iletilerin nasıl hazırlanacağını, okunacağını ve yazılacağını ayrıntılı olarak içerir.

C++ dilinde ileti verileri hazırlanıyor

İleti verileri, sistem ya da uygulama tarafından sağlanabilen bir arabellekte hazırlanır. Her iki yöntemin de avantajları vardır. Arabellek kullanma örnekleri verilmiştir.

Bir ileti gönderdiğinizde, ileti verileri ilk olarak ImqCache nesnesi tarafından yönetilen bir arabellekte hazırlanır (bkz. ImqCache C++ sınıfı). Her ImqMessage nesnesiyle bir arabellek (kalıtım yoluyla) ilişkilendirilir (bkz. ImqMessage C++ sınıfı): uygulama tarafından (**useEmptyBuffer** ya da **useFullBuffer** yöntemi kullanılarak) ya da sistem tarafından otomatik olarak sağlanabilir. İleti arabelleğini sağlayan uygulamanın yararı, uygulamanın hazırlanan veri alanlarını doğrudan kullanabilmesi nedeniyle birçok durumda veri kopyalamaya gerek olmamasıdır. Dezavantajı, sağlanan arabelleğin sabit uzunlukta olması.

Arabellek yeniden kullanılabilir ve iletilen bayt sayısı, iletiden önce **setMessageLength** yöntemi kullanılarak her seferinde değişebilir.

Sistem tarafından otomatik olarak sağlandığında, kullanılabilir bayt sayısı sistem tarafından yönetilir ve veriler, örneğin, ImqCache **write** yöntemi ya da ImqMessage **writeItem** yöntemi kullanılarak ileti arabelleğine kopyalanabilir. İleti arabelleği gereksinme göre büyür. Arabellek büyüdükçe, önceden yazılmış veriler kaybolmaz. Büyük ya da çok parçalı bir ileti sıralı olarak yazılabilir.

Aşağıdaki örnekler, basitleştirilmiş ileti gönderimini göstermektedir.

1. Kullanıcı tarafından sağlanan bir arabellekte hazırlanmış verileri kullan

```
char szBuffer[ ] = "Hello world" ;

msg.useFullBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
```

2. Kullanıcı tarafından sağlanan bir arabellekte, arabellek büyüklüğünün veri büyüklüğünü aştığı hazırlanmış verileri kullan

```
char szBuffer[ 24 ] = "Hello world" ;

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.setMessageLength( 12 );
```

3. Verileri kullanıcı tarafından sağlanan bir arabelleğe kopyalama

```
char szBuffer[ 12 ];

msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

4. Verilerin sistem tarafından sağlanan bir arabelleğe kopyalanması

```
msg.setFormat( MQFMT_STRING );
msg.write( 12, "Hello world" );
```

5. Verileri nesneleri kullanarak sistem tarafından sağlanan bir arabelleğe kopyalama (içerik yanı sıra ileti biçimini de ayarlanan nesneler)

```
ImqString strText( "Hello world" );

msg.writeItem( strText );
```

C++ dilinde ileti okunması

Uygulama ya da sistem tarafından bir arabellek sağlanabilir. Verilere doğrudan arabellekten erişilebilir ya da sırayla okunabilir. Her ileti tipine eşdeğer bir sınıf vardır. Örnek kod verildi.

Veri alırken, uygulama ya da sistem uygun bir ileti arabelleği sağlayabilir. Aynı arabellek, belirli bir ImqMessage nesnesi için birden çok iletim ve birden çok giriş için kullanılabilir. İleti arabelleği otomatik olarak sağlanırsa, alınan verilerin uzunluğunu karşılamak için büyür. Ancak, uygulama tarafından sağlanan bir ileti arabelleği, alınan verileri tutacak kadar büyük olmayabilir. Daha sonra, ileti girişi için kullanılan seçeneklere bağlı olarak kesme ya da hata oluşabilir.

Gelen verilere doğrudan ileti arabelleğinden erişilebilir; bu durumda, veri uzunluğu gelen verilerin toplam miktarını gösterir. Diğer bir seçenek olarak, gelen veriler ileti arabelleğinden sıralı olarak okunabilir. Bu durumda, veri işaretçisi gelen verilerin sonraki baytını adresler ve veriler her okunuşunda veri işaretçisi ve veri uzunluğu güncellenir.

Öğeler , sıralı olarak ve ayrı olarak işlenmesi gereken, ileti arabelleğinin kullanıcı alanında bulunan bir iletinin parçalarıdır. Normal kullanıcı verilerinin yanı sıra, bir öğe bir ölmeyen harf üstbilgisi ya da bir tetikleyici iletisi olabilir. Öğeler her zaman ileti biçimleriyle ilişkilendirilir; ileti biçimleri her zaman öğelerle **ilişkilendirilmez** .

Tanınabilir bir IBM MQ ileti biçimine karşılık gelen her öğe için bir nesne sınıfı vardır. Bir tane gitmeyen harf üstbilgisi için, bir tane de tetikleyici iletisi için. Kullanıcı verileri için nesne sınıfı yok. Yani, tanınabilir biçimler tükendikten sonra, geri kalan biçimlerin işlenmesi uygulama programına bırakılır. Kullanıcı verilerine ilişkin sınıflar, ImqItem sınıfı özelleştirilerek yazılabilir.

Aşağıdaki örnek, hayali bir durumda, kullanıcı verilerinden önce gelebilecek bir dizi olası öğeyi dikkate alan bir ileti girişini göstermektedir. Öğe olmayan kullanıcı verileri, tanımlanabilen öğelerden sonra oluşan

herhangi bir öğe olarak tanımlanır. Otomatik arabellek (varsayılan), isteğe bağlı ileti verisi miktarını tutmak için kullanılır.

```
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.                      */
      ...

    }

  }
}
```

Bu örnekte FMT_USERCLASS , `UserClass`sınıfındaki bir nesneyle ilişkilendirilmiş 8 karakterlik biçim adını gösteren bir değişmez vardır ve uygulama tarafından tanımlanır.

`UserClass` , ImqItem sınıfından türetilir (bkz. ImqItem C++ sınıfı) ve o sınıftan sanal **copyOut** ve **pasteIn** yöntemlerini uygular.

Sonraki iki örnekte, ImqDeadLetterHeader sınıfındaki kod gösterilmektedir (bkz. ImqDeadLetterHeader C++ sınıfı). İlk örnek, özel kapsüllenmiş ileti- *yazma* kodunu gösterir.

```
// 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 ;
}
```

İkinci örnek, özel kapsüllenmiş ileti- *okunuyor* kodunu gösterir.

```
// 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 ;
}
```

Otomatik arabellekle, arabellek saklama alanı *uçucu*olur. Yani, her **get** yöntemi çağrıldıktan sonra arabellek verileri farklı bir fiziksel yerde tutulabilir. Bu nedenle, arabellek verilerine her gönderme yapıldığında, ileti verilerine erişmek için **bufferPointer** ya da **dataPointer** yöntemlerini kullanın.

Bir programın, ileti verilerini almak için sabit bir alan ayırmasını isteyebilirsiniz. Bu durumda, **get** yöntemini kullanmadan önce **useEmptyBuffer** yöntemini çağırın.

Sabit, otomatik olmayan bir alanın kullanılması iletileri büyüklük üst sınırıyla sınırlar; bu nedenle, ImqGetMessageOptions nesnesinin MQGMO_ACCEPT_TRUNCATED_MSG seçeneğini göz önünde bulundurmanız önemlidir. Bu seçenek belirlenmezse (varsayılan), MQRC_TRUNCATED_MSG_FAILED neden kodu beklenebilir. Bu seçenek belirtilirse, uygulamanın tasarımına bağlı olarak MQRC_TRUNCATED_MSG_ACCEPTED neden kodu beklenir.

Sonraki örnekte, iletileri almak için sabit bir depolama alanının nasıl kullanılabileceği gösterilmektedir:

```
char * pszBuffer = new char[ 100 ];

msg.useEmptyBuffer( pszBuffer, 100 );
gmo.setOptions( MQGMO_ACCEPT_TRUNCATED_MSG );
queue.get( msg, gmo );

delete [ ] pszBuffer ;
```

Bu kod parçasında, arabellek her zaman **bufferPointer** yönteminin aksine *pszBuffer*ile doğrudan adreslenir. Ancak, genel amaçlı erişim için **dataPointer** yönteminin kullanılması daha iyidir. Uygulamanın (ImqCache sınıf nesnesi değil) kullanıcı tanımlı (otomatik olmayan) bir arabelleği atması gerekir.

Dikkat: useEmptyBuffer ile boş değerli bir gösterge ve sıfır uzunluklu bir arabellek belirtilmesi, beklendiği gibi sıfır uzunluklu sabit uzunluklu bir arabelleği aday göstermez. Bu birleşim, önceki kullanıcı tanımlı arabelleği yoksayma ve otomatik arabellek kullanımına geri döndürme isteği olarak yorumlanır.

C++ dilinde teslim kuyruğuna ileti yazılması

İleti gönderilmeyen iletiler kuyruğuna yazılması için örnek program kodu.

Çok parçalı bir iletinin tipik bir örneği, bir fazla harfli üstbilgi içeren bir iletidir. İşlenemeyen bir iletideki veriler, almayan harf üstbilgisine eklenir.

```
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 );
```

C++ dilinde IMS köprüsüne ileti yazılması

IMS köprüsüne ileti yazmak için örnek program kodu.

IBM MQ - IMS köprüsüne gönderilen iletiler özel bir üstbilgi kullanabilir. IMS köprü üstbilgisinin başına normal ileti verileri eklenir.

```
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 );
```

C++ dilinde CICS bridge dosyasına ileti yazılması

CICS bridge' e ileti yazmak için örnek program kodu.

CICS bridge kullanılarak IBM MQ for z/OS ' e gönderilen iletiler özel bir üstbilgi gerektirir. CICS bridge üstbilgisinin başına normal ileti verileri eklenir.

```
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 );
```

C++ dilinde çalışma üstbilgisiyle ileti yazılması

z/OS Workload Manager tarafından yönetilen bir kuyruğa yönlendirilen bir iletinin yazılması için örnek program kodu.

z/OS Workload Manager tarafından yönetilen bir kuyruğa yönlendirilen IBM MQ for z/OSadresine gönderilen iletiler için özel bir üstbilgi gerekir. İş üstbilgisinin başına normal ileti verileri eklenir.

```
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 );
```

IBM MQ C++ programları oluşturuluyor

Desteklenen derleyicilerin URL , IBM MQ altyapılarında C++ programlarını ve örneklerini derlemek, bağlamak ve çalıştırmak için kullanılacak komutlarla birlikte listelenir.

Desteklenen her platform ve IBM MQsürümü için derleyicilerin bir listesi için bkz. IBM MQ için Sistem Gereksinimleri.

IBM MQ C++ programınızı derlemek ve bağlamak için gereksinim duyduğunuz komut, kuruluşunuza ve gereksinimlerinize bağlıdır. Aşağıdaki örnekler, bazı derleyicilere ilişkin tipik derleme ve bağlama komutlarını birkaç altyapıda varsayılan IBM MQ kuruluşunu kullanarak gösterir.

AIX AIX üzerinde C++ programları oluşturma

XL C Enterprise Edition derleyicisini kullanarak AIX üzerinde IBM MQ C++ programları oluşturun.

V 9.3.5 XLC 16 ve XLC 17 derleyicileri arasındaki derleyici seçeneklerinin farklı eşlemesiyle ilgili ek bilgi için Seçeneklerin eşlenmesibaşlıklı konuya bakın.

V 9.3.5 Deprecated AIX üzerinde XL C/C++ for AIX 16 derleyicisi desteği IBM MQ 9.3.5tarafından kullanımdan kaldırılmıştır.

Müşteri

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

32 bit iş parçacıklı olmayan uygulama

```
xlC -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ia -limqb23ia -lmqic
```

32 bit iş parçacıklı uygulama

```
xlC_r -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ia_r -limqb23ia_r -lmqic_r
```

64 bit iş parçacıklı olmayan uygulama

```
xlC -q64 -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ia -limqb23ia -lmqic
```

64 bit iş parçacıklı uygulama

```
xlC_r -q64 -o imqsputc_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ia_r -limqb23ia_r -lmqic_r
```

V 9.3.5 32 bit iş parçacıklı olmayan uygulama (XLC 17)

```
ibm-clang++_r -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ca -limqb23ca -lmqic
```

V 9.3.5 32 bit iş parçacıklı uygulama (XLC 17)

```
ibm-clang++_r -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqc23ca_r -limqb23ca_r -lmqic_r
```

V 9.3.5 64 bit iş parçacıklı olmayan uygulama (XLC 17)

```
ibm-clang++_r -m64 -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ca -limqb23ca -lmqic
```

V 9.3.5 64 bit iş parçacıklı uygulama (XLC 17)

```
ibm-clang++_r -m64 -o imqsputc_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqc23ca_r -limqb23ca_r -lmqic_r
```

Sunucu

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

32 bit iş parçacıklı olmayan uygulama

```
xlC -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ia -limqb23ia -lmqm
```

32 bit iş parçacıklı uygulama

```
xlC_r -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ia_r -limqb23ia_r -lmqm_r
```

64 bit iş parçacıklı olmayan uygulama

```
xlC -q64 -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ia -limqb23ia -lmqm
```

64 bit iş parçacıklı uygulama

```
xlC_r -q64 -o imqsput_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ia_r -limqb23ia_r -lmqm_r
```

V 9.3.5 **32 bit iş parçacıklı olmayan uygulama (XLC 17)**

```
ibm-clang++_r -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ca -limqb23ca -lmqm
```

V 9.3.5 **32 bit iş parçacıklı uygulama (XLC 17)**

```
ibm-clang++_r -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -limqs23ca_r -limqb23ca_r -lmqm_r
```

V 9.3.5 **64 bit iş parçacıklı olmayan uygulama (XLC 17)**

```
ibm-clang++_r -m64 -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ca -limqb23ca -lmqm
```

V 9.3.5 **64 bit iş parçacıklı uygulama (XLC 17)**

```
ibm-clang++_r -m64 -o imqsput_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -limqs23ca_r -limqb23ca_r -lmqm_r
```

IBM i IBM i üzerinde C++ programları oluşturma

ILE C++ derleyicisini kullanarak IBM i üzerinde IBM MQ C++ programları oluşturun.

IBM ILE C++ for IBM i , C++ programları için yerel bir derleyicidir. Aşağıdaki yönergelerde, *Merhaba Dünya!* kullanılarak IBM MQ C++ uygulamaları yaratmak için bu derleyicinin nasıl kullanılacağı açıklanmaktadır. Örnek olarak IBM MQ örnek programı.

1. ILE C++ for IBM i derleyicisini *Read Me first!* (Önce Beni Oku!) başlıklı bölüme bakın. Ürünle birlikte gönderilen el kitabı.
2. QCXXN kitaplığının kitaplık listenizde olduğundan emin olun.
3. HELLO WORLD örnek programını yaratın:
   a. Modül yarat:

```
CRTCPPMOD MODULE(MYLIB/IMQWRLD) +
SRCSTMF('/QIBM/ProdData/mqm/samp/imqwrld.cpp') +
INCDIR('/QIBM/ProdData/mqm/inc') DFTCHAR(*SIGNED) +
TERASPACE(*YES)
```

C++ örnek programlarına ilişkin kaynak `/QIBM/ProdData/mqm/samp` ve içerme dosyalarını `/QIBM/ProdData/mqm/inc`içinde bulabilirsiniz.

Diğer bir seçenek olarak, kaynak `SRCFILE(QCPPSRC/LIB) SRCMBR(IMQWRLD)`kitaplığında bulunabilir.

b. Bir program nesnesi oluşturmak için bunu IBM MQtarafından sağlanan hizmet programlarıyla bağlayın:

```
CRTPGM PGM(MYLIB/IMQWRLD) MODULE(MYLIB/IMQWRLD) +
BNDSRVPGM(QMQM/IMQB23I4 QMQM/IMQS23I4)
```

İş parçacıklı bir uygulama oluşturmak için yeniden girişli hizmet programlarını kullanın:

```
CRTPGM PGM(MYLIB/IMQWRLD) MODULE(MYLIB/IMQWRLD) +
BNDSRVPGM(QMQM/IMQB23I4[_R] QMQM/IMQS23I4[_R])
```

c. SYSTEM.DEFAULT.LOCAL.QUEUE:

```
CALL PGM(MYLIB/IMQWRLD)
```

Linux Linux üzerinde C++ programları oluşturma

GNU g + + derleyicisini kullanarak Linux üzerinde IBM MQ C++ programları oluşturun.

System p

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

İstemci:System p

32 bit iş parçacıklı olmayan uygulama

```
g++ -m32 -o imqsputc_32 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

32 bit iş parçacıklı uygulama

```
g++ -m32 -o imqsputc_r32 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r
```

64 bit iş parçacıklı olmayan uygulama

```
g++ -m64 -o imqsputc_64 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

64 bit iş parçacıklı uygulama

```
g++ -m64 -o imqsputc_r64 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r
```

Sunucu:System p

32 bit iş parçacıklı olmayan uygulama

```
g++ -m32 -o imqsput_32 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

32 bit iş parçacıklı uygulama

```
g++ -m32 -o imqsput_r32 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r
-limqb23gl_r -lmqm_r
```

64 bit iş parçacıklı olmayan uygulama

```
g++ -m64 -o imqsput_64 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

64 bit iş parçacıklı uygulama

```
g++ -m64 -o imqsput_r64 imqsput.cpp -fsigned-char -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r
```

IBM Z

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

İstemci:IBM Z

32 bit iş parçacıklı olmayan uygulama

```
g++ -m31 -fsigned-char -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

32 bit iş parçacıklı uygulama

```
g++ -m31 -fsigned-char -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r
-lpthread
```

64 bit iş parçacıklı olmayan uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

64 bit iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

Sunucu:IBM Z

32 bit iş parçacıklı olmayan uygulama

```
g++ -m31 -fsigned-char -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

32 bit iş parçacıklı uygulama

```
g++ -m31 -fsigned-char -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

64 bit iş parçacıklı olmayan uygulama

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl -limqb23gl -lmqm
```

64 bit iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

x86-64 (32 bit)

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

İstemci: x86-64 (32 bit)

32 bit iş parçacıklı olmayan uygulama

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -L
MQ_INSTALLATION_PATH/lib -Wl,
-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl -limqb23gl -lmqic
```

32 bit iş parçacıklı uygulama

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

64 bit iş parçacıklı olmayan uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -L
MQ_INSTALLATION_PATH/lib64
-Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl -limqb23gl
-lmqic
```

64 bit iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -L
MQ_INSTALLATION_PATH/lib64
-Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqc23gl_r -limqb23gl_r
-lmqic_r -lpthread
```

Sunucu: x86-64 (32 bit)

32 bit iş parçacıklı olmayan uygulama

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl -limqb23gl -lmqm
```

32 bit iş parçacıklı uygulama

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

64 bit iş parçacıklı olmayan uygulama

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -L
MQ_INSTALLATION_PATH/lib64
-Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

64 bit iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -L
MQ_INSTALLATION_PATH/lib64
-Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r
-lmqm_r -lpthread
```

Windows Windows üzerinde C++ programları oluşturma

Microsoft Visual Studio C++ derleyicisini kullanarak Windows üzerinde IBM MQ C++ programları oluşturun.

Uyarı: IBM MQ ile gönderilen kitaplıklar, statik kitaplıklar değil, dinamik kitaplıklardır. IBM MQ , yalnızca derleme sırasında kullanabileceğiniz "`import libraries`" olarak bilinen bir şey sağlar. Yürütme ortamı için dinamik kitaplıkları kullanmalısınız.

IBM MQ 8.0.0 Fix Pack 4' den IBM MQ , IBM MQ uygulamalarını çalıştırmak için gerekli kitaplıkları içeren yeniden dağıtılabilir istemcileri sunar. Bu kitaplıklar, istemci uygulamalarıyla birlikte paketlenebilir ve yeniden dağıtılabilir. Daha fazla bilgi için bkz. Redistributable clients on Windows.

32 bit uygulamalarla kullanılacak kitaplık (.lib) dosyaları ve dll dosyaları *MQ_INSTALLATION_PATH*/`Tools/Lib`içine kurulur. 64 bit uygulamalarla kullanılacak dosyalar *MQ_INSTALLATION_PATH*/`Tools/Lib64`içinde kurulur. *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Müşteri

```
cl -MD imqsput.cpp /Feimqsputc.exe imqb23vn.lib imqc23vn.lib
```

Sunucu

```
cl -MD imqsput.cpp /Feimqsput.exe imqb23vn.lib imqs23vn.lib
```

Evrensel C yürütme ortamının kurulması

Windows 8.1 ya da Windows Server 2012 R2kullanıyorsanız, Microsoftadresinden evrensel C çalıştırma zamanı güncellemesini (Universal CRT) kurmanız gerekir. Bu çalıştırma zamanı, Windows 10 ve Windows Server 2016 'nın bir parçası olarak dahil edilmiştir.

Evrensel CRT güncellemesi Microsoft update KB3118401(güncelle). `C:\Windows\System32` dizininizde `ucrtbase.dll` adlı bir dosyayı arayarak bu güncelleştirmeye sahip olup olmadığınızı kontrol edebilirsiniz. Yoksa, güncelleştirmeyi şu Microsoft sayfasından yükleyebilirsiniz: https://www.catalog.update.microsoft.com/Search.aspx?q=kb3118401.

Bir IBM MQ programını ya da Microsoft Visual Studio 2017komutunu kullanarak derlediğiniz bir programı çalıştırma zamanı kurulu olmadan çalıştırma girişimi, aşağıdaki hata gibi hatalarla sonuçlanır:

```
The program can't start because api-ms-win-crt-runtime-|1-1-0.dll
is missing from your computer. Try reinstalling the program to
fix this problem.
```

Microsoft Visual Studio 2012 programları için çalıştırma zamanları sağlanması

Bir IBM MQ programını Microsoft Visual Studio 2012komutunu kullanarak derlediyseniz, IBM MQ kuruluş programının Microsoft Visual Studio 2012 C/C++ çalıştırma zamanlarını kurmadığını unutmayın. Önceki IBM MQ sürümünüz aynı bilgisayara kurulduysa, Microsoft Visual Studio 2012 yürütme ortamları o kuruluştan kullanılabilir.

Ancak, Microsoft Visual Studio 2012 kullanılarak oluşturulmuş bir program kullanıyorsanız ve IBM MQ ürününün önceki bir sürümü kurulu değilse, aşağıdaki işlemlerden birini gerçekleştirmeniz gerekir:

- **`Microsoft Visual C++ Redistributable for VisualStudio 2017 (32 and 64-bit versions)`** dosyasını Microsoftsitesinden yükleyin ve kurun.
- Programınızı Microsoft Visual Studio 2017ya da çalıştırma zamanının kurulu olduğu başka bir Microsoft Visual Studio düzeyiyle yeniden derleyin.

Microsoft Visual Studio 2015 derleyicisi kullanılarak oluşturulan C++ istemci kitaplıkları

IBM MQ , Microsoft Visual Studio 2015 C++ derleyicisi ve Microsoft Visual Studio 2017 C++ derleyicisiyle oluşturulan C++ istemcisi kitaplıklarını sağlar.

IBM MQ C++ kitaplıklarının 32 bit ve 64 bit sürümleri sağlanır. 32 bit kitaplıklar `bin\vs2015` klasörü altına kurulur ve 64 bit kitaplıklar `bin64\vs2015` klasörlerinin altına kurulur.

Varsayılan olarak IBM MQ , Microsoft Visual Studio 2017 kitaplıklarını kullanacak şekilde yapılandırılır. Microsoft Visual Studio 2015 kitaplıklarını kullanmak için, IBM MQürününü kurmadan önce ya da **`setmqenv`** ya da **`setmqinst`** komutunu kullanmadan önce `MQ_PREFIX_VS_LIBRARIES` ortam değişkenini `MQ_PREFIX_VS_LIBRARIES=vs2015` olarak ayarlamanız gerekir.

IBM MQ C++ kitaplıklarını farklı şekilde kullanma

IBM MQ , farklı şekilde adlandırılan bazı ek C++ istemcisi kitaplıkları sağlar. Bu kitaplıklar Microsoft Visual Studio 2015 ve Microsoft Visual Studio 2017 C++ derleyicileriyle oluşturulur. Bu kitaplıklar, Microsoft Visual Studio 2017 C++ derleyicisiyle oluşturulan var olan C++ kitaplıklarına ek olarak sağlanır. Bu ek IBM MQ C++ kitaplıkları farklı adlara sahip olduğundan, IBM MQ C++ kullanılarak oluşturulan ve aynı bilgisayarda ürünün Microsoft Visual Studio 2017 ve önceki sürümleriyle derlenen IBM MQ C++ uygulamalarını çalıştırabilirsiniz.

Ek Microsoft Visual Studio 2017 kitaplıkları aşağıdaki adlara sahip:

- `imqb23vnvs2017.dll`
- `imqc23vnvs2017.dll`
- `imqs23vnvs2017.dll`

- `imqx23vnvs2017.dll`

Ek Microsoft Visual Studio 2015 kitaplıkları aşağıdaki adlara sahip:

- `imqb23vnvs2015.dll`
- `imqc23vnvs2015.dll`
- `imqs23vnvs2015.dll`
- `imqx23vnvs2015.dll`

Bu kitaplıkların hem 32 bit, hem de 64 bit sürümleri sağlanır. 32 bit kitaplıklar `bin` klasörü altına kurulur ve 64 bit kitaplıklar `bin64` klasörü altına kurulur. İlgili içe aktarma kitaplıkları `Tools\lib` ve `Tools\lib64` dizinlerinin altına kurulur.

Uygulamanız `imq*vs2015.lib` dosyalarını kullanıyorsa, bunu Microsoft Visual Studio 2015 derleyicisini kullanarak derlemelisiniz. Microsoft Visual Studio 2015ile derlenen IBM MQ C++ uygulamalarını ya da aynı bilgisayarda ürünün önceki bir sürümüyle derlenen uygulamaları çalıştırmak için, PATH ortam değişkeninin öneki aşağıdaki örneklerde gösterildiği gibi olmalıdır:

- 32 bit uygulamalar için:

```
SET PATH=installation folder\bin\vs2015;%PATH%
```

- 64 bit uygulamalar için:

```
SET PATH=installation folder\bin64\vs2015;%PATH%
```

İlgili kavramlar

Windows: IBM MQ 8.0 içinden yapılan değişiklikler

z/OS z/OS Batch, RRS Batch ve CICS üzerinde C++ programları oluşturma

Toplu iş, RRS toplu iş ya da CICS ortamları için z/OS üzerinde IBM MQ C++ programları oluşturun ve örnek programları çalıştırın.

IBM MQ for z/OS ' in desteklediği üç ortam için C++ programları yazabilirsiniz:

- Toplu
- RRS toplu işi
- CICS

Derle, ön bağlantı oluştur ve bağla

C++ kaynak kodunuzu derleyerek, önceden bağlayarak ve bağlantı düzenleyerek bir z/OS uygulaması yaratın.

IBM MQ C++ for z/OS , IBM C++ for z/OS dili için z/OS DLL ' leri olarak gerçekleştirilir. DLL ' leri kullanarak, sağlanan tanımlama yan destelerini ön bağlantı sırasında derleyici çıkışıyla birleştirin. Bu, bağlantı çizicinin IBM MQ C++ üye işlevlerine yapılan çağrıları denetlemesini sağlar.

Not: Üç ortamın her biri için üç set yan güverte vardır.

Bir IBM MQ for z/OS C++ uygulaması oluşturmak için JCL yaratın ve çalıştırın. Aşağıdaki yordamı kullanın:

1. Uygulamanız CICSaltında çalışıyorsa, programınızdaki CICS komutlarını çevirmek için CICStarafından sağlanan yordamı kullanın.

   Ayrıca, CICS uygulamaları için aşağıdakileri yapmanız gerekir:

   a. SCSQLOAD kitaplığını DFHRPL birleştirmesine ekleyin.

   b. CSQCAT1 CEDA grubunu, SCSQPROC kitaplığındaki IMQ4B100 üyesini kullanarak tanımlayın.

c. CSQCAT1' i kurun.

2. Nesne kodu üretmek için programı derleyin. Derlemeye ilişkin JCL, ürün verileri tanımlama dosyalarını derleyicinin kullanımına sağlayan deyimleri içermelidir. Veri tanımlamaları aşağıdaki IBM MQ for z/OS kitaplıklarında sağlanır:

   - **thlqual**.SCSQC370
   - **thlqual**.SCSQHPPS

   `/cxx` derleyici seçeneğini belirttiğinizden emin olun.

   Not: thlqual adı, z/OSüzerindeki IBM MQ kuruluş kitaplığının üst düzey niteleyicidir.

3. **thlqual**.SCSQDEFS içinde sağlanan aşağıdaki tanımlama yan desteleri de içinde olmak üzere, adım “2” sayfa 531içinde oluşturulan nesne kodunu önceden bağlayın:

   a. Toplu iş için imqs23dm ve imqb23dm

   b. RRS toplu işi için imqs23dr ve imqb23dr

   c. CICS için imqs23dc ve imqb23dc

   Bunlar karşılık gelen DLL ' lerdir.

   a. Toplu iş için imqs23im ve imqb23im

   b. RRS toplu işi için imqs23ir ve imqb23ir

   c. CICS için imqs23ic ve imqb23ic

4. “3” sayfa 531. adımda yaratılan nesne kodunu düzenleyin, bir yükleme modülü üretin ve bunu uygulama yükleme kitaplığınızda saklayın.

Toplu iş ya da RRS toplu iş programlarını çalıştırmak için, STEPLIB ya da JOBLIB veri kümesi birleşiminde **thlqual**.SCSQAUTH ve **thlqual**.SCSQLOAD kitaplıklarını ekleyin.

Bir CICS programını çalıştırmak için öncelikle sistem yöneticinizden programı IBM MQ programı ve hareketi olarak CICS olarak tanımlamasını isteyin. O zaman her zamanki gibi çalıştırabilirsin.

Örnek programları çalıştır

Programlar “C++ örnek programları” sayfa 508içinde açıklanmıştır.

Örnek uygulamalar yalnızca kaynak biçimde sağlanır. Dosyalar şunlardır:

Çizelge 75. z/OS örnek program dosyaları

Örnek	Kaynak program (thlqual.SCSQCPPSkitaplığında)	JCL (thlqual.SCSQPROCkitaplığında)
MERHABA DüNYA	imqwrld	imqwrldr
GİRİŞ	imqsput	imqsputr
SGET	imqsget	imqsgetr

Örnekleri çalıştırmak için, bunları herhangi bir C++ programında olduğu gibi derleyin ve bağlayın (bkz. “z/OS Batch, RRS Batch ve CICS üzerinde C++ programları oluşturma” sayfa 530). Bir toplu iş oluşturmak ve çalıştırmak için sağlanan JCL ' yi kullanın. Başlangıçta, JCL 'nin içerdiği açıklamanın ardından JCL' yi uyarlamanız gerekir.

z/OS z/OS UNIX System Services üzerinde C++ programları oluşturma

z/OS UNIX System Services (z/OS UNIX) üzerinde IBM MQ C++ programları oluşturun.

z/OS UNIX kabuğu altında bir uygulama oluşturmak için derleyiciye IBM MQ içerme dosyaları (`thlqual.SCSQC370` ve `hlqual.SCSQHPPS` içinde bulunur) için erişim vermeniz ve iki DLL yan destesi (`thlqual.SCSQDEFS` içinde bulunur) ile bağlantı kurmanız gerekir. Yürütme sırasında, uygulamanın IBM

MQ veri kümelerine `thlqual.SCSQLOAD`, `thlqual.SCSQAUTH`ve `thlqual.SCSQANLE` gibi dile özgü veri kümelerinden birine erişmesi gerekir.[6].

Derleniyor

1. Örneği, TSO **oput** komutunu kullanarak dosya sistemine kopyalayın ya da FTP kullanın. Bu örneğin geri kalanı, örneği `/u/fred/sample`adlı bir dizine kopyaladığınızı ve `imqwrld.cpp`olarak adlandırdığınızı varsayar.
2. z/OS UNIX kabuğunda oturum açın ve örneği yerleştirdiğiniz dizine geçin.
3. C++ derleyicisini, DLL kenar destesi ve .cpp dosyalarını giriş olarak kabul edecek şekilde ayarlayın:

```
/u/fred/sample:> export _CXX_EXTRA_ARGS=1
/u/fred/sample:> export _CXX_CXXSUFFIX="cpp"
```

4. Örnek programı derleyin ve bağlayın. Aşağıdaki komut, programı toplu iş yan taraflarıyla bağlar; bunun yerine RRS toplu iş yan tarafları kullanılabilir. \ karakteri, komutu birden çok satıra bölmek için kullanılır. Bu karakteri girmeyin; komutu tek bir satır olarak girin:

```
/u/fred/sample:> c++ -o imqwrld -I "//'thlqual.SCSQC370'" \
-I "//'thlqual.SCSQHPPS'" imqwrld.cpp \
"//'thlqual.SCSQDEFS(IMQS23DM)'" "//'thlqual.SCSQDEFS(IMQB23DM)'"
```

TSO **oput** komutuyla ilgili daha fazla bilgi için z/OS UNIX Command Referenceadlı yayına bakın.

C++ programları oluşturmayı kolaylaştırmak için Make yardımcı programını da kullanabilirsiniz. HELLO WORLD C++ örnek programını oluşturmak için örnek bir makefile. Derleme ve bağlama aşamalarını ayırır. Ürünü çalıştırmadan önce ortamı “3” sayfa 532 . adımda olduğu gibi ayarlayın.

```
flags = -I "//'thlqual.SCSQC370'" -I "//'thlqual.SCSQHPPS'"
decks = "//'thlqual.SCSQDEFS(IMQS23DM)'" "//'thlqual.SCSQDEFS(IMQB23DM)'"

imqwrld: imqwrld.o
   c++ -o imqwrld imqwrld.o $(decks)

imqwrld.o: imqwrld.cpp
   c++ -c -o imqwrld $(flags) imqwrld.cpp
```

Bkz. z/OS UNIX System Services Programming Tools (Programlama Araçları).

Çalışıyor

1. z/OS UNIX kabuğunda oturum açın ve örneği oluşturmuş olduğunuz dizine geçin.
2. STEPLIB ortam değişkenini IBM MQ veri kümelerini içerecek şekilde ayarlayın:

```
/u/fred/sample:> export STEPLIB=$STEPLIB:thlqual.SCSQLOAD
/u/fred/sample:> export STEPLIB=$STEPLIB:thlqual.SCSQAUTH
/u/fred/sample:> export STEPLIB=$STEPLIB:thlqual.SCSQANLE
```

3. Örneği çalıştırın:

```
/u/fred/sample:> ./imqwrld
```

[6] z/OS UNIX ürününüzü üç ortamdan herhangi birinde çalıştırmak için " Nesne kodunu önceden bağla ' da listelenen herhangi bir kenar destesiyle bağlantı oluşturabilirsiniz, “z/OS Batch, RRS Batch ve CICS üzerinde C++ programları oluşturma” sayfa 530

.NET uygulamalarının geliştirilmesi

IBM MQ classes for .NET , .NET uygulamalarının bir IBM MQ MQI client olarak IBM MQ ' e bağlanmasına ya da bir IBM MQ sunucusuna doğrudan bağlanmasına izin verir.

Microsoft .NET Framework olanağını kullanan ve IBM MQolanaklarından yararlanmak isteyen uygulamalarınız varsa, IBM MQ classes for .NETkomutunu kullanmanız gerekir. Daha fazla bilgi için bkz “kurmaIBM MQ classes for .NET Framework” sayfa 540.

IBM MQ 9.1.1'den IBM MQ , Windows ortamlarındaki uygulamalar için .NET Core ' yi destekler. Daha fazla bilgi için bkz “kurmaIBM MQ classes for .NET” sayfa 534.

IBM MQ 9.1.2'den IBM MQ , Linux ortamlarındaki uygulamalar için .NET Core ' yi destekler.

IBM MQ 9.1.4içinden IBM MQ .NET yönetilen uygulamaları, kümelenmiş kuyruk yöneticileri arasındaki bağlantıları otomatik olarak dengeleyebilirler. Hem .NET Framework hem de .NET Standard kitaplıkları desteklenir. Daha fazla bilgi için bkz. Tek biçimli kümeler hakkında ve Otomatik uygulama dengeleme.

Nesne yönelimli IBM MQ .NET arabirimi, MQI fiillerini kullanmak yerine nesne yöntemlerini kullanması için MQI arabiriminden farklıdır.

Yordamsal IBM MQ uygulama programlama arabirimi, aşağıdaki listedekiler gibi fiillerin etrafına oluşturulmuştur:

```
MQCONN, MQDISC, MQOPEN, MQCLOSE,
MQINQ, MQSET, MQGET, MQPUT, MQSUB
```

Bu fiillerin tümü, parametre olarak, üzerinde çalışacakları IBM MQ nesnesi için bir tanıtıcı alır. .NET nesne odaklı olduğundan, .NET programlama arabirimi bu turu çevirir. Programınız, bu nesneler üzerinde yöntemleri çağırarak işlem yapacağınız bir IBM MQ nesneleri kümesinden oluşur. Programları .NETtarafından desteklenen herhangi bir dilde yazabilirsiniz.

Yordamsal arabirimi kullandığınızda, *Hconn* ' un kuyruk yöneticisi için bir tanıtıcı olduğu MQDISC (*Hconn*, CompCode, Reason) çağrısını kullanarak bir kuyruk yöneticisiyle bağlantınızı kesersiniz.

.NET arabiriminde, kuyruk yöneticisi MQQueueManagersınıfındaki bir nesneyle gösterilir. Bu sınıftaki Disconnect () yöntemini çağırarak kuyruk yöneticisiyle bağlantıyı kesersiniz.

```
// declare an object of type queue manager
MQQueueManager queueManager=new MQQueueManager();
...
// do something...
...
// disconnect from the queue manager
queueManager.Disconnect();
```

IBM MQ classes for .NET , .NET uygulamalarının IBM MQile etkileşimde bulunmasını sağlayan bir sınıf kümesidir. Bunlar, IBM MQ ' in kuyruk yöneticileri, kuyruklar, kanallar ve iletiler gibi uygulamanızın kullandığı çeşitli bileşenlerini gösterir. Bu sınıflara ilişkin ayrıntılar için IBM MQ .NET sınıflarına ve arabirimlerinebakın.

Yazdığınız uygulamaları derlemeden önce bir .NET Framework kurulu olmalıdır. IBM MQ classes for .NET ve .NET Framework ürününü kurmaya ilişkin yönergeler için bkz. “kurmaIBM MQ classes for .NET Framework” sayfa 540.

İlgili kavramlar

Teknik genel bakış

“IBM MQ için uygulama geliştirilmesi” sayfa 5

İleti göndermek ve almak için uygulamalar geliştirebilir ve kuyruk yöneticilerinizi ve ilgili kaynakları yönetebilirsiniz. IBM MQ , birçok farklı dilde ve çerçevede yazılmış uygulamaları destekler.

İlgili görevler

IBM Desteği ile iletişim kurulması

IBM MQ .NET sorunlarının giderilmesi

“IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi” sayfa 1210
IBM MQ için Microsoft Windows Communication Foundation (WCF) özel kanalı, WCF istemcileri ve hizmetleri arasında ileti gönderir ve alır.

“XMS .NET uygulamalarının geliştirilmesi” sayfa 591
IBM MQ Message Service Client (XMS) for .NET (XMS .NET), Java Message Service (JMS) ile aynı arabirim kümesine sahip XMS adlı bir uygulama programlama arabirimi (API) sağlar. API. IBM MQ Message Service Client (XMS) for .NET , herhangi bir .NET uyumlu dil tarafından kullanılabilen, tam olarak yönetilen bir XMSuygulamasını içerir.

Linux Windows kurmaIBM MQ classes for .NET

Örnekler de içinde olmak üzere IBM MQ classes for .NET, Windows ve Linux üzerinde IBM MQ ile birlikte kurulur.

Önkoşullar ve kuruluş

IBM MQ 9.2.0' de .NET Standard kullanılarak oluşturulan IBM MQ .NET istemci kitaplığı Windows ve Linuxsistemlerinde bulunur. IBM MQ classes for .NET Standardprogramını çalıştırmak için Microsoft .NET Coreprogramını kurmanız gerekir. Microsoft.NET Core 3.1 , IBM MQ classes for .NET Standardürününü çalıştırmak için gerekli en düşük sürümdür.

V 9.3.0 V 9.3.0 IBM MQ 9.3.0' den IBM MQ , IBM MQ classes for .NET Standardkullanan .NET 6 uygulamalarını destekler. Bir .NET Core 3.1 uygulaması kullanıyorsanız, bu uygulamayı `csproj` dosyasında küçük bir düzenleme ile çalıştırabilirsiniz; bu işlem `targetframeworkversion` değerini `"net6.0"`olarak ayarlayabilir ve yeniden derleme gerektirmez.

V 9.3.1 IBM MQ 9.3.1 , hedef çerçeve olarak .NET 6 için oluşturulmuş bir IBM MQ .NET istemci kitaplığı sağlar. IBM MQ 9.3.1' den Microsoft .NET 6.0 , hedef çerçeve olarak .NET 6 kullanılarak oluşturulan IBM MQ kitaplıklarını kullanan uygulamaların çalıştırılması için gerekli en düşük sürümdür.

V 9.3.1 IBM MQ 9.3.1içinden, .NET Standard kullanılarak oluşturulan IBM MQ .NET istemci kitaplığı, `netstandard2.0` yeni bir klasör altında bulunur ve hedef çerçeve olarak .NET 6 kullanılarak oluşturulan IBM MQ .NET istemci kitaplığı, Windows üzerinde *MQ_INSTALLATION_PATH*`/bin` altında ve Linuxüzerinde *MQ_INSTALLATION_PATH*`/lib64` altında bulunur.

En son IBM MQ classes for .NET sürümü, varsayılan olarak *Java ve .NET Messaging and Web Services* özelliğinde standart IBM MQ kuruluşunun bir parçası olarak kurulur.

Windows Windowsüzerinde önkoşullar ve kuruluşla ilgili daha fazla bilgi için:

- IBM MQ classes for .NETyazılımını çalıştırmak için önkoşul olan Requirements for IBM MQ classes for .NETbaşlıklı konuya bakın.
- Kuruluş yönergeleri için bkz. IBM MQ sunucusunu Windows üzerine kurma ya da IBM MQ istemcisini Windows sistemlerine kurma .

Linux Linuxüzerinde önkoşullar ve kuruluşla ilgili daha fazla bilgi için:

- IBM MQ classes for .NETyazılımını çalıştırmak için önkoşul olan Requirements for IBM MQ classes for .NETbaşlıklı konuya bakın.
- Rpm kuruluş yönergeleri için bkz. Linux sistemlerine IBM MQ istemcisi kurulması.
- Linux Ubuntuiçin Debian paketlerini kullanma, bkz. Linux sistemlerinde IBM MQ istemcisini kurma.

IBM MQ classes for .NET Standard kitaplığı (`amqmdnetstd.dll`), NuGet havuzundan yüklenebilir. Daha fazla bilgi için bkz “NuGet havuzundan IBM MQ classes for .NET ürününü karşıdan yükleme” sayfa 539.

amqmdnetstd.dll kitaplık

V 9.3.1 IBM MQ 9.3.1' den amqmdnetstd.dll kitaplığını aşağıdaki yerlerde bulabilirsiniz:

Hedef çerçeve olarak .NET Standard 2.0 kullanılarak oluşturulan kitaplık

- Windows Windowssistemlerinde: *MQ_INSTALLATION_PATH*\bin\netstandard2.0.
- Linux Linuxsistemlerinde: *MQ_INSTALLATION_PATH*\lib64\netstandard2.0.

Deprecated Bu kitaplıklar kullanımdan kaldırılmıştır ve IBM ilerideki yayınlarda bu kitaplıkları kaldırmayı planlamaktadır.

Hedef çerçeve olarak .NET 6 kullanılarak oluşturulan kitaplık

- Windows Windowssistemlerinde: *MQ_INSTALLATION_PATH*\bin. Örnek uygulamalar *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/baseiçine kurulur.
- Linux Linuxsistemlerinde: *MQ_INSTALLATION_PATH*\lib64. .NET örnekleri *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/baseiçinde yer almaktadır.

LTS IBM MQ 9.3.0 Long Term Supportiçin amqmdnetstd.dll kitaplığı aşağıdaki yerlerde bulunur:

- Windows Windowssistemlerinde: *MQ_INSTALLATION_PATH*\bin. Örnek uygulamalar *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/baseiçine kurulur.
- Linux Linuxsistemlerinde: *MQ_INSTALLATION_PATH*/lib64 path. .NET örnekleri *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/baseiçinde yer almaktadır.

Uyarı: V 9.3.1 Deprecated IBM MQ 9.3.1olanağından, hedef çerçeve olarak .NET Standard 2.0 kullanılarak oluşturulan IBM MQ .NET istemci kitaplıkları kullanımdan kaldırılmıştır ve bu kitaplıklara gönderme yapan uygulamalar derleme sırasında CS0618 uyarısı verilir.

LTS Stabilized .NET Framework için amqmdnet.dll kitaplığı hala sağlanmıştır, ancak bu kitaplık dengelenmiştir; başka bir değişle, kitaplığa yeni özellikler eklenmez. En son özelliklerden herhangi biri için amqmdnetstd.dll kitaplığına geçmeniz gerekir. Ancak, amqmdnet.dll kitaplığını IBM MQ 9.1 ya da daha sonraki Long Term Support ya da Continuous Delivery yayın düzeylerinde kullanmaya devam edebilirsiniz.

V 9.3.1 Bir .NET Framework uygulaması, IBM MQ 9.3.1 sürümünden daha düşük bir sürümden amqmdnetstd.dll ya da amqmxmsstd.dll kullanılarak derlenirse ve aynı uygulama .NET 6 tabanlı IBM MQ istemci kitaplıkları kullanılarak çalıştırılırsa, .NETtarafından şu FileLoadException tipi kural dışı durum yayınlanır:

```
Kural dışı durum saptandı: System.IO.FileLoadException: Dosya ya da yapıbirimi yüklenemedi
'amqmdnetstd, Version =x.x.x.x, Culture=nötr, PublicKeyToken=23d6cb914eeaac0e' ya da
Bağımlılıklarından biri. Bulunan yapıbiriminin bildirge tanımlaması,
düzenek referansı. (HRESULT kural dışı durumu: 0x80131040)

Dosya adı: ' amqmdnetstd, Sürüm =x.x.x.x, Culture=nötr,
PublicKeyToken=23d6cb914eeaac0e'
```

Bu hatayı çözmek için, *MQ_INSTALLATION_PATH*/bin/netstandard2.0 içinde bulunan kitaplıkların .NET Framework uygulamasının çalıştığı dizine kopyalanması gerekir.

dspmqver DELETE ...

.NET Core bileşenine ilişkin sürüm ve oluşturma bilgilerini görüntülemek için **dspmqver** komutunu kullanabilirsiniz.

IBM MQ classes for .NET Framework ve IBM MQ classes for .NET arasında özellik karşılaştırması (.NET Standard ve .NET 6 kitaplıkları)

Aşağıdaki çizelgede, IBM MQ classes for .NET (.NET Standard ve .NET 6 kitaplıkları) özellikleriyle karşılaştırıldığında IBM MQ classes for .NET Framework özellikleri listelenmektedir.

Çizelge 76. IBM MQ classes for .NET Framework ve IBM MQ classes for .NET arasındaki farklar (.NET Standard ve .NET 6 kitaplıkları)

Özellik	**IBM MQ classes for .NET Framework**	**IBM MQ classes for .NET (.NET Standard ve .NET 6 kitaplıkları)**
Sınıf Adları (API ' ler)	Tüm sınıflar her ağda aynı kalır.	Tüm sınıflar her ağda aynı kalır.
İşletim Sistemi	Windows	Windows Dockerized konteynerleri Linux macOS
`app.config` dosyası (Yeniden dağıtılabilir istemcide İzlemeyi etkinleştirmek için yapılandırma dosyası)	`app.config` dosyası, yeniden dağıtılabilir paket ve bağımsız IBM MQ .NET istemcisi için izlemeyi etkinleştirmek üzere kullanılır. İzleme için kullandığınız değişkenlere (**MQTRACEPATH** ve **MQTRACELEVEL**de içinde olmak üzere) ilişkin ek bilgi için Uygulama yapılandırma dosyasını kullanarak IBM MQ classes for .NET Framework istemcisinin izlenmesi başlıklı konuya bakın.	`app.config` desteklenmiyor. Ortam değişkenlerini kullanın.

Çizelge 76. IBM MQ classes for .NET Framework ve IBM MQ classes for .NET arasındaki farklar (.NET Standard ve .NET 6 kitaplıkları) (devamı var)

Özellik	IBM MQ classes for .NET Framework	IBM MQ classes for .NET (.NET Standard ve .NET 6 kitaplıkları)
Takip edin	IBM MQ' in tam istemci kuruluşu için, IBM MQ classes for .NET Frameworkiçin izlemeyi etkinleştirmek üzere **strmqtrc** komutunu kullanabilirsiniz. Yeniden dağıtılabilen istemciler için, `app.config` kütüğü izlemeyi etkinleştirmek için de kullanılır. Daha fazla bilgi için IBM MQ .NET uygulamalarının izlenmesibaşlıklı konuya bakın. V 9.3.3 IBM MQ 9.3.3' den `mqclient.ini` dosyasını kullanarak ve İzleme kısmına ilişkin uygun özellikleri ayarlayarak izlemeyi etkinleştirebilir ve geçersiz kılabilirsiniz. Ayrıca, `mqclient.ini` dosyasıyla izlemeyi dinamik olarak etkinleştirebilir ve devre dışı bırakabilirsiniz. Daha fazla bilgi için bakınız: Tracing IBM MQ .NET applications with mqclient.ini.	**MQDOTNET_TRACE_ON** ortam değişkeni, yeniden dağıtılabilen istemciler için izlemeyi etkinleştirmek üzere kullanılır. 0 'a eşit ve 0 'dan küçük değerler izlemeyi etkinleştirmez. 1 değeri, varsayılan düzey izlemeyi etkinleştirir. 1 'den büyük bir değer, ayrıntılı izlemeyi etkinleştirir. Bu ortam değişkeninin dizgi gibi başka bir değere ayarlanması izlemeyi etkinleştirmez. Ortam değişkenlerini kullanarak IBM MQ .NET uygulamalarının izlenmesibaşlıklı konuya bakın. **MQDOTNET_TRACE_ON** ortam değişkeni, IBM MQ izleme dizininin kullanılabilir olup olmadığını denetler. İzleme dizini kullanılabiliyorsa, izleme dosyası izleme dizininde oluşturulur. Ancak, IBM MQ kurulu değilse, izleme dosyası yürürlükteki çalışma dizinine kopyalanır. IBM MQ classes for .NET Frameworkiçin kullanılan **MQERRORPATH**, **MQLOGLEVEL**, **MQSERVER**gibi diğer ortam değişkenleri de kullanılabilir ve aynı şekilde çalışabilir. V 9.3.3 IBM MQ 9.3.3' den `mqclient.ini` dosyasını kullanarak ve İzleme kısmına ilişkin uygun özellikleri ayarlayarak izlemeyi etkinleştirebilir ve geçersiz kılabilirsiniz. Ayrıca, `mqclient.ini` dosyasıyla izlemeyi dinamik olarak etkinleştirebilir ve devre dışı bırakabilirsiniz. Daha fazla bilgi için bakınız: Tracing IBM MQ .NET applications with mqclient.ini.
İletim Kipleri	Yönetilen, Yönetilmeyen ve Bağ Tanımları	Yönetilen

Çizelge 76. IBM MQ classes for .NET Framework ve IBM MQ classes for .NET arasındaki farklar (.NET Standard ve .NET 6 kitaplıkları) (devamı var)

Özellik	IBM MQ classes for .NET Framework	IBM MQ classes for .NET (.NET Standard ve .NET 6 kitaplıkları)
TLS	Windows anahtar deposu, sertifikaları saklamak için kullanılır.	Windows Windowsüzerinde, sertifikaları saklamak için anahtar deposu kullanılmalıdır. İzin verilen değerler şunlardır: *USER ya da *SYSTEM. Girişe dayalı olarak, IBM MQ .NET istemcisi yürürlükteki kullanıcının Windows anahtar deposuna ya da sistem genişliğine bakar. Linux Linuxüzerinde, sertifikaları kurmak için X509Store sınıfını kullanmanız ve .NET Core sertifikalarını aşağıdaki konuma kurmanız önerilir: `".dotnet/corefx/cryptography/x509stores"`.
CCDT	Desteklenenler:	Desteklenir ve CCDT yolunun ayarları .NET Framework sınıflarıyla aynıdır.
İstemci otomatik yeniden bağlanma	Desteklenenler:	Desteklenenler:
Dağıtılmış hareketler	Desteklenenler:	Desteklenmiyor
Dinamik bağlantılı kitaplıkların (dll) genel derleme önbelleğine (GAC) kurulması	Dll, IBM MQ kuruluşunun bir parçası olarak GAC ' ye kurulur.	Dll, IBM MQ kuruluşunun bir parçası olarak GAC ' ye kurulmaz.

Not: Windows Windows güvenlik tanıtıcıları (SID):

Etki alanı düzeyi kimlik doğrulaması, IBM MQ classes for .NET (.NET Standard ve .NET 6 kitaplıkları) için desteklenmez. Oturum açan kullanıcı kimliği kimlik doğrulaması için kullanılır.

macOS üzerinde IBM MQ .NET Core uygulamalarının geliştirilmesi

macOS

IBM MQ .NET Core uygulamaları macOSüzerinde geliştirilebilir.

IBM MQ .NET kitaplıkları macOS araç takımıyla birlikte paketlenmez; bu nedenle bunları bir Windows ya da Linux IBM MQ istemcisinden macOS' e kopyalamanız gerekir. Daha sonra macOSüzerinde IBM MQ .NET Core uygulamaları geliştirmek için bu kitaplıkları kullanabilirsiniz.

Geliştirildikten sonra, bu uygulamalar Windows ya da Linux ortamlarında desteklenebilir.

İlgili kavramlar

“kurmaIBM MQ classes for .NET Framework” sayfa 540
Örnekler de içinde olmak üzere IBM MQ classes for .NET Framework, IBM MQile kurulur. Windowsüzerinde Microsoft.NET Framework önkoşulu vardır.

“kurmaIBM MQ classes for XMS .NET” sayfa 595

Örnekler de içinde olmak üzere IBM MQ classes for XMS .NET, Windows ve Linuxüzerinde IBM MQ ile birlikte kurulur.

Linux Windows NuGet havuzundan IBM MQ classes for .NET ürününü karşıdan yükleme

IBM MQ classes for .NET , .NET Developers tarafından kolayca kullanılabilmesi için NuGet havuzundan karşıdan yüklenebilir.

Bu görev hakkında

NuGet , .NETda dahil olmak üzere Microsoft geliştirme platformları için paket yöneticisidir. NuGet istemci araçları, paket üretme ve kullanma yeteneği sağlar. NuGet paketi, derlenmiş kod (DLL), bu kodla ilgili diğer dosyaları ve paketin sürüm numarası gibi bilgileri içeren açıklayıcı bir bildirgeyi içeren `.nupkg` uzantılı tek bir sıkıştırılmış dosyadır.

`amqmdnetstd.dll` kitaplığını içeren `IBMMQDotnetClient` NuGet paketini, tüm paket yazarları ve tüketiciler tarafından kullanılan merkezi paket havuzu olan NuGet Gallery 'den yükleyebilirsiniz.

Not: V 9.3.1 IBM MQ 9.3.1' den NuGet paketi, hedef çerçeve olarak .NET Standard 2.0 ve .NET 6 kullanılarak oluşturulan kitaplıkları içerir. .NET Standard 2.0 kitaplıkları, `netstandard2.0` klasörü altında bulunur ve .NET 6 kitaplıkları `net6.0` klasörü altında bulunur.

`IBMMQDotnetClient` paketini karşıdan yüklemenin üç yolu vardır:

- Microsoft Visual Studiokomutunu kullanarak. NuGet , Microsoft Visual Studio uzantısı olarak dağıtılır. Microsoft Visual Studio 2012' dan NuGet varsayılan olarak önceden kurulur.
- NuGet Package Manager ya da .NET CLI kullanılarak komut satırından.
- Bir web tarayıcısı kullanarak.

Yeniden dağıtılabilir pakete gelince, **MQDOTNET_TRACE_ON**ortam değişkenini kullanarak izlemeyi etkinleştirebilirsiniz.

Yordam

- Microsoft Visual Studioiçindeki Package Manager UI 'sini kullanarak `IBMMQDotnetClient` paketini karşıdan yüklemek için aşağıdaki adımları tamamlayın:
  a) .NET projesini farenin sağ düğmesiyle tıklatın ve **Nuget Packages**öğesini seçin.
  b) **Göz At** etiketini tıklatın ve `"IBMMQDotnetClient"`için arama yapın.
  c) Paketi seçin ve **Kur**düğmesini tıklatın.
  Kuruluş sırasında Package Manager, konsol deyimleri biçiminde aşama bilgileri sağlar.
- `IBMMQDotnetClient` paketini komut satırından yüklemek için aşağıdaki seçeneklerden birini belirleyin:
  - NuGet Package Manager 'ı kullanarak aşağıdaki komutu girin:

    ```
    Install-Package IBMMQDotnetClient -Version 9.1.4.0
    ```

    Kuruluş sırasında Package Manager, konsol deyimleri biçiminde aşama bilgileri sağlar. Çıkışı bir günlük dosyasına yeniden yönlendirebilirsiniz.
  - .NET CLI ' yı kullanarak şu komutu girin:

    ```
    dotnet add package IBMMQDotnetClient --version 9.1.4
    ```
- Bir web tarayıcısı kullanarak, `IBMMQDotnetClient` paketini https://www.nuget.org/packages/IBMMQDotnetClientadresinden yükleyin.

İlgili kavramlar

IBM MQ Client for .NET lisans bilgileri

İlgili görevler

“IBM MQ classes for XMS .NET dosyasını NuGet havuzundan yükleme” sayfa 598
IBM MQ classes for XMS .NET , .NET Developers tarafından kolayca kullanılabilmesi için NuGet havuzundan karşıdan yüklenebilir.

Windows kurmaIBM MQ classes for .NET Framework

Örnekler de içinde olmak üzere IBM MQ classes for .NET Framework, IBM MQile kurulur. Windowsüzerinde Microsoft.NET Framework önkoşulu vardır.

En son IBM MQ classes for .NET Framework sürümü, *Java ve .NET Messaging and Web Services* özelliğinde standart IBM MQ kuruluşunun bir parçası olarak varsayılan olarak kurulur. Kuruluş yönergeleri için bkz. IBM MQ sunucusunu Windows üzerine kurma ya da IBM MQ istemcisini Windows sistemlerine kurma.

V 9.3.0 V 9.3.0 IBM MQ 9.3.0'den IBM MQ classes for .NET Framework ' yi çalıştırmak için Microsoft.NET Framework V4.7.2 ya da sonraki bir sürümünü kurmanız gerekir. Bu, gerekli minimum sürümün V4.6.2olduğu IBM MQ 9.2 sürümünden bir değişikliktir.

V 9.3.0 V 9.3.0 Microsoft.NET Framework V3.5 ile derlenen var olan uygulamalar, uygulamanın `app.config` dosyasına aşağıdaki etiket eklenerek yeniden derlenmeden çalıştırılabilir:

```
<configuration>
 <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2"/>
 </startup>
</configuration>
```

Not: IBM MQkurulmadan önce Microsoft .NET Framework V4.7.2 ya da üstü kurulmazsa, IBM MQ ürün kuruluşu hatasız devam eder, ancak IBM MQ classes for .NET kullanılamaz. .NET Framework , IBM MQkurulduktan sonra kuruluysa, IBM MQ.NET yapıbirimleri *WMQInstallDir*`\bin\amqiRegisterdotNet.cmd` komut dosyası çalıştırılarak kaydedilmelidir; burada *WMQInstallDir* , IBM MQ programının kurulu olduğu dizindir. Bu komut dosyası, Global Assembly Cache (GAC) içindeki gerekli düzenekleri kurar. Yapılan işlemleri kaydeden bir `amqi*.log` dosyaları kümesi, %TEMP% dizininde oluşturulur. .NET , daha önceki bir sürümden (örneğin, .NET V3.5) V4.7.2 ya da daha sonraki bir sürüme yükseltildiyse, `amqiRegisterdotNet.cmd` komut dosyasını yeniden çalıştırmanız gerekmez.

Birden çok kuruluş ortamında, IBM MQ classes for .NET ürününü destek paketi olarak önceden kurduysanız, önce destek paketini kaldırmadan IBM MQ kuramazsınız. IBM MQ ile kurulan IBM MQ classes for .NET özelliği, destek paketiyle aynı işlevleri içerir.

Kaynak dosyalar da içinde olmak üzere örnek uygulamalar da sağlanır; bkz. “.NET için örnek uygulamalar” sayfa 541.

Microsoft WCF with .NETiçin IBM MQ özel kanalını kullanma hakkında bilgi için bkz. “IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi” sayfa 1210

İlgili kavramlar

“kurmaIBM MQ classes for .NET” sayfa 534
Örnekler de içinde olmak üzere IBM MQ classes for .NET, Windows ve Linux üzerinde IBM MQ ile birlikte kurulur.

İlgili görevler

IBM MQ .NET uygulamalarını izleme

IBM MQ classes for .NET ' in bir kuyruk yöneticisine bağlanmasına ilişkin seçenekler

IBM MQ classes for .NET ' i bir kuyruk yöneticisine bağlamak için üç kip vardır. Gereksinimlerinize en uygun bağlantı tipini göz önünde bulundurun.

İstemci bağ tanımları bağlantısı

IBM MQ classes for .NET ürününü IBM MQ MQI clientolarak kullanmak için, IBM MQ MQI clientile birlikte IBM MQ sunucu makinesine ya da ayrı bir makineye kurabilirsiniz. İstemci bağ tanımları bağlantısı XA hareketlerini ya da XA olmayan hareketleri kullanabilir

Sunucu bağ tanımları bağlantısı

Sunucu bağ tanımları kipinde kullanıldığında IBM MQ classes for .NET , bir ağ üzerinden iletişim kurmak yerine kuyruk yöneticisi API 'sini kullanır. Bu, IBM MQ uygulamaları için ağ bağlantılarını kullanmaktan daha iyi performans sağlar.

Bağ tanımları bağlantısını kullanmak için, IBM MQ classes for .NET ürününü IBM MQ sunucusuna kurmanız gerekir.

Yönetilen istemci bağlantısı

Bu kipte yapılan bir bağlantı, yerel ya da uzak makinede çalışan bir IBM MQ sunucusuna IBM MQ istemcisi olarak bağlanır.

Bu kipte IBM MQ classes for .NET bağlantısı .NET yönetilen kodunda kalır ve yerel hizmetlere çağrı yapmaz. Yönetilen kodla ilgili daha fazla bilgi için Microsoft belgelerine bakın.

Yönetilen istemcinin kullanılmasına ilişkin bazı sınırlamalar vardır. Bunlar hakkında daha fazla bilgi için bkz. “Yönetilen istemci bağlantıları” sayfa 556.

.NET için örnek uygulamalar

Kendi .NET uygulamalarınızı çalıştırmak için, örnek uygulamalar yerine uygulama adınızı değiştirerek doğrulama programlarına ilişkin yönergeleri kullanın.

Aşağıdaki örnek uygulamalar sağlanır:

- Koyma iletisi uygulaması
- İleti alma uygulaması
- Bir 'merhaba dünya' uygulaması
- Bir yayınlama/abone olma uygulaması
- İleti özelliklerini kullanan bir uygulama

Tüm bu örnek uygulamalar C# dilinde sağlanır ve bazıları C++ ve Visual Basic dilinde de sağlanır. .NETtarafından desteklenen herhangi bir dilde uygulama yazabilirsiniz.

"Put message" programı SPUT (nmqsput.cs, mmqsput.cpp, vmqsput.vb)
Bu program, bir iletinin adlandırılmış bir kuyruğa nasıl konacağını gösterir. Programın üç parametresi vardır:

- Bir kuyruğun adı (gerekli); örneğin, SYSTEM.DEFAULT.LOCAL.QUEUE
- Kuyruk yöneticisinin adı (isteğe bağlı)
- Bir kanalın tanımı (isteğe bağlı); örneğin, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Kuyruk yöneticisi adı verilmezse, kuyruk yöneticisi varsayılan olarak varsayılan yerel kuyruk yöneticisini kullanır. Bir kanal tanımlandıysa, MQSERVER ortam değişkeniyle aynı biçime sahiptir.

"Get message" program SGET (nmqsget.cs, mmqsget.cpp, vmqsget.vb)
Bu program, adı belirlenmiş bir kuyruktan ileti nasıl alacağınızı gösterir. Programın üç parametresi vardır:

- Bir kuyruğun adı (gerekli); örneğin, SYSTEM.DEFAULT.LOCAL.QUEUE
- Kuyruk yöneticisinin adı (isteğe bağlı)
- Bir kanalın tanımı (isteğe bağlı); örneğin, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Kuyruk yöneticisi adı verilmezse, kuyruk yöneticisi varsayılan olarak varsayılan yerel kuyruk yöneticisini kullanır. Bir kanal tanımlandıysa, MQSERVER ortam değişkeniyle aynı biçime sahiptir.

"Merhaba Dünya" programı (nmqwrld.cs, mmqwrld.cpp, vmqwrld.vb)
Bu program, bir iletinin nasıl konacağını ve gönderileceğini gösterir. Programın üç parametresi vardır:

- Bir kuyruğun adı (isteğe bağlı); örneğin, SYSTEM.DEFAULT.LOCAL.QUEUE ya da SYSTEM.DEFAULT.MODEL.QUEUE
- Kuyruk yöneticisinin adı (isteğe bağlı)
- Kanal tanımlaması (isteğe bağlı); örneğin, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Herhangi bir kuyruk adı verilmezse, ad varsayılan olarak SYSTEM.DEFAULT.LOCAL.QUEUE. Kuyruk yöneticisi adı verilmezse, kuyruk yöneticisi varsayılan olarak varsayılan yerel kuyruk yöneticisini kullanır.

"Yayınlama/abone olma" programı (MQPubSubSample.cs)
Bu program, IBM MQ yayınlama/abone olma özelliğini nasıl kullanacağınızı gösterir. Yalnızca C# olarak sağlanır. Programın iki parametresi vardır:

- Kuyruk yöneticisinin adı (isteğe bağlı)
- Kanal tanımı (isteğe bağlı)

"İleti özellikleri" programı (MQMessagePropertiesSample.cs)
Bu program, ileti özelliklerinin nasıl kullanılacağını gösterir. Yalnızca C# olarak sağlanır. Programın iki parametresi vardır:

- Kuyruk yöneticisinin adı (isteğe bağlı)
- Kanal tanımı (isteğe bağlı)

Bu uygulamaları derleyerek ve çalıştırarak kuruluşunuzu doğrulayabilirsiniz.

Kuruluş konumları

Örnek uygulamalar, yazıldığı dile göre aşağıdaki konumlara kurulur. *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

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

Yönetilen 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

Örnek uygulamaları oluşturma

Örnek uygulamaları oluşturmak için her dil için bir toplu iş dosyası sağlanır.

C#

MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\bldcssamp.bat

bldcssamp.bat dosyası, bu örnek programı oluşturmak için gerekli olan her örnek için bir satır içerir:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib: MQ_INSTALLATION_PATH\bin
/out:nmqwrld.exe nmqwrld.cs
```

Yönetilen C++

MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\bldmcpsamp.bat

bldmcpsamp.bat dosyası, bu örnek programı oluşturmak için gerekli olan her örnek için bir satır içerir:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Bu uygulamaları Microsoft Visual Studio 2003/.NET SDKv1.1üzerinde derlemek istiyorsanız, derleme komutunu değiştirin:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

birlikte

```
cl /clr MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Visual Basic

MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\bldvbsamp.bat

bldvbsamp.bat dosyası, bu örnek programı oluşturmak için gerekli olan her örnek için bir satır içerir:

```
vbc /r:System.dll /r: MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:vmqwrld.exe vmqwrld.vb
```

Microsoft .NET Core ile IBM MQ kullanımı için örnekler

IBM MQ 9.2.0'den IBM MQ , Windows ortamlarındaki IBM MQ .NET uygulamaları için .NET Core ' yi destekler. IBM MQ classes for .NET Standart, örnekler de içinde olmak üzere, standart IBM MQ kuruluşunun bir parçası olarak varsayılan olarak kurulur.

IBM MQ .NET için örnek uygulamalar `&MQINSTALL_PATH&/samp/dotnet/samples/cs/core/base`eiçine kurulur. Örnekleri derlemek için kullanılabilecek bir komut dosyası da sağlanır.

Sağlanan `build.bat` dosyalarını kullanarak örnekleri oluşturabilirsiniz. Windowsüzerinde aşağıdaki konumda her örnek için bir `build.bat` vardır:

- `MQ\tools\dotnet\samples\cs\core\base\SimpleGet`
- `MQ\tools\dotnet\samples\cs\core\base\SimplePut`

Linux IBM MQ 9.2.0' den IBM MQ , Linux ortamlarındaki uygulamalar için Çekirdek 'i de destekler.

IBM MQ ' i Microsoft .NET Coreile kullanma hakkında daha fazla bilgi için bkz. “kurmaIBM MQ classes for .NET” sayfa 534.

Kuyruk yöneticinizin TCP/IP istemci bağlantılarını kabul edecek şekilde yapılandırılması

Bir kuyruk yöneticisini, istemcilerden gelen bağlantı isteklerini kabul edecek şekilde yapılandırın.

Bu görev hakkında

Bu görev, bir kuyruk yöneticisinin TCP/IP istemci bağlantılarını kabul edecek şekilde yapılandırılmasına ilişkin temel adımları açıklar. Bir üretim sistemi için, kuyruk yöneticilerini yapılandırırken güvenlik etkilerini de göz önünde bulundurmanız gerekir.

Yordam

1. Bir sunucu bağlantı kanalı tanımlayın:
   a. Kuyruk yöneticisini başlatın.
   b. NET.CHANNEL:

```
DEF CHL('NET.CHANNEL') CHLTYPE(SVRCONN) TRPTYPE(TCP) MCAUSER(' ') +
DESCR('Sample channel for IBM MQ classes for .NET')
```

   Önemli: Bu örnek, güvenlik etkilerinin dikkate alınmaması nedeniyle, yalnızca bir kum havuzu ortamında kullanılmak üzere tasarlanmıştır. Bir üretim sistemi için TLS ya da bir güvenlik çıkışı kullanmayı düşünün. Ek bilgi için bkz. IBM MQ güvenliğinin sağlanması .
2. Dinleyici başlat:

```
runmqlsr -t tcp [-m qmnqme ] [-p portnum ]
```

   Not: Köşeli ayraç isteğe bağlı parametreleri gösterir; varsayılan kuyruk yöneticisi için *qmname* gerekli değildir ve varsayılan değeri (1414) kullanıyorsanız *portnum* kapı numarası gerekli değildir.

.NET içinde dağıtılmış hareketler

Dağıtılmış hareketler ya da genel hareketler, istemci uygulamalarının tek bir işlemde iki ya da daha fazla ağa bağlı sistemde farklı veri kaynaklarını içermesine olanak sağlar.

Dağıtılmış hareketlerde, bir hareket yöneticisi iki ya da daha fazla kaynak yöneticisi arasındaki işlemi koordine eder ve yönetir.

Hareketler tek aşamalı ya da iki aşamalı kesinleştirme işlemi olabilir. Tek aşamalı kesinleştirme, harekette yalnızca bir kaynak yöneticisinin katıldığı ve iki aşamalı kesinleştirme işleminin harekette yer alan birden çok kaynak yöneticisinin bulunduğu bir süreçtir. İki aşamalı kesinleştirme işleminde, hareket yöneticisi tüm kaynak yöneticilerinin kesinleştirmeye hazır olup olmadığını denetlemek için bir hazırlık çağrısı gönderir. Tüm kaynak yöneticilerinden onay aldığında, kesinleştirme çağrısı yayınlanır. Yoksa, tüm harekette bir geri dönüş olur. Daha fazla ayrıntı için bkz. İşlem yönetimi ve desteği . Kaynak yöneticileri, hareket yöneticilerine harekete katılımlarını bildirmelidir. Kaynak yöneticisi hareket yöneticisine katılımıyla ilgili bilgi verdiğinde, hareket kesinleştirilecek ya da geriye işlenecek olduğunda kaynak yöneticisi hareket yöneticisinden geri çağırmalar alır.

IBM MQ .NET sınıfları, yönetilmeyen ve sunucu bağ tanımları kipi bağlantılarında dağıtılmış hareketleri zaten destekler. Bu kiplerde, IBM MQ .NET sınıfları tüm çağrılarını .NETadına işlem işlemeyi yöneten C genişletilmiş işlem istemcisine devreder.

IBM MQ.NET sınıfları artık IBM MQ .NET Sınıflarının dağıtılmış hareket desteği için System.Transactions ad alanını kullandığı yönetilen kipte dağıtılmış hareketleri desteklemektedir. System.Transactions altyapısı, IBM MQda dahil olmak üzere tüm kaynak yöneticisinde başlatılan işlemleri destekleyerek işlemsel programlamayı basit ve verimli hale getirir. IBM MQ .NET uygulaması, .NET örtük hareket programlama ya da belirtik hareket programlama modelini kullanarak ileti yerleştirip alabilir. Örtük hareketlerde, hareket sınırları, işlemin ne zaman kesinleştirileceğine, geriye işleneceğine (belirtik hareketler için) ya

da tamamlanacağına karar veren uygulama programı tarafından yaratılır. Belirtik hareketlerde, hareketi kesinleştirmek, geriye işlemek ve tamamlamak isteyip istemediğinizi belirtik olarak belirtmeniz gerekir.

IBM MQ.NET , birden çok kaynak yöneticisi arasındaki hareketi koordine eden ve yöneten hareket yöneticisi olarak Microsoft dağıtımlı hareket koordinatörünü (MS DTC) kullanır. Kaynak yöneticisi olarak IBM MQ kullanılır. XA işlemleriyle TLS kullanamayacağınız unutulmamalıdır. CCDT kullanmalısınız. Daha fazla bilgi için TLS kanallarıyla genişletilmiş işlem istemcisini kullanmabaşlıklı konuya bakın.

IBM MQ.NET , X/Open Distributed Transaction Processing (DTP) modelini izler. X/Open Distributed Transaction Processing modeli, bir satıcı konsorsiyumu olan Open Group tarafından önerilen dağıtılmış bir hareket işleme modelidir. Bu model, işlem işleme ve veritabanı etki alanlarındaki ticari satıcıların çoğu arasında bir standarttır. Ticari işlem yönetimi ürünlerinin çoğu X/DTP modelini destekler.

Hareket kipleri

- ".NET tarafından yönetilen kipte dağıtılmış hareketler" sayfa 546
- Yönetilmeyen kip için dağıtılmış hareketler

Çeşitli senaryolarda hareketlerin koordine edilmesi

- Bir bağlantı birden çok harekete katılabilir, ancak herhangi bir zamanda yalnızca bir hareket etkindir.
- Bir hareket sırasında MQQueueManager.Bağlantı kesme çağrısı onurlandırıldı. Bu durumda, hareketin geriye işlemesi istenir.
- Bir hareket sırasında, MQQueue.Close ya da MQTopic.Close çağrısı yerine getirilmiştir. Bu durumda, işlemin geriye işlemesi istenir.
- Hareket sınırları, hareketin ne zaman kesinleştirileceğine, geriye işleneceğine (belirtik hareketler için) ya da tamamlanacağına (örtük hareketler için) karar veren uygulama programı tarafından yaratılır.
- İstemci uygulaması bir kuyruk ya da konu çağrısında Put ya da Get çağrısı yayınlamadan önce beklenmeyen bir hatayla bir hareket sırasında keserse, hareket geriye işlenir ve bir MQException yayınlanır.
- Bir kuyrukta ya da Konu çağrısında Put ya da Get çağrısı sırasında MQCC_FAILED neden kodu döndürülürse, neden koduyla bir MQException yayınlanır ve hareket işlenir. Hareket yöneticisi tarafından önceden bir hazırlama çağrısı verildiyse, IBM MQ .NET işlemi zorla geriye işleyerek hazırlama isteğini döndürür. Bundan sonra, hareket yöneticisi DTC, yürürlükteki çalışma üzerinde yürürlükteki ortam hareketlerindeki tüm kaynak yöneticileriyle geriye işlemeye neden olur.
- Birden çok kaynak yöneticisini içeren bir işlem sırasında, bir ortam nedeni nedeniyle Put ya da Get çağrılarının süresiz olarak askıda kalmasına neden olursa, hareket yöneticisi belirtilen süre kadar bekler. Zaman bittikten sonra, yürürlükteki ortam hareketlerindeki tüm kaynak yöneticileriyle yürürlükteki işlerin geriye işlenmesine neden olur. Bu belirsiz bekleme hazırlama aşamasında gerçekleşirse, hareket yöneticisi kaynağı zamanaşımına uğratabilir ya da kaynağa belirsiz bir çağrı yürütebilir; bu durumda hareket geriye işlenir.
- Hareketleri kullanan uygulamaların SYNC_POINT altına ileti koyması ya da alması gerekir. SYNC_POINT altında olmayan bir hareket bağlamı altında Put ya da Get çağrısı verilirse, çağrı MQRC_UNIT_OF_WORK_NOT_BAŞLATILAN neden koduyla başarısız olur.

Microsoft.NET System.Transactions ad alanı kullanılarak Yönetilen ve Yönetilmeyen Müşteri işlem desteği arasındaki davranış farklılıkları

İç İçe Geçmiş İşlemlerin başka bir TransactionScope içinde TransactionScope vardır

- IBM MQ .NET tam olarak yönetilen istemci, içiçe TransactionScope özelliğini destekler
- IBM MQ .NET yönetilmeyen istemci, içiçe TransactionScope özelliğini desteklemez

System.Transactions içinden bağımlı hareketler

- IBM MQ .NET tam olarak yönetilen istemcisi, System.Transactionstarafından sağlanan bağımlı hareket olanağını destekler.

- IBM MQ .NET yönetilmeyen istemcisi, System.Transactionstarafından sağlanan bağımlı hareket olanağını desteklemez.

Ürün örnekleri

Ürün örnekleri `SimpleXAPut`ve `SimpleXAGet` , `WebSphere MQ\tools\dotnet\samples\cs\base`altında bulunur. Örnekler, SystemTransactions ad alanını kullanarak Distributed Transactions altında MQPUT ve MQGET kullanılmasını gösteren C# uygulamalarıdır. Bu örneklerle ilgili daha fazla bilgi için bkz. "TransactionScope içinde basit koyma ve alma iletileri oluşturma" sayfa 549.

.NET tarafından yönetilen kipte dağıtılmış hareketler

IBM MQ .NET sınıfları, yönetilen kipte dağıtılmış hareket desteği için System.Transactions ad alanını kullanır. Yönetilen kipte MS DTC, bir işlemde kayıtlı tüm sunucular arasında dağıtılmış işlemleri koordine eder ve yönetir.

IBM MQ .NET sınıfları, hareketlerin altyapı tarafından otomatik olarak yönetildiği System.Transactions.TransactionScopesınıfını kullanarak System.Transactions.Transaction sınıfına ve örtük bir programlama modeline dayalı olarak belirtik bir programlama modeli sağlar.

Örtük Hareket

Aşağıdaki kod parçası, bir IBM MQ .NET uygulamasının .NET örtük hareket programlamasını kullanarak bir iletiyi nasıl koyduğunu açıklar.

```
Using (TransactionScope scope = new TransactionScope ())
{
    Q.Put (putMsg,pmo);
    scope.Complete ();
}

Q.close();
qMgr.Disconect();}
```

Örtük hareketin kod akışının açıklaması

Kod, *TransactionScope* ' u oluşturur ve iletiyi kapsam altına koyar. Daha sonra, hareket koordinatörüne hareketin tamamlandığını bildirmek için *Tamamlandı* ' yı çağırır. Hareket koordinatörü şimdi hareketi tamamlamak için *prepare* ve *commit* komutunu verir. Bir sorun saptanırsa, *rollback* çağrılır.

Belirtik İşlem

Aşağıdaki kod, bir IBM MQ .NET uygulamasının .NET belirtik hareket programlama modelini kullanarak iletileri nasıl koyduğunu açıklar.

```
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();
}
```

Belirtik hareketin kod akışının açıklaması

Kod parçası, *CommitableTransaction* sınıfını kullanarak hareket yaratır. Bir iletiyi bu kapsamın altına koyar ve hareketi tamamlamak için belirtik olarak *kesinleştir* ' i çağırır. Herhangi bir sorun varsa, *rollback* çağrılır.

.NET yönetilmeyen kipte dağıtılmış hareketler

IBM MQ.NET sınıfları, örtülü ya da belirtik hareket programlama modeli kullanılarak hareket eşgüdümcüsü olarak genişletilmiş hareket istemcisi ve COM + /MTS kullanılarak yönetilmeyen bağlantıları (istemci) destekler. Yönetilmeyen kipte IBM MQ .NET sınıfları, tüm çağrılarını .NETadına işlem işlemeyi yöneten C genişletilmiş hareket istemcisine devreder.

Hareket işleme, hareket yöneticisinin API denetimi altında genel iş birimini koordine eden bir dış hareket yöneticisi tarafından denetlenir. MQBEGIN, MQCMIT ve MQBACK fiilleri kullanılamıyor. IBM MQ .NET sınıfları bu desteği, yönetilmeyen iletim kipi (C istemcisi) yoluyla gösterir. Bkz. XA uyumlu hareket yöneticilerinin yapılandırılması

MTS, Windows NT üzerinde CICS, Tuxedo ve diğer platformlarda bulunanla aynı özellikleri sağlayacak bir işlem işleme (TP) sistemi olarak geliştirilir. MTS kurulduğunda, Windows NT ' e Microsoft Distributed Transaction eşgüdümcüsü (MSDTC) adı verilen ayrı bir hizmet eklenir. MSDTC , ayrı veri depolarına ya da kaynaklarına yayılan işlemleri koordine eder. Çalışmak için, her veri deposunun kendi özel kaynak yöneticisini gerçekleştirmesini gerektirir.

IBM MQ , DTC XA çağrılarını IBM MQ(X/Open) çağrılarıyla eşlemeyi yönettiği bir arabirim (patentli kaynak yöneticisi arabirimi) uygulayarak MSDTC ile uyumlu olur. IBM MQ bir kaynak yöneticisi rolünü oynar.

COM + gibi bir bileşen bir IBM MQürününe erişim istediğinde, COM genellikle bir işlem gerekip gerekmediğini uygun MTS bağlam nesnesiyle denetler. Bir işlem gerekiyorsa, COM DTC ' yi bilgilendirir ve bu işlem için otomatik olarak bir bütünleşik IBM MQ işlemi başlatır. Daha sonra COM, MQMTS yazılımı aracılığıyla verilerle çalışır ve iletileri gerektiği gibi yerleştirerek ve alarak çalışır. COM ' dan alınan nesne örneği, veriler üzerindeki tüm işlemler bittikten sonra SetComplete ya da SetAbort yöntemini çağırır. Uygulama SetCompletekomutunu yayınladığında, çağrı DTC 'ye uygulamanın işlemi tamamladığını ve DTC' nin iki aşamalı kesinleştirme işlemine devam edebileceğini gösterir. Daha sonra DTC, MQMTS 'ye çağrılar verir; bu da hareketi kesinleştirmek ya da geriye işlemek için IBM MQ ' e çağrıları verir.

Yönetilmeyen istemci kullanılarak bir IBM MQ .NET uygulaması yazılması

COM + bağlamında çalıştırılabilmesi için .NET sınıfının System.EnterpriseServices.ServicedComponent' den edinilmesi gerekir. Hizmet verilen bileşenleri kullanan düzenekler oluşturmaya ilişkin kurallar ve öneriler şunlardır:

Not: Aşağıdaki adımlar yalnızca System.EnterpriseServices kipini kullanıyorsanız geçerlidir.

- COM + ' da başlatılmakta olan sınıf ve yöntemin her ikisi de genel (iç sınıf yok, korunan ya da durağan yöntem yok) olmalıdır.
- Sınıf ve yöntem öznitelikleri: TransactionOption özniteliği, hareketin geçersiz kılınıp kılınmadığını, desteklenip desteklenmediğini ya da gerekli olup olmadığını, sınıfın hareket düzeyini belirler. ExecuteUOW() yöntemindeki AutoComplete özniteliği, işlenemeyen bir kural dışı durum yayınlanmazsa, COM + ' dan hareketi kesinleştirmesini bildirir.
- Bir düzeneğin güçlü bir şekilde adlandırılması: Düzeneğin güçlü bir şekilde adlandırılması ve Küresel Montaj Önbelleğine (Global Assembly Cache; GAC) kaydedilmesi gerekir. Montaj, GAC 'de kaydedildikten sonra COM +' da açıkça veya tembel kayıt ile kaydedilir.
- Bir montajın COM + ' da kaydedilmesi: Düzeneğin COM istemcilerine gösterilmesi için hazırlanması. Daha sonra, Düzenek Kaydı aracını (regasm.exe) kullanarak bir tip kitaplığı oluşturun.

```
regasm UnmanagedToManagedXa.dll
```

- Düzeneği GAC ' ye kaydedin `gacutil /i UnmanagedToManagedXa.dll`.

- .NET Services kuruluş programı aracını (regsvcs.exe) kullanarak düzeneği COM + ' a kaydedin. regasm.exe:

```
Regsvcs /appname:UnmanagedToManagedXa /tlb:UnmanagedToManagedXa.tlb  UnmanagedToManagedXa.dll
```

- Montaj, GAC 'ye yerleştirilir ve daha sonra tembel kayıt ile COM +' a kaydedilir. Kod ilk kez çalıştırıldıktan sonra .NET çerçevesi kaydın icabına bakar.

System.EnterpriseServices modeli ve System.Transactions ve COM + kullanan örnek kod akışı aşağıdaki bölümlerde açıklanmıştır:

System.EnterpriseServices modelini kullanan örnek kod akışı

```
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();
}
```

COM + ile etkileşimler için System.Transactions kullanan örnek kod akışı

```
[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();
}
```

TransactionScope içinde basit koyma ve alma iletileri oluşturma

Ürün örnek C# uygulamaları IBM MQiçinde bulunur. Bu basit uygulamalar, iletilerin bir TransactionScopeiçine konmasını ve bu kapsama girmesini gösterir. Görevin sonunda, bir kuyruğa ya da konuya ileti yerleştirebilecek ve kuyruktan ya da konudan ileti alabileceksiniz.

Başlamadan önce

MSDTC hizmeti çalışıyor olmalı ve XA İşlemleri için etkinleştirilmiş olmalıdır.

Bu görev hakkında

Bu örnek, `SimpleXAPut` ve `SimpleXAGet`adlı basit bir uygulamadır. `SimpleXAPut` ve `SimpleXAGet` programları, IBM MQiçinde bulunan C# uygulamalarıdır. `SimpleXAPut` , SystemTransactions ad alanını kullanarak Distributed Transactions altında MQPUT kullanımı gösterir. `SimpleXAGet` , SystemTransactions ad alanını kullanarak Distributed Transactions altında MQGET kullanımı gösterir.

`SimpleXAPut` , `MQ\tools\dotnet\samples\cs\base` içinde bulunur

Yordam

Uygulamalar, `tools\dotnet\samples\cs\base\bin` içinden komut satırı parametreleriyle çalıştırılabilir

```
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]
```

burada parametreler şunlardır:

`-destinationURI`

Bu kuyruk ya da konu olabilir. Bir kuyruk için `queue://queueName` olarak ve bir konu için `topic://topicName`olarak belirtin.

`-host`

Bu, localhost ya da IP adresi gibi bir anasistem adı olabilir.

`-port`

Kuyruk yöneticisinin çalıştığı kapı.

`-channel`

Kullanılmakta olan bağlantı kanalı. Varsayılan değer `SYSTEM.DEF.SVRCONN`

`-transaction`

Hareket sonucu; örneğin, kesinleştirme ya da geriye işleme.

-mode

İletim kipi; örneğin, yönetilen ya da yönetilmeyen.

-numberOfMsgs

İleti sayısı. Varsayılan değer 1'dir.

Örnek

```
SimpleXAPut -d topic://T01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

```
SimpleXAGet -d queue://Q01 -h localhost -p 2345 -tx rollback -tm unmanaged
```

IBM MQ .NET içinde işlemlerin kurtarılması

Bu bölümde, yönetilen kip kullanılarak IBM MQ .NET XA ' da işlemlerin kurtarılması süreci açıklanmaktadır.

Bu görev hakkında

Dağıtılmış hareket işlemede hareketler başarıyla tamamlanabilir, ancak bir hareketin birçok nedenden ötürü başarısız olabileceği senaryolar olabilir. Bu nedenler, bir sistem hatası, donanım hatası, ağ hatası, yanlış ya da geçersiz veri, uygulama hataları ya da doğal ya da insan kaynaklı olağanüstü durumlar olabilir. Hareket hatalarını önlemek mümkün değildir. Dağıtılmış hareket sistemi bu hataları işleyebilmelidir. Hatalar oluştuğunda bunları saptayıp düzeltebilmelidir. Bu işlem, İşlem Kurtarma olarak bilinir.

Dağıtılmış İşlem İşleminin önemli bir yönü, tamamlanmamış ya da belirsiz hareketleri kurtarmaktır. Belirli bir hareketin İş Birimi bölümü kurtarılıncaya kadar kilitli tutulduğu için kurtarma işleminin çalıştırılması önemlidir. Microsoft.NET , System.Transactions sınıf kitaplığından eksik/belirsiz hareketlerin kurtarılması için seçenek sağlar. Bu kurtarma desteği, Resource Manager ' in işlem günlüklerini tutmasını ve gerekli olduğunda kurtarmayı çalıştırmasını bekler.

Microsoft .NET hareket kurtarma modelinde, Hareket Yöneticisi (System.Transactionsya da Microsoft Distributed Transaction coordinator (MS DTC) ya da her ikisi), hareket kurtarmayı başlatır, koordine eder ve denetler. OLE Tx İletişim Kuralı (Microsoft XA iletişim kuralı) tabanlı Kaynak Yöneticileri, DTC ' yi sürücü, koordinasyon ve denetim için yapılandırma seçeneklerini sağlar. Bunu yapmak için, Kaynak Yöneticilerinin yerel arabirimi kullanarak XA_Switch 'i MS DTC ' ye kaydetmeleri gerekir.

XA_Switch, dağıtımlı hareket koordinatörüne Resource Manager ' ndeki xa_start, xa_end ve xa_recover gibi XA işlevlerinin giriş noktalarını sağlar.

Microsoft Distributed Transaction coordinator (DTC) kullanılarak kurtarma:

Microsoft Distributed Transaction coordinator, iki tür kurtarma süreci sağlar.

Soğuk İyileşme

Bir XA kaynak yöneticisine yönelik bağlantı açıkken hareket yöneticisi işlemi başarısız olursa, soğuk kurtarma gerçekleştirilir. Hareket yöneticisi yeniden başlatıldığında, hareket yöneticisi günlüklerini okur ve XA kaynak yöneticisiyle bağlantıyı yeniden kurar ve kurtarma işlemini başlatır.

Çalışırken Kurtarma

XA kaynak yöneticisi ya da ağ başarısız olduğu için hareket yöneticisi ile XA kaynak yöneticisi arasındaki bağlantı başarısız olduğunda hareket yöneticisi çalışır durumda kalırsa, çalışırken kurtarma gerçekleştirilir. Hatadan sonra, hareket yöneticisi düzenli aralıklarla XA kaynak yöneticisine yeniden bağlanmayı dener. Bağlantı yeniden kurulduğunda, hareket yöneticisi XA kurtarmayı başlatır.

System.Transactions ad alanı, hareket yöneticisi olarak MS DTC ' ye dayalı dağıtımlı hareketlerin yönetilen uygulamasını sağlar. MS DTC ' nin yerel arabirimine benzer özellikler sağlar, ancak tamamen yönetilen bir ortamda. Tek fark hareket kurtarmayla ilgili. System.Transactions , Kaynak Yöneticilerinin

kurtarmayı kendi başlarına gerçekleştirmesini ve ardından Hareket Yöneticileriyle (MS DTC) koordine etmesini bekler. Resource Manager , belirli bir tamamlanmamış işlemin kurtarılmasını istemeli ve İşlem Yöneticisi bu işlemi ve ilgili işlemin gerçek sonucuna göre koordinatları kabul etmiş olmalıdır.

IBM MQ .NET için işlem kurtarma süreci

Bu bölümde, dağıtılmış hareketlerin IBM MQ .NET sınıflarıyla nasıl kurtarılabileceği açıklanmaktadır.

Genel Bakış

Eksik bir hareketi kurtarmak için kurtarma bilgileri gereklidir. Hareket kurtarma bilgileri, kaynak yöneticileri tarafından saklama alanına kaydedilmelidir. IBM MQ .NET sınıfları benzer bir yolu izler. Hareket kurtarma bilgileri, SYSTEM.DOTNET.XARECOVERY.QUEUE.

IBM MQ .NET içinde işlem kurtarma iki aşamalı bir süreçtir:

1. SYSTEM.DOTNET.XARECOVERY.QUEUE.
2. WmqDotnetXAMonitor XA Monitor uygulaması kullanılarak hareketlerin kurtarılması.

SYSTEM.DOTNET.XARECOVERY.QUEUE

SYSTEM.DOTNET.XARECOVERY.QUEUE , tamamlanmamış hareketlere ilişkin hareket kurtarma bilgilerini tutan bir sistem kuyruğudur. Bu kuyruk, bir kuyruk yöneticisi yaratıldığında yaratılır.

Her hareket için, hazırlık aşamasında, kurtarma bilgilerini içeren kalıcı bir ileti SYSTEM.DOTNET.XARECOVERY.QUEUE. Kesinleştirme çağrısı başarılı olursa ileti silinir.

Not: SYSTEM.DOTNET.XARECOVERY.QUEUE (KUYRUK).

WMQDotnetXAMonitor uygulaması

IBM MQ .NET XA Monitor uygulaması WmqDotnetXAMonitor, bir kuyruk yöneticisini izleyen, SYSTEM.DOTNET.XARECOVERY.QUEUE içindeki iletileri işleyen ve tamamlanmamış hareketleri kurtaran .NET tarafından yönetilen bir uygulamadır.

İleti kanalı aracısı (MCA) iletiyi hedef kuyruğa koyamazsa, özgün iletiyi içeren bir kural dışı durum raporu oluşturur ve iletiyi özgün iletide belirtilen yanıt kuyruğuna gönderilecek bir iletim kuyruğuna koyar. (Yanıt kuyruğu MCA ile aynı kuyruk yöneticisiyse, ileti bir iletim kuyruğuna değil, doğrudan o kuyruğa yerleştirilir.)

Aşağıdakiler tamamlanmamış işlemler olarak kabul edilir ve kurtarılabilir:

- Hareket hazırlanıyorsa, ancak zamanaşımı süresi içinde COMMIT tamamlanmamışsa.
- Hareket hazırlanıyorsa, ancak IBM MQ kuyruk yöneticisi kapalı durumda ise.
- İşlem hazırlanıyorsa, ancak İşlem Yöneticisi kapalı durumda ise.

  XA Monitor uygulaması, IBM MQ .NET istemci uygulamanızın çalıştığı sistemden çalıştırılmalıdır. Birden çok sistemde çalışan ve aynı kuyruk yöneticisine bağlanan uygulamalar varsa, WmqDotnetXAMonitor uygulaması tüm sistemlerden çalıştırılmalıdır. Her istemci makinenin uygulamayı kurtarmak için çalışan bir XA Monitor uygulaması yönetim ortamı vardır; ancak, her XA Monitor yönetim ortamı, yürürlükteki XA Monitor 'un yerel MS DTC ' nin eşgüdümlemekte olduğu harekete karşılık gelen iletiyi saptayabilmelidir; böylece, makineyi yeniden listeleyebilir ve tamamlayabilir.

İlgili kavramlar

“IBM MQ .NET için işlem kurtarma kullanım senaryoları” sayfa 552
İşlemlerin kurtarılması gerekebilecek birkaç farklı kullanım senaryosu vardır.

İlgili görevler

“WMQDotnetXAMonitor uygulamasının kullanılması” sayfa 553
IBM MQ .NET istemcisi, tamamlanmamış dağıtılmış hareketleri kurtarmak için kullanabileceğiniz bir XA Monitor uygulaması (WmqDotnetXAMonitor) sağlar. WmqDotnetXAMonitor uygulaması, hareketlerin belirsiz olduğu kuyruk yöneticisiyle bağlantı kurar ve ayarladığınız değiştirgelere dayalı olarak hareketi çözer.

IBM MQ .NET için işlem kurtarma kullanım senaryoları

İşlemlerin kurtarılması gerekebilecek birkaç farklı kullanım senaryosu vardır.

- **IBM MQ Tek DTC ve tek kuyruk yöneticisi yönetim ortamı kullanan uygulama:** Bu durumda, kuyruk yöneticisine bağlandığınızda ve hareket altında İş Birimi 'ni (UoW) çalıştırdığınızda ve hareket başarısız olursa ve tamamlanmazsa, XA Monitor uygulaması hareketi kurtarır ve tamamlar.

  Bu kullanım durumunda, tek bir kuyruk yöneticisi hareketlerle ilişkilendirildiğinden, XA Monitor uygulamasının tek bir eşgörünümü çalışır.
- **Tek DTC ve tek kuyruk yöneticisi eşgörünümünü kullanan birden çok IBM MQ uygulaması:** Bu kullanım durumunda, tek DTC altında birden çok IBM MQ uygulaması vardır ve tümü aynı kuyruk yöneticisine bağlanıyor ve işlemler altında UoW çalıştırılıyor.

  Hareketler başarısız olursa ve tamamlanmazsa, XA Monitor uygulaması bunları kurtarır ve tüm uygulamalarla ilgili işlemleri tamamlar.

  Bu kullanım durumunda, hareketlerde bir kuyruk yöneticisi kullanıldığından, XA Monitor uygulamasının tek bir eşgörünümü çalışır.
- **Birden çok IBM MQ Uygulama, birden çok DTC, farklı kuyruk yöneticisi örneği:** Bu kullanım durumunda, farklı DTC ' ler altında birden çok IBM MQ uygulaması vardır (yani, her uygulama farklı bir makinede çalışır) ve farklı kuyruk yöneticilerine bağlanıyor.

  Hata oluşursa ve işlem tamamlanmazsa, izleyici uygulaması DTC adresini saptamak için iletideki TransactionManageryerini denetler. TransactionManagerWhereabouts değeri, monitörün çalıştığı DTC adresiyle eşleşiyorsa, kurtarma işlemini tamamlar; tersi durumda, DTC ' ye karşılık gelen ileti bulununcaya kadar aramaya devam eder.

  Bu kullanım durumunda, her istemcinin hareketlerde kullanılan kendi kuyruk yöneticisi olduğu için, her istemci (kullanıcı ya da bilgisayar) için çalışan XA Monitor uygulamasının tek bir eşgörünümü olacaktır.
- **Birden çok IBM MQ uygulama, birden çok DTC, birden çok aynı kuyruk yöneticisi örneği:** Bu kullanım durumunda, farklı DTC ' ler altında birden çok IBM MQ uygulaması vardır (her uygulama farklı bir makinede çalışır) ve bunların tümü aynı kuyruk yöneticisine bağlanmaktadır.

  Hata oluşursa ve işlem tamamlanmazsa, izleme programı uygulaması, DTC adresinin ve değerinin, izleme programının çalıştığı DTC ile eşleşip eşleşmediğini denetlemek için iletideki TransactionManageröğesini doğrular. Her iki değer de eşleşirse, kurtarma işlemini tamamlar ya da DTC ' ye karşılık gelen iletiyi buluncaya kadar aramaya devam eder.

  Bu kullanım durumunda, her istemcinin hareketlerde kullanılan kendi kuyruk yöneticisi ilişkilendirmesi olduğu için, XA Monitor uygulamasının her istemci (kullanıcı ya da bilgisayar) için çalışan tek bir eşgörünümü olacaktır.
- **Birden çok IBM MQ Uygulama, tek DTC, farklı kuyruk yöneticisi örneği:** Bu kullanım durumunda, tek bir DTC altında birden çok IBM MQ uygulaması vardır (yani, bir bilgisayarda çalışan birden çok IBM MQ uygulaması vardır) ve farklı kuyruk yöneticilerine bağlanıyor.

  İşlem başarısız olursa ve tamamlanmazsa, izleme uygulaması hareketi kurtarır.

  Bu kullanım senaryounda, her uygulamanın hareketlerde kullanılan kendi kuyruk yöneticisi olduğundan ve her birinin kurtarılması gerekdiğinden, bağlı kuyruk yöneticileri kadar çok sayıda izleyici uygulaması eşgörünümü çalışır.

  Not: XA izleyici uygulaması artalanda çalışmıyorsa, uygulamayı başlatabilirsiniz.

İlgili kavramlar

“IBM MQ .NET için işlem kurtarma süreci” sayfa 551
Bu bölümde, dağıtılmış hareketlerin IBM MQ .NET sınıflarıyla nasıl kurtarılabileceği açıklanmaktadır.

İlgili görevler

“WMQDotnetXAMonitor uygulamasının kullanılması” sayfa 553
IBM MQ .NET istemcisi, tamamlanmamış dağıtılmış hareketleri kurtarmak için kullanabileceğiniz bir XA Monitor uygulaması (WmqDotnetXAMonitor) sağlar. WmqDotnetXAMonitor uygulaması, hareketlerin

belirsiz olduğu kuyruk yöneticisiyle bağlantı kurar ve ayarladığınız değiştirgelere dayalı olarak hareketi çözer.

WMQDotnetXAMonitor uygulamasının kullanılması

IBM MQ .NET istemcisi, tamamlanmamış dağıtılmış hareketleri kurtarmak için kullanabileceğiniz bir XA Monitor uygulaması (WmqDotnetXAMonitor) sağlar. WmqDotnetXAMonitor uygulaması, hareketlerin belirsiz olduğu kuyruk yöneticisiyle bağlantı kurar ve ayarladığınız değiştirgelere dayalı olarak hareketi çözer.

Bu görev hakkında

WMQDotnetXAMonitor uygulaması el ile çalıştırılmalıdır. Her an başlatılabilir. SYSTEM.DOTNET.XARECOVERY.QUEUE ya da IBM MQ .NET sınıfları kullanılarak yazılan uygulamalarla herhangi bir işlem çalışması yapmadan önce arka planda çalışmaya devam edebilirsiniz.

WMQDotnetXAMonitor için değiştirge değerlerini komut satırı aracılığıyla ya da bir uygulama yapılanış kütüğü kullanarak ayarlayabilirsiniz. Uygulama yapılandırma dosyası aracılığıyla sağlanan değerler, komut satırı aracılığıyla ayarlanan değerlerden önceliklidir.

IBM MQ 9.3.0' den önce, WMQDotnetXAMonitor tarafından kurulan bağlantı güvenli olmayan bir bağlantıdır.

V 9.3.0 IBM MQ 9.3.0' den, WMQDotnetXAMonitoriçin ek değiştirgeler ayarlayarak kuyruk yöneticisiyle güvenli bir bağlantı kurma seçeneğiniz vardır.

Yordam

- Bir uygulama yapılandırma dosyasını kullanarak WmqDotNETXAMonitor 'a giriş sağlamak için bkz. “WmqDotNETXAMonitor uygulama yapılanış dosyası ayarları” sayfa 555.
- WMQDotnetXAMonitor uygulamasını komut satırından başlatmak için, gerekli değiştirgelerle birlikte aşağıdaki komutu kullanın:

  Önce IBM MQ 9.3.0:

  ```
  WmqDotnetXAMonitor.exe -m QueueManagerName -n ConnectionName -c ChannelName -i
  ```

  V 9.3.0 IBM MQ 9.3.0' dan:

  ```
  WmqDotnetXAMonitor.exe -m QueueManagerName -n ConnectionName -c ChannelName -i -k SSL Key
  Repository -s Cipher Spec
  ```

  Belirtebileceğiniz parametreler şunlardır:

  – **-m *QueueManagerAdı***
  Kuyruk yöneticisi adı.

  İsteğe Bağlı

  -n *ConnectionName*
  Anasistem (kapı) biçimindeki bağlantı adı. *ConnectionName* birden çok bağlantı adı içerebilir. Virgülle ayrılmış bir listede birden çok bağlantı adı belirtilmelidir; örneğin, `localhost (1414), localhost (1415), localhost (1416)`. WMQDotnetXAMonitor uygulaması, virgülle ayrılmış listede belirtilen bağlantı adlarının her biri için kurtarma işlemini çalıştırır.

  -c *ChannelName*
  Kanal adı.

  -i
  Buluşsal dal tamamlama.

  İsteğe Bağlı

V 9.3.0 **-k *SSL Anahtar Havuzu***

SSL anahtar havuzunun adı. Desteklenen değerler şunlardır:

- *SYSTEM (varsayılan değer budur)
- *KULLANıCı

İsteğe Bağlı

V 9.3.0 **-s *Şifre Belirtimi***

Ayarladığınız CipherSpec , desteklenen sürüm için CipherSpecs ' den biri olmalıdır ve tercihen Windows Grup İlkesinde belirtilenle aynı olabilir. Daha fazla bilgi için bkz “Yönetilen .NET istemcisi için CipherSpec desteği” sayfa 575.

Kuyruk yöneticisiyle güvenli bir bağlantı kurmak için zorunludur.

V 9.3.0 **-dn *SSLPeer Adı***

Eşdüzey kuyruk yöneticisinden sertifikanın ayırt edici adını (DN) denetlemek için kullanılan SSL eşdüzey adı.

İsteğe Bağlı

V 9.3.0 **-cl *Sertifika Etiketi***

Sertifikayı tanıtan etiket adı.

İsteğe Bağlı

V 9.3.0 **-sn *OutboundSNI***

Sunucu Adı Göstergesi (SNI), bir TLS bağlantısı başlatılırken hedef IBM MQ kanal adına mı, yoksa anasistem adına mı ayarlanmalıdır? Bu seçenek için desteklenen değerler şunlardır:

- CHANNEL (varsayılan değer budur)
- ANASİSTEM ADI
- *

Herhangi bir değer ayarlanmazsa, varsayılan değer (CHANNEL) kullanılır.

İsteğe Bağlı

V 9.3.0 **-cr *Sertifika İptal Denetimi***

Sertifikasyon iptal denetiminin yapılıp yapılmayacağını belirler. Bu seçenek için desteklenen değerler şunlardır:

- doğru
- false (bu varsayılan değerdir)

İsteğe Bağlı

V 9.3.0 **-kr *KeyResetSayı***

Şifreleme için kullanılan gizli anahtar yeniden anlaşılmadan önce kanalda gönderilen ve alınan toplam şifrelenmemiş bayt sayısı.

Varsayılan değer olan 0, gizli anahtarların hiçbir zaman yeniden anlaşılmadığını gösterir.

İsteğe Bağlı

WMQDotnetXAMonitor uygulaması aşağıdaki işlemleri gerçekleştirir:

1. SYSTEM.DOTNET.XARECOVERY.QUEUE , 100 saniyelik bir aralıkta.
2. Kuyruk derinliği sıfırdan büyükse, ileti kuyruğuna göz atılır ve iletilerin tamamlanmamış hareket ölçütlerine uyup uymadığını denetler.
3. Bir ileti tamamlanmamış hareket ölçütlerini karşılıyorsa, iletiyi çeker ve işlem kurtarma bilgilerini alır.

4. Kurtarma bilgilerinin yerel Microsoft Distributed Transaction coordinator (MS DTC) ile ilgili olup olmadığını belirler. Böyle bir durumda, WMQDotnetXAMonitor hareketi kurtarmaya devam eder; tersi durumda, sonraki iletiye göz atmak için geri döner.
5. Tamamlanmamış hareketi kurtarmak için kuyruk yöneticisine çağrı yapar.

WmqDotNETXAMonitor uygulama yapılanış dosyası ayarları
Bir uygulama yapılandırma dosyası kullanarak IBM MQ .NET XA Monitor uygulaması WmqDotNETXAMonitor 'a giriş sağlayabilirsiniz. Örnek bir uygulama yapılandırma dosyası IBM MQ .NETile birlikte gönderilir. Bu örnek dosyayı gereksinimlerinize göre değiştirebilirsiniz.

Uygulama yapılandırma dosyası aracılığıyla sağlanan giriş değerleri en yüksek önceliği alır. Komut satırında hem “WMQDotnetXAMonitor uygulamasının kullanılması” sayfa 553 içinde, hem de uygulama yapılandırma dosyasında açıklandığı gibi giriş değerleri sağlıyorsanız, uygulama yapılandırma dosyasındaki değerler öncelikli olur.

IBM MQ 9.3.0öncesi için örnek uygulama yapılandırma dosyası.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<configSections>
<sectionGroup name="IBM.WMQ">
<section name="dnetxa" type="System.Configuration.NameValueFileSectionHandler" />
</sectionGroup>
</configSections>
<IBM.WMQ>
<dnetxa>
<add key="ConnectionName" value=""/>
<add key="ChannelName" value="" />
<add key="QueueManagerName" value="" />
<add key="UserId" value="" />
<add key="SecurityExit" value="" />
<add key="SecurityExitUserData" value = "">
</dnetxa>
</dnetxa>
</configuration>
```

V 9.3.0 IBM MQ 9.3.0' den örnek uygulama yapılandırma dosyası.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
<configSections>
<sectionGroup name="IBM.WMQ">
<section name="dnetxa" type="System.Configuration.NameValueFileSectionHandler" />
</sectionGroup>
</configSections>
<IBM.WMQ>
<dnetxa>
<add key="ConnectionName" value=""/>
<add key="ChannelName" value="" />
<add key="QueueManagerName" value="" />
<add key="UserId" value="" />
<add key="SecurityExit" value="" />
<add key="SecurityExitUserData" value = "">
<add key="SSLKeyRepository" value="" />
<add key="SSLCipherSpec" value="" />
<add key="SSLPeerName" value="" />
<add key="SSLKeyResetCount" value="" />
<add key="SSLCertRevocationCheck" value="" />
<add key="CertificateLabel" value="" />
<add key="OutboundSNI" value="" />
</dnetxa>
</dnetxa>
</configuration>
```

WmqDotNetXAMonitor Uygulama günlüğü
Monitor Application, Monitor 'un ilerleme durumunu ve işlem kurtarma durumunu günlüğe kaydetmek için uygulama dizininde bir günlük dosyası yaratır. Günlük kaydı, kurtarma işleminin yürütüldüğü yürürlükteki kuyruk yöneticisini göstermek için bağlantı adı ve kanal ayrıntılarıyla başlar.

Kurtarma başladıktan sonra, işlem kurtarma iletisinin MessageId , tamamlanmamış işlemin TransactionId ve İşlem Yöneticisi Koordinasyonu uyarınca işlemin gerçek sonucu günlüğe kaydedilir.

Örnek günlük dosyası:

```
Time|ProcessId|ThreadId|WMQ .NET XA Recovery Monitor, Running now for
ConnectionName:xxxx, Time|ProcessId|ThreadId|Channel=xxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Roll back
Time|ProcessId|ThreadId|Recovery Completed for TransactionId= xxxxx
Time|ProcessId|ThreadId|Current QueueDepth = n
Time|ProcessId|ThreadId|Current MessageId = xxxx
Time|ProcessId|ThreadId|Current Incomplete Transaction being recovered = xxxxx
Time|ProcessId|ThreadId|Actual Outcome of the transaction(as per DTC)= Commit/Roll back
Time|ProcessId|ThreadId| Recovery Completed for TransactionId= xxxxx
```

IBM MQ .NET programlarının yazılması ve konuşlandırılması

IBM MQ kuyruklarına erişmek için IBM MQ classes for .NET komutunu kullanmak üzere, .NET tarafından desteklenen herhangi bir dilde programlar yazarak IBM MQ kuyruklarından ileti alır ve ileti alır.

IBM MQ belgeleri yalnızca C#, C++ ve Visual Basic dillerine ilişkin bilgileri içerir.

Bu konu derlemi, IBM MQ sistemleriyle etkileşimde bulunacak uygulamaların yazılmasına yardımcı olacak bilgiler sağlar. Tek tek sınıflara ilişkin ayrıntılar için bkz. IBM MQ .NET sınıfları ve arabirimleri.

Bağlantı farkları

IBM MQ.NET için programınızın kullanmak istediğiniz bağlantı kiplerine bazı bağımlılıkları vardır.

IBM MQ classes for .NET , yönetilen istemci olarak kullanıldığında, bazı özellikler yönetilen istemci tarafından kullanılamadığı için, standart IBM MQ MQI clientile arasında birçok fark vardır.

IBM MQ.NET , bağlantı adı, kanal adı, uyarlama değeri NMQ_MQ_LIB ve özellik MQC.TRANSPORT_PROPERTY.

Yönetilen istemci bağlantıları

IBM MQ classes for .NET , yönetilen istemci olarak kullanıldığında, standart IBM MQ MQI clientile arasında bir dizi fark vardır.

Aşağıdaki özellikler yönetilen bir istemci tarafından kullanılamaz:

- Kanal sıkıştırma
- Kanal çıkışı zincirleme

Bu özellikleri bir yönetilen istemciyle kullanmayı denerseniz, bu bir MQException döndürür. Hata bir bağlantının istemci ucunda saptanırsa, MQRC_ENVIRONMENT_ERROR neden kodunu kullanır. Sunucu sona erdiğinde saptanırsa, sunucu tarafından döndürülen neden kodu kullanılır.

Yönetilmeyen bir istemci için yazılan kanal çıkışları çalışmıyor. Özellikle yönetilen istemci için yeni çıkışlar yazmanız gerekir. İstemci kanal tanımlama çizelgesinde (CCDT) geçersiz kanal çıkışları belirtilmediğini doğrulayın.

Yönetilen kanal çıkışının adı en çok 999 karakter uzunluğunda olabilir. Ancak, kanal çıkış adını belirtmek için CCDT kullanırsanız, bu ad 128 karakterle sınırlıdır.

İletişim yalnızca TCP/IP üzerinden desteklenir.

endmqm komutunu kullanarak bir kuyruk yöneticisini durdurduğunuzda, .NET yönetilen istemcisine yönelik bir sunucu bağlantısı kanalının kapanması, diğer istemcilere yönelik sunucu bağlantısı kanallarından daha uzun sürebilir.

Yönetilen IBM MQ sorun tanılama programlarını kullanmak için *NMQ_MQ_LIB* değiştirgesini `yönetilen` olarak ayarladıysanız, **strmqtrc** komutunun -i, -p, -s, -b ya da -c değiştirgelerinden hiçbiri desteklenmez.

XA hareketlerini kullanan yönetilen bir .NET uygulaması, z/OS kuyruk yöneticisiyle çalışmaz. z/OS kuyruk yöneticisine bağlanmayı deneyen bir yönetilen .NET istemcisi, MQOPEN çağrısında MQRC_UOW_ENLISTMENT_ERROR (mqrc=2354) hatasıyla başarısız oluyor. Ancak, XA hareketlerini kullanan yönetilen bir .NET uygulaması, dağıtılmış kuyruk yöneticisiyle çalışır.

Kullanılacak bağlantı tipini tanımlama

Bağlantı tipi, bağlantı adı, kanal adı, NMQ_MQ_LIB uyarlama değeri ve MQC.TRANSPORT_PROPERTY.

Bağlantı adını aşağıdaki gibi belirtebilirsiniz:

- Bir MQQueueManager oluşturucusunda belirtik olarak:

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- MQQueueManager oluşturucusundaki bir Hashtable girişinde MQC.HOST_NAME_PROPERTY ve isteğe bağlı olarak MQC.PORT_PROPERTY özelliklerini ayarlayarak:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Belirtik MQEnvironment değerleri olarak

```
MQEnvironment.Hostname
```

  `MQEnvironment.Port` (isteğe bağlı).
- MQEnvironment.properties Hashtable içinde MQC.HOST_NAME_PROPERTY ve isteğe bağlı olarak MQC.PORT_PROPERTY özelliklerini ayarlayarak.

Kanal adını aşağıdaki gibi belirtebilirsiniz:

- Bir MQQueueManager oluşturucusunda belirtik olarak:

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- MQQueueManager oluşturucusundaki bir hashtable girişinde MQC.CHANNEL_PROPERTY özelliğini ayarlayarak:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Belirtik bir MQEnvironment değeri olarak

```
MQEnvironment.Channel
```

- MQEnvironment.properties hashtable içinde MQC.CHANNEL_PROPERTY özelliğini ayarlayarak.

İletim özelliğini aşağıdaki gibi belirtebilirsiniz:

- MQQueueManager oluşturucusundaki bir hashtable girişinde MQC.TRANSPORT_PROPERTY özelliğini ayarlayarak:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- MQEnvironment.properties hashtable içinde MQC.TRANSPORT_PROPERTY özelliğini ayarlayarak.

Aşağıdaki değerlerden birini kullanarak istediğiniz bağlantı tipini seçin:

MQC.TRANSPORT_MQSERIES_BINDINGS -sunucu olarak bağlan
MQC.TRANSPORT_MQSERIES_CLIENT -XA istemcisi olmayan bir istemci olarak bağlanma
MQC.TRANSPORT_MQSERIES_XACLIENT -XA istemcisi olarak bağlan
MQC.TRANSPORT_MQSERIES_MANAGED -XA olmayan yönetilen istemci olarak bağlan

NMQ_MQ_LIB uyarlama değerini, aşağıdaki çizelgede gösterildiği gibi bağlantı tipini belirtik olarak seçecek şekilde ayarlayabilirsiniz.

NMQ_MQ_LIB değeri	**Bağlantı tipi**
mqic.dll	XA olmayan bir istemci olarak bağlan
mqicxa.dll	XA istemcisi olarak bağlan
mqm.dll	Sunucu olarak ya da XA olmayan bir istemci olarak bağlan
yönetilen	XA olmayan yönetilen istemci olarak bağlan

Not: mqic32.dll ve mqic32xa.dll değerleri, önceki yayınlarla uyumluluk için mqic.dll ve mqicxa.dll eşanlamlıları olarak kabul edilir. Ancak, mqm.dll ve mqm.pdb , IBM WebSphere MQ 7.1 ' den başlayarak istemci paketinin yalnızca bir parçasıdır.

Ortamınızda kullanılamayan bir bağlantı tipi seçerseniz, örneğin mqic32xa.dll belirtirseniz ve XA desteğine sahip değilseniz, IBM MQ.NET bir kural dışı durum verir.

NMQ_MQ_LIB ' nin "yönetilen" olarak ayarlanması, istemcinin yönetilen IBM MQ sorun tanılama sınamalarını, .NET veri dönüştürme ve diğer yönetilen düşük düzey IBM MQ işlevlerini kullanmasına neden olur.

NMQ_MQ_LIB için diğer tüm değerler, .NET işleminin yönetilmeyen IBM MQ sorun tanılama sınamalarını ve veri dönüştürmelerini ve diğer yönetilmeyen düşük düzeyli IBM MQ işlevlerini kullanmasına neden olur (sistemde bir IBM MQ MQI client ya da sunucu kurulu olduğu varsayılarak).

IBM MQ.NET , bağlantı tipini aşağıdaki gibi seçer:

1. MQC.TRANSPORT_PROPERTY belirtildi, ancak MQC.TRANSPORT_PROPERTY.

   Ancak, MQC.TRANSPORT_PROPERTY - MQC.TRANSPORT_MQSERIES_MANAGED , istemci işleminin yönetilen olarak çalıştığını garanti etmez. Bu ayarda bile, istemci aşağıdaki durumlarda yönetilmez:

   - İşlemdeki başka bir iş parçacığı MQC.TRANSPORT_PROPERTY , MQC.TRANSPORT_MQSERIES_MANAGED.
   - NMQ_MQ_LIB "yönetilen" olarak ayarlanmazsa, sorun tanılama sınamaları, veri dönüştürme ve diğer düşük düzeyli işlevler tam olarak yönetilmez (sistemde bir IBM MQ MQI client ya da sunucu kurulu olduğu varsayılarak).
2. Kanal adı olmadan bir bağlantı adı belirlendiyse ya da bağlantı adı olmadan bir kanal adı belirlendiyse, bir hata oluşur.
3. Hem bir bağlantı adı, hem de bir kanal adı belirlendiyse:
   - NMQ_MQ_LIB mqic32xa.dllolarak ayarlanırsa, bir XA istemcisi olarak bağlanır.
   - NMQ_MQ_LIB yönetilen olarak ayarlanırsa, yönetilen istemci olarak bağlanır.
   - Tersi durumda, XA olmayan bir istemci olarak bağlanır.
4. NMQ_MQ_LIB belirtilirse, NMQ_MQ_LIB değerine göre bağlanır.
5. Bir IBM MQ sunucusu kuruluysa, sunucu olarak bağlanır.
6. Bir IBM MQ MQI client kuruluysa, XA olmayan bir istemci olarak bağlanır.
7. Tersi durumda, yönetilen istemci olarak bağlanır.

Windows IBM MQ .NET proje şablonunun kullanılması

IBM MQ .NET istemcisi, .NET Core uygulamalarınızı geliştirmenize yardımcı olmak için bir proje şablonu kullanma yeteneği sunar.

Başlamadan önce

Sisteminizde Microsoft Visual Studio 2017ya da üstü ve .NET Core 2.1 olmalıdır.

.NET şablonunu

&MQ_INSTALL_ROOT&\tools\dotnet\samples\cs\core\base\ProjectTemplates\IBMMQ.NETClientApp.zip

dizin

&USER_HOME_DIRECTORY&\Documents\&Visual_Studio_Version&\Templates\ProjectTemplates

dizin, burada:

- *& MQ_INSTALL_ROOT* kuruluşunuzun kök dizinidir
- *& USER_HOME_DIRECTORY* sizin ana dizininizdir.

Şablonu almak için Microsoft Visual Studio ' i durdurup yeniden başlatmanız gerekir.

Bu görev hakkında

.NET proje şablonu, uygulamalarınızı geliştirmenize yardımcı olmak için kullanabileceğiniz bazı ortak kodları içerir. Yerleşik kodla IBM MQ kuyruk yöneticisine bağlanabilir ve yerleşik koddaki özellikleri değiştirerek bir koyma ya da alma işlemi gerçekleştirebilirsiniz.

Yordam

1. Microsoft Visual Studio uygulamasını açın.
2. **Dosya**'yı, ardından **Yeni** ' yi ve ardından **Proje**' yi tıklatın.
3. *Yeni proje yarat penceresinde* `IBM MQ .NET Client App (.NET Core)` seçeneğini belirleyin ve **İleri**düğmesini tıklatın.
4. *Yeni projenizi yapılandır* penceresinde, isterseniz projenizin *Proje adını* değiştirin ve .NET projesini yaratmak için **Yarat** düğmesini tıklatın.

   `MQDotnetApp.cs` , proje dosyasıyla birlikte yaratılan dosyadır. Bu dosya, kuyruk yöneticisine bağlanan ve bir koyma ve alma işlemi gerçekleştiren kodu içerir.

   Bağlantı özellikleri varsayılan değerlere ayarlanır:

   - MQC.CONNECTION_NAME_PROPERTY , *localhost (1414)* olarak ayarlandı
   - MQC.CHANNEL_PROPERTY , *DOTNET.SVRCONN* olarak ayarlandı

   Kuyruk *Q1*olarak ayarlanır ve bu özellikleri buna göre değiştirebilirsiniz.
5. Uygulamayı derleyin ve çalıştırın.

İlgili kavramlar

IBM MQ bileşenleri ve özellikleri

.NET uygulama yürütme ortamı-yalnızca Windows

IBM MQ classes for .NET için yapılandırma dosyaları

Bir .NET istemci uygulaması, bir IBM MQ MQI client yapılandırma dosyasını ve yönetilen bağlantı tipini kullanıyorsanız, bir .NET uygulama yapılandırma dosyasını kullanabilir. Uygulama yapılandırma dosyasındaki ayarların önceliği vardır.

İstemci yapılanış kütüğü

Bir IBM MQ classes for .NET istemci uygulaması, bir istemci yapılandırma dosyasını diğer herhangi bir IBM MQ MQI clientile aynı şekilde kullanabilir. Bu dosya genellikle `mqclient.ini`olarak adlandırılır, ancak farklı bir dosya adı belirtebilirsiniz. İstemci yapılandırma dosyası hakkında daha fazla bilgi için bkz. IBM MQ MQI client yapılandırma dosyası, `mqclient.ini`.

Bir IBM MQ MQI client yapılandırma dosyasında yalnızca aşağıdaki öznitelikler IBM MQ classes for .NETile ilgilidir. Diğer öznitelikleri belirtirseniz, bunun bir etkisi olmaz.

Çizelge 77. IBM MQ classes for .NET ile ilgili istemci yapılandırma dosyası öznitelikleri

Kıta	**Öznitelik**
Kanallar	CCSID
Kanallar	ChannelDefinitionDizini
Kanallar	ChannelDefinitionDosyası
Kanallar	ReconDelay
Kanallar	DefRecon
Kanallar	MQReconnectTimeout
Kanallar	ServerConnectionParms
Kanallar	Put1DefaultAlwaysSync
Kanallar	PasswordProtection
ClientExitYolu	ExitsDefaultYolu
ClientExitYolu	ExitsDefaultPath64
MessageBuffer	MaximumSize
MessageBuffer	PurgeTime
MessageBuffer	UpdatePercentage
V 9.3.0 V 9.3.0 Güvenlik	MQIInitialKeyDosyası
V 9.3.0 V 9.3.0 SSL	SSLKeyRepository
V 9.3.0 V 9.3.0 SSL	SSLKeyRepositoryParolası
TCP	ClntRcvBufSize
TCP	ClntSndBufSize
TCP	IPAddressVersion
TCP	KeepAlive

Uygun ortam değişkenini kullanarak bu özniteliklerden herhangi birini geçersiz kılabilirsiniz.

Uygulama yapılandırma dosyası

Yönetilen bağlantı tipiyle çalışıyorsanız, .NET uygulama yapılandırma dosyasını kullanarak IBM MQ istemci yapılandırma dosyasını ve eşdeğer ortam değişkenlerini de geçersiz kılabilirsiniz.

.NET uygulama yapılandırma dosyası ayarları yalnızca yönetilen bağlantı tipiyle çalıştırıldığında yürütülebilir ve diğer bağlantı tipleri için yoksayılır.

.NET uygulama yapılandırma dosyası ve biçimi, .NET çerçevesi içinde genel kullanım için Microsoft tarafından tanımlanır, ancak bu belgede sözü edilen bölüm adları, anahtarlar ve değerler IBM MQ' e özgüdür.

.NET uygulama yapılandırma dosyasının biçimi çeşitli *kısımlardır*. Her bir bölüm bir ya da daha fazla *anahtar*içerir ve her bir anahtarın ilişkili bir *değeri*vardır. Aşağıdaki örnekte, TCP/IP KeepAlive özelliğini denetlemek için bir .NET uygulama yapılandırma dosyasında kullanılan bölümler, anahtarlar ve değerler gösterilmektedir:

```
<configuration>
  <configSections>
    <section name="TCP" type="System.Configuration.NameValueSectionHandler"/>
  </configSections>
  <TCP>
    <add key="KeepAlive" value="true"></add>
  </TCP>
<configuration>
```

.NET uygulama yapılandırma dosyası bölüm adlarında ve anahtarlarında kullanılan anahtar sözcükler, istemci yapılandırma dosyasında tanımlı olan stanzas ve öznitelikler için anahtar sözcüklerle tam olarak eşleşiyor.

<configSections> kısmı, <configuration> öğesinin ilk alt öğesi olmalıdır.

Daha fazla bilgi için Microsoft belgelerinize bakın.

.NET ile kullanılacak örnek C# kod parçası

Bir uygulamanın bir kuyruk yöneticisine bağlandığını, bir iletiyi kuyruğa koyduğunu ve bir yanıt aldığını gösteren bir C# kod parçası.

Aşağıdaki C# kod parçası, üç işlem gerçekleştiren bir uygulamayı gösterir:

1. Kuyruk yöneticisine bağlan
2. SYSTEM.DEFAULT.LOCAL.QUEUE
3. İletiyi geri al

Bağlantı tipinin nasıl değiştirileceğini de gösterir.

```
// ============================================================================
// Licensed Materials - Property of IBM
// 5724-H72
// (c) Copyright IBM Corp. 2003, 2024
// ============================================================================
using System;
using System.Collections;

using IBM.WMQ;

class MQSample
{
  // The type of connection to use, this can be:-
  // MQC.TRANSPORT_MQSERIES_BINDINGS for a server connection.
  // MQC.TRANSPORT_MQSERIES_CLIENT for a non-XA client connection
  // MQC.TRANSPORT_MQSERIES_XACLIENT for an XA client connection
  // MQC.TRANSPORT_MQSERIES_MANAGED for a managed client connection
  const String connectionType = MQC.TRANSPORT_MQSERIES_CLIENT;

  // Define the name of the queue manager to use (applies to all connections)
  const String qManager = "your_Q_manager";

  // Define the name of your host connection (applies to client connections only)
  const String hostName = "your_hostname";

  // Define the name of the channel to use (applies to client connections only)
  const String channel = "your_channelname";



  /// <summary>
  /// Initialise the connection properties for the connection type requested
```

```
/// </summary>
/// <param name="connectionType">One of the MQC.TRANSPORT_MQSERIES_ values</param>
static Hashtable init(String connectionType)
{
  Hashtable connectionProperties = new Hashtable();

  // Add the connection type
  connectionProperties.Add(MQC.TRANSPORT_PROPERTY, connectionType);

  // Set up the rest of the connection properties, based on the
  // connection type requested
  switch(connectionType)
  {
    case MQC.TRANSPORT_MQSERIES_BINDINGS:
      break;
    case MQC.TRANSPORT_MQSERIES_CLIENT:
    case MQC.TRANSPORT_MQSERIES_XACLIENT:
    case MQC.TRANSPORT_MQSERIES_MANAGED:
      connectionProperties.Add(MQC.HOST_NAME_PROPERTY, hostName);
      connectionProperties.Add(MQC.CHANNEL_PROPERTY, channel);
      break;
  }

  return connectionProperties;
}
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static int Main(string[] args)
{
  try
  {
    Hashtable connectionProperties = init(connectionType);

    // Create a connection to the queue manager using the connection
    // properties just defined
    MQQueueManager qMgr = new MQQueueManager(qManager, connectionProperties);

    // Set up the options on the queue we want to open
    int openOptions = MQC.MQOO_INPUT_AS_Q_DEF | MQC.MQOO_OUTPUT;

    // Now specify the queue that we want to open,and the open options
    MQQueue system_default_local_queue =
      qMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", openOptions);

    // Define an IBM MQ message, writing some text in UTF format
    MQMessage hello_world = new MQMessage();
    hello_world.WriteUTF("Hello World!");

    // Specify the message options
    MQPutMessageOptions pmo = new MQPutMessageOptions(); // accept the defaults,
                                                         // same as MQPMO_DEFAULT

    // Put the message on the queue
    system_default_local_queue.Put(hello_world, pmo);



    // Get the message back again

    // First define an IBM MQ message buffer to receive the message
    MQMessage retrievedMessage =new MQMessage();
    retrievedMessage.MessageId =hello_world.MessageId;

    // Set the get message options
    MQGetMessageOptions gmo =new MQGetMessageOptions(); //accept the defaults
                                                        //same as MQGMO_DEFAULT

    // Get the message off the queue
    system_default_local_queue.Get(retrievedMessage,gmo);

    // Prove we have the message by displaying the UTF message text
    String msgText = retrievedMessage.ReadUTF();
    Console.WriteLine("The message is: {0}", msgText);

    // Close the queue
    system_default_local_queue.Close();

    // Disconnect from the queue manager
    qMgr.Disconnect();
  }
```

```
    //If an error has occurred,try to identify what went wrong.

    //Was it an IBM MQ error?
    catch (MQException ex)
    {
      Console.WriteLine("An IBM MQ error occurred: {0}", ex.ToString());
    }

    catch (System.Exception ex)
    {
      Console.WriteLine("A System error occurred: {0}", ex.ToString());
    }

    return 0;
  }//end of start
}//end of sample
```

IBM MQ ortamının ayarlanması

Bir kuyruk yöneticisine bağlanmak için istemci bağlantısını kullanmadan önce IBM MQ ortamını ayarlamanız gerekir.

Not: Sunucu bağ tanımları kipinde IBM MQ classes for .NET kullanılırken bu adım gerekli değildir.

.NET programlama arabirimi, NMQ_MQ_LIB uyarlama değerini kullanmanıza olanak sağlar, ancak bir sınıf MQEnvironment içerir. Bu sınıf, bağlantı girişimi sırasında kullanılacak ayrıntıları belirtmenizi sağlar (örneğin, aşağıdaki listede yer alan ayrıntılar gibi):

- Kanal adı
- Anasistem adı
- Kapı numarası
- Kanal çıkışları
- SSL parametreleri
- Kullanıcı kimliği ve parola

MQEnvironment sınıfıyla ilgili tam bilgi için MQEnvironment.NET class başlıklı konuya bakın.

Kanal adını ve anasistem adını belirtmek için aşağıdaki kodu kullanın:

```
MQEnvironment.Hostname = "host.domain.com";
MQEnvironment.Channel  = "client.channel";
```

Varsayılan olarak, istemciler 1414 kapısındaki bir IBM MQ dinleyicisine bağlanmayı dener. Farklı bir kapı belirtmek için şu kodu kullanın:

```
MQEnvironment.Port = nnnn;
```

Kuyruk yöneticisine bağlanma ve bağlantı kesme

IBM MQ ortamını yapılandırdığınızda, bir kuyruk yöneticisine bağlanmaya hazırsınız.

Bir kuyruk yöneticisine bağlanmak için MQQueueManager sınıfının yeni bir eşgörünümünü yaratın:

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Bir kuyruk yöneticisiyle bağlantıyı kesmek için, kuyruk yöneticisinde `Disconnect` yöntemini çağırın:

```
queueManager.Disconnect();
```

Kuyruk yöneticisine bağlanmayı denerken, kuyruk yöneticisi üzerinde sorma (`inq`) yetkiniz olmalıdır. Sorma yetkisi olmadan, bağlantı girişimi başarısız olur.

`Disconnect` yöntemini çağırırsanız, o kuyruk yöneticisinden eriştiğiniz tüm açık kuyruklar ve işlemler kapatılır. Ancak, bu kaynakları kullanmayı bitirdiğinizde açık bir şekilde kapatmak iyi bir programlama uygulamasıdır. Kaynakları kapatmak için, her kaynakla ilişkili nesnede `Close` yöntemini kullanın.

Bir kuyruk yöneticisindeki `Commit` ve `Backout` yöntemleri, yordamsal arabirimle kullanılan MQCMIT ve MQBACK çağrılarının yerine geçer.

Kuyruklara ve konulara erişilmesi

Kuyruklara ve konulara MQQueueManager yöntemlerini ya da uygun oluşturucuları kullanarak erişebilirsiniz.

Kuyruklara erişmek için MQQueueManager sınıfının yöntemlerini kullanın. MQOD (nesne tanımlayıcı yapısı) bu yöntemlerin değiştirgelerine daraltıldı. Örneğin, queueManageradlı bir MQQueueManager nesnesiyle gösterilen bir kuyruk yöneticisinde kuyruk açmak için aşağıdaki kodu kullanın:

```
MQQueue queue = queueManager.AccessQueue("qName",
                                         MQC.MQOO_OUTPUT,
                                         "qMgrName",
                                         "dynamicQName",
                                         "altUserId");
```

*option*s değiştirgesi, MQOPEN çağrısındaki Options değiştirgesiyle aynıdır.

AccessQueue yöntemi, MQQueue sınıfının yeni bir nesnesini döndürür.

Kuyruğu kullanmayı tamamladığınızda, aşağıdaki örnekte olduğu gibi, bu kuyruğu kapatmak için Close () yöntemini kullanın:

```
queue.Close();
```

IBM MQ .NETile MQQueue oluşturucusunu kullanarak da kuyruk yaratabilirsiniz. Bu değiştirgeler, kullanılacak somutlaştırılmış MQQueueManager nesnesini belirten bir kuyruk yöneticisi değiştirgesinin eklenmesiyle accessQueue yöntemiyle tamamen aynıdır. Örneğin:

```
MQQueue queue = new MQQueue(queueManager,
                            "qName",
                            MQC.MQOO_OUTPUT,
                            "qMgrName",
                            "dynamicQName",
                            "altUserId");
```

Bu şekilde bir kuyruk nesnesi oluşturulması, kendi MQQueue alt sınıflarınızı yazmanızı sağlar.

Benzer şekilde, konulara MQQueueManager sınıfının yöntemlerini kullanarak da erişebilirsiniz. Bir konuyu açmak için AccessTopic() yöntemini kullanın. Bu, MQTopic sınıfının yeni bir nesnesini döndürür. Konuyu kullanmayı tamamladığınızda, konuyu kapatmak için MQTopic 'in Close () yöntemini kullanın.

MQTopic oluşturucusunu kullanarak da konu yaratabilirsiniz. Konular için bir dizi oluşturucu vardır; daha fazla bilgi için bkz. MQTopic.NET sınıf.

İletilerin işlenmesi

İletiler, kuyruğun ya da konu sınıflarının yöntemleri kullanılarak işlenir. Yeni bir ileti oluşturmak için yeni bir MQMessageobject yaratın.

MQQueue ya da MQTopic sınıfının Put () yöntemini kullanarak iletileri kuyruklara ya da konulara koyun. MQQueue ya da MQTopic sınıfının Get () yöntemini kullanarak kuyruklardan ya da konulardan ileti alın. MQPUT ve MQGET ' in byte dizilerini koyduğu ve aldığı yordamsal arabirimden farklı olarak, IBM MQ classes for .NET MQMessage sınıfının eşgörünümlerini koyar ve alır. MQMessage sınıfı, gerçek ileti verilerini içeren veri arabelleğini, o iletiyi tanımlayan tüm MQMD (ileti tanımlayıcı) değiştirgeleriyle birlikte içerir.

Yeni bir ileti oluşturmak için MQMessage sınıfının yeni bir eşgörünümünü oluşturun ve verileri ileti arabelleğine koymak için WriteXXX yöntemlerini kullanın.

Yeni ileti eşgörünümü yaratıldığında, MQMD için başlangıç değerleri ve dil bildirimleriiçinde tanımlandığı şekilde, tüm MQMD parametreleri otomatik olarak varsayılan değerlerine ayarlanır. MQQueue nesnesinin Put () yöntemi, parametre olarak MQPutMessageOptions sınıfının bir eşgörünümünü de alır. Bu sınıf MQPMO yapısını temsil eder. Aşağıdaki örnek bir ileti yaratır ve bir kuyruğa koyar:

```
// 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);
```

MQQueue nesnesinin Get () yöntemi, kuyruktan alınan iletiyi gösteren yeni bir MQMessage eşgörünümü döndürür. Değiştirge olarak MQGetMessageSeçenekleri sınıfının bir eşgörünümünü de alır. Bu sınıf MQGMO yapısını temsil eder.

İleti büyüklüğü üst sınırını belirtmenize gerek yoktur; Get () yöntemi, iç arabelleğinin büyüklüğünü gelen iletiye sığacak şekilde otomatik olarak ayarlar. Döndürülen iletideki verilere erişmek için MQMessage sınıfının ReadXXX yöntemlerini kullanın.

Aşağıdaki örnek, bir kuyruktan ileti nasıl alacağınızı göstermektedir:

```
// 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();
```

Okuma ve yazma yöntemlerinin kullandığı sayı biçimini *kodlama* üye değişkenini ayarlayarak değiştirebilirsiniz.

characterSet üye değişkenini ayarlayarak karakter kümesini, dizgileri okumak ve yazmak için kullanılacak şekilde değiştirebilirsiniz.

Daha fazla ayrıntı için bkz. MQMessage.NET sınıf .

Not: MQMessage WriteUTF() yöntemi, dizginin uzunluğunu ve içerdiği Unicode byte 'ları otomatik olarak kodlar. İletiniz başka bir .NET programı tarafından okunduğunda (ReadUTF() kullanılarak), dizgi bilgisi göndermenin en basit yolu budur.

İleti özelliklerinin işlenmesi

İleti özellikleri, iletileri seçmenizi ya da üstbilgilerine erişmeden bir iletiye ilişkin bilgileri almanızı sağlar. MQMessage sınıfı, özellikleri almak ve ayarlamak için yöntemler içerir.

Bir uygulamanın işlenecek iletileri seçmesine ya da MQMD ya da MQRFH2 üstbilgilerine erişmeden bir iletiyle ilgili bilgileri almasına izin vermek için ileti özelliklerini kullanabilirsiniz. IBM MQ ve JMS uygulamaları arasındaki iletişimi de kolaylaştırır. IBM MQiçindeki ileti özellikleri hakkında daha fazla bilgi için bkz. İleti özellikleri.

MQMessage sınıfı, özelliğin veri tipine göre özellikleri almak ve ayarlamak için bir dizi yöntem sağlar. Alma (get) yöntemlerinin adları Get * Property biçimindedir ve küme yöntemlerinin adları Set * Property biçimindedir; burada yıldız işareti (*) aşağıdaki dizgilerden birini gösterir:

- Boole

- Byte
- Bayt
- Çift
- Kayar Noktalı Sayı
- Tamsayı
- Int2
- Int4
- Int8
- Uzun
- Nesne
- Kısa
- Dizgi

Örneğin, IBM MQ özelliğini (bir karakter dizgisi) almak için `message.GetStringProperty('myproperty')` çağrıyı kullanın. İsteğe bağlı olarak, IBM MQ ' un tamamlayacağı bir özellik tanımlayıcısı iletebilirsiniz.

Hataların işlenmesi

`try` ve `catch` bloklarını kullanarak IBM MQ classes for .NET ' den kaynaklanan hataları işleyin.

.NET arabirimindeki yöntemler bir tamamlanma kodu ve neden kodu döndürmez. Bunun yerine, bir IBM MQ çağrısından kaynaklanan tamamlanma kodu ve neden kodu sıfır değilse bir kural dışı durum oluşur. Bu, IBM MQ' e her çağrıdan sonra dönüş kodlarını denetlemeniz gerekmemesi için program mantığını basitleştirir. Programınızda hangi noktalarda başarısızlık olasılığıyla başa çıkmak istediğinize karar verebilirsiniz. Bu noktalarda, kodunuzu aşağıdaki örnekte olduğu gibi `try` ve `catch` bloklarıyla çevreleyebilirsiniz:

```
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 );
}
```

Öznitelik değerlerini alma ve ayarlama

MQManagedObject, MQQueue ve MQQueueManager sınıfları, öznitelik değerlerini almanızı ve ayarlamanızı sağlayan yöntemler içerir. MQQueue için, yöntemlerin yalnızca, kuyruğu açtığınızda uygun sorgunun ve işaretlerin belirtilmesi durumunda işe yarayacağı unutulmamalıdır.

Sık kullanılan öznitelikler için, MQQueueManager ve MQQueue sınıfları MQManagedObjectadlı bir sınıftan edinilir. Bu sınıf, Inquire () ve Set () arabirimlerini tanımlar.

Yeni işlecini kullanarak yeni bir kuyruk yöneticisi nesnesi yarattığınızda, nesne sorgu için otomatik olarak açılır. Bir kuyruk nesnesine erişmek için AccessQueue() yöntemini kullandığınızda, bu nesne sorgu ya da ayarlama işlemleri için otomatik olarak açılmaz; bu, bazı uzak kuyruk tiplerinde sorunlara neden olabilir. Sorma ve Ayarlama yöntemlerini kullanmak ve bir kuyruktaki özellikleri ayarlamak için, AccessQueue() yönteminin openOptions parametresinde uygun sorgulamayı belirtmeniz ve işaretleri ayarlamanız gerekir.

Sorma ve ayarlama yöntemleri üç parametre alır:

- seçiciler dizisi
- intAttrs dizisi
- charAttrs dizisi

Bir dizinin uzunluğu her zaman bilindiğinden, MQINQ ' da bulunan SelectorCount, IntAttrCount ve CharAttrLength değiştirgelerine gerek yoktur. Aşağıdaki örnek, bir kuyrukta nasıl sorgu yapılacağını göstermektedir:

```
//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);
```

Bu nesnelerin tüm öznitelikleri sorulabiliyor. Öznitelikler bir alt kümesi, bir nesnenin özellikleri olarak gösterilir. Nesne özniteliklerinin bir listesi için Nesnelerin öznitelikleribaşlıklı konuya bakın. Nesne özellikleri için uygun sınıf tanımına bakın.

Çok parçacıklı programlar

.NET yürütme ortamı, doğal olarak çoklu iş parçacıklıdır. IBM MQ classes for .NET , bir kuyruk yöneticisi nesnesinin birden çok iş parçacığı arasında paylaşılmasını sağlar, ancak hedef kuyruk yöneticisine tüm erişimin eşitlenmesini sağlar.

Bir kuyruk yöneticisine bağlanan ve başlatma sırasında bir kuyruk açan basit bir program düşünün. Program ekranda tek bir düğme görüntüler. Bir kullanıcı bu düğmeyi tıklattığında, program kuyruktan bir ileti getirir. Bu durumda, uygulama kullanıma hazırlama tek bir iş parçacığında gerçekleşir ve düğmeye basıldığında yürütülen kod ayrı bir iş parçacığında (kullanıcı arabirimi iş parçacığı) yürütülür.

IBM MQ .NET uygulaması, belirli bir bağlantı (MQQueueManager nesne eşgörünümü) için, hedef IBM MQ kuyruk yöneticisine tüm erişimin uyumlulaştırılmasını sağlar. Varsayılan davranış, bir kuyruk yöneticisine çağrı yapmak isteyen bir iş parçacığının, o bağlantı için devam eden diğer tüm çağrılar tamamlanıncaya kadar engellenmesidir. Programınızdaki birden çok iş parçacığından aynı kuyruk yöneticisine eşzamanlı erişim gerekiyorsa, eşzamanlı erişim gerektiren her iş parçacığı için yeni bir MQQueueManager nesnesi yaratın. (Bu, her iş parçacığı için ayrı bir MQCONN çağrısı yayınlanmasına eşdeğerdir.)

Varsayılan bağlantı seçenekleri MQC.MQCNO_HANDLE_SHARE_NONE ya da MQC.MQCNO_SHARE_NO_BLOCK , kuyruk yöneticisi artık eşitlenmiyor.

.NET ile istemci kanal tanımlama çizelgesinin kullanılması

IBM MQ classes for .NETile bir istemci kanal tanımlama çizelgesi (CCDT) kullanabilirsiniz. Yönetilen ya da yönetilmeyen bir bağlantı kullanıp kullanmadığınıza bağlı olarak, CCDT ' nin konumunu farklı şekillerde belirtirsiniz.

XA ya da XA yönetilmeyen istemci bağlantısı tipi

Yönetilmeyen bir bağlantı tipiyle, CCDT ' nin konumunu iki şekilde belirtebilirsiniz:

- Çizelgenin bulunduğu dizini belirtmek için MQCHLLIB ortam değişkenlerini ve çizelgenin dosya adını belirtmek için MQCHLTAB ' yi kullanın.
- İstemci yapılanış kütüğü kullanılıyor. KANAL kısmında, çizelgenin bulunduğu dizini belirtmek için **ChannelDefinitionDirectory** özniteliklerini ve dosya adını belirtmek için **ChannelDefinitionFile** özniteliklerini kullanın.

Konum hem istemci yapılandırma dosyasında hem de ortam değişkenleri kullanılarak belirtilirse, ortam değişkenleri önceliklidir. İstemci yapılandırma dosyasında standart bir konum belirtmek ve gerektiğinde ortam değişkenlerini kullanarak bu özelliği geçersiz kılmak için bu özelliği kullanabilirsiniz.

Yönetilen istemci bağlantısı tipi

Yönetilen bağlantı tipiyle, CCDT ' nin yerini üç şekilde belirtebilirsiniz:

- .NET uygulama yapılandırma dosyası kullanılıyor. KANAL bölümünde, çizelgenin bulunduğu dizini belirtmek için **ChannelDefinitionDirectory** ve kütük adını belirtmek için **ChannelDefinitionFile** tuşlarını kullanın.
- Çizelgenin bulunduğu dizini belirtmek için MQCHLLIB ortam değişkenlerini ve çizelgenin dosya adını belirtmek için MQCHLTAB ' yi kullanın.
- İstemci yapılanış kütüğü kullanılıyor. KANAL kısmında, çizelgenin bulunduğu dizini belirtmek için **ChannelDefinitionDirectory** özniteliklerini ve dosya adını belirtmek için **ChannelDefinitionFile** özniteliklerini kullanın.

Konum birden çok şekilde belirtilirse, ortam değişkenleri istemci yapılanış kütüğüne göre önceliklidir ve .NET Application Configuration File (Uygulama Yapılandırma Dosyası) diğer her iki yönteme göre önceliklidir. Bu özelliği, istemci yapılanış dosyasında standart bir yer belirtmek ve gerektiğinde ortam değişkenlerini ya da uygulama yapılanış kütüğünü kullanarak geçersiz kılmak için kullanabilirsiniz.

IBM MQ 9.3.0' den önce, kuyruk yöneticisi gruplamasıyla CCDT kullanırken, yönetilen .NET istemcisi ile IBM MQ Java ve C istemcileri arasında davranış farkı vardır. Bir CCDT dosyası üç kuyruk yöneticisinden oluşan bir kuyruk yöneticisi grubu ve aynı üç kuyruk yöneticisine üç belirtik CLNTCONN içerdiğinde ve uygulama, kuyruk yöneticisi olarak "*" , C ve Java istemcileri MQRC_Q_MGR_NAME_ERRORdeğerini döndürdüğünde. Ancak, yönetilen .NET istemcisi kullanılabilir ilk CLNTCONN ' u kullanır ve kullanılamıyorsa, CLNTCONN gruplamış kuyruk yöneticisini kullanır.

V 9.3.0 V 9.3.0 IBM MQ 9.3.0' den .NET istemcisi, C ve Java istemcileriyle aynı şekilde davranır ve kuyruk yöneticisi gruplamasıyla CCDT kullanırken MQRC_Q_MGR_NAME_ERROR değerini döndürür.

.NET uygulamasının hangi kanal tanımının kullanılacağını belirleme şekli

IBM MQ .NET istemcisi ortamında, kullanılacak kanal tanımı çeşitli şekillerde belirtilebilir. Kanal tanımının birden çok belirtimi olabilir. Bir uygulama kanal tanımını bir ya da daha çok kaynaktan türetir.

Birden çok kanal tanımı varsa, kullanılan kanal aşağıdaki öncelik sırasıyla seçilir:

1. MQQueueManager oluşturucusunda belirtik olarak ya da *MQC.CHANNEL_PROPERTY*
2. MQEnvironment.properties hashtable içinde bir özellik *MQC.CHANNEL_PROPERTY*
3. MQEnvironment içindeki *Kanal* özelliği
4. .NET uygulama yapılandırma dosyası, bölüm adı KANAL, anahtar ServerConnectionParms (yalnızca yönetilen bağlantılar için geçerlidir)
5. *MQSERVER* ortam değişkeni
6. İstemci yapılanış kütüğü, stanza KANAL, Attribute ServerConnectionParms
7. İstemci kanal tanımlama çizelgesi (CCDT). CCDT ' nin konumu .NET uygulama yapılandırma dosyasında belirtilir (yalnızca yönetilen bağlantılar için geçerlidir)
8. İstemci kanal tanımlama çizelgesi (CCDT). CCDT ' nin yeri *MQCHLIB* ve *MQCHLTAB* ortam değişkenleri kullanılarak belirtilir.
9. İstemci kanal tanımlama çizelgesi (CCDT). CCDT ' nin yeri istemci yapılanış kütüğü kullanılarak belirtilir.

1-3 numaralı öğeler için kanal tanımı, uygulama tarafından sağlanan değerlerden alan temelinde oluşturulur. Bu değerler farklı arabirimler kullanılarak sağlanabilir ve her biri için birden çok değer bulunabilir. Alan değerleri, belirtilen öncelik sırasına göre kanal tanımına eklenir:

1. MQQueueManager oluşturucusundaki *connName* değeri

2. MQQueueManager.properties hashtable içindeki özelliklerin değerleri
3. MQEnvironment.properties hashtable içindeki özelliklerin değerleri
4. MQEnvironment alanları olarak ayarlanan değerler (örneğin, MQEnvironment.Hostname, MQEnvironment.Port)

4-6 numaralı öğeler için, tüm kanal tanımı değer olarak sağlanır. Kanal tanımında belirlenmemiş alanlar sistem varsayılanlarını alır. Diğer kanal tanımlama yöntemlerinden ve bunların alanlarından gelen değerler bu belirtimlerle birleştirilmez.

7-9 numaralı öğeler için, tüm kanal tanımı CCDT ' den alınır. Kanal tanımlandığında açık olarak belirtilmeyen alanlar sistem varsayılanlarını alır. Diğer kanal tanımlama yöntemlerinden ve bunların alanlarından gelen değerler bu belirtimlerle birleştirilmez.

IBM MQ .NET içinde kanal çıkışlarının kullanılması

İstemci bağ tanımlarını kullanıyorsanız, kanal çıkışlarını başka bir istemci bağlantısında olduğu gibi kullanabilirsiniz. Yönetilen bağ tanımlarını kullanıyorsanız, uygun bir arabirimi uygulayan bir çıkış programı yazmanız gerekir.

İstemci bağ tanımları

İstemci bağ tanımlarını kullanıyorsanız, kanal çıkışlarını Kanal çıkışlarıkonusunda açıklandığı gibi kullanabilirsiniz. Yönetilen bağ tanımları için yazılan kanal çıkışlarını kullanamazsınız.

Yönetilen bağ tanımları

Yönetilen bir bağlantı kullanıyorsanız, bir çıkış uygulamak için, uygun arabirimi uygulayan yeni bir .NET sınıfı tanımlarsınız. IBM MQ paketinde üç çıkış arabirimi tanımlanır:

- MQSendExit
- MQReceiveExit
- MQSecurityExit

Not: Bu arabirimler kullanılarak yazılan kullanıcı çıkışları, yönetilmeyen ortamda kanal çıkışları olarak desteklenmez.

Aşağıdaki örnek, üçünü de uygulayan bir sınıfı tanımlar:

```
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
  }

}
```

Her çıkışa bir MQChannelExit ve bir MQChannelDefinition nesnesi eşgörünümü geçirilir. Bu nesneler, yordamsal arabirimde tanımlı MQCXP ve MQCD yapılarını gösterir.

Bir gönderme çıkışı tarafından gönderilecek veriler ve bir güvenlik ya da alma çıkışında alınan veriler, çıkışın parametreleri kullanılarak belirlenir.

Girdide, bayt dizisindeki *dataOffset* uzunluğu *dataLength* ile göreli konumundaki veriler *dataBuffer* , bir gönderme çıkışı tarafından gönderilmek üzere olan verilerdir ve bir güvenlik ya da alma çıkışında alınan verilerdir. *dataMaxLength* parametresi, uzunluk üst sınırını verir (*dataOffset* içinden) *dataBuffer*içindeki çıkışta kullanılabilir. Not: Bir güvenlik çıkışı için, dataBuffer boş değerli olabilir; bu, çıkışın ilk çağrılışıysa ya da iş ortağı sona veri göndermeden seçildiyse.

Dönüşte, *dataOffset* ve *dataLength* değeri, .NET sınıflarının kullanması gereken, döndürülen bayt dizisi içindeki göreli konumu ve uzunluğu gösterecek şekilde ayarlanmalıdır. Gönderme çıkışında, bu, göndermesi gereken verileri ve bir güvenlik ya da alma çıkışı için yorumlanması gereken verileri gösterir. Çıkışın normalde bir bayt dizisi döndürmesi gerekir; kural dışı durumlar, veri göndermemeyi seçebilen bir güvenlik çıkışıdır ve INIT ya da TERM nedenleriyle çağrılan herhangi bir çıkış çağrılır. Bu nedenle yazılabilir en basit çıkış biçimi dataBufferdöndürmekten başka bir şey yapmayan bir çıkış biçimidir:

Olası en basit çıkış gövdesi:

```
{
  return dataBuffer;
}
```

MQChannelDefinition sınıfı

Yönetilen .NET istemci uygulamasıyla belirtilen kullanıcı kimliği ve parola, istemci güvenlik çıkışına geçirilen IBM MQ .NET MQChannelDefinition sınıfında ayarlanır. Güvenlik çıkışı, kullanıcı kimliğini ve parolayı MQCD.RemoteUserIdentifier ve MQCD.RemotePassword alanları (bkz. “Güvenlik çıkışı yazılması” sayfa 933).

Kanal çıkışlarının belirtilmesi (yönetilen istemci)

MQQueueManager nesnenizi yaratırken (MQEnvironment ya da MQQueueManager oluşturucusunda) bir kanal adı ve bağlantı adı belirtirseniz, kanal çıkışlarını iki şekilde belirtebilirsiniz.

Öncelik sırasına göre şunlar:

1. MQQueueManager oluşturucusuna MQC.SECURITY_EXIT_PROPERTY, MQC.SEND_EXIT_PROPERTY ya da MQC.RECEIVE_EXIT_PROPERTY hashtable özelliklerinin geçirilmesi.
2. MQEnvironment SecurityExit, SendExit ya da ReceiveExit özelliklerinin ayarlanması

Bir kanal adı ve bağlantı adı belirlemezseniz, kullanılacak kanal çıkışları, bir istemci kanal tanımlama çizelgesinden (CCDT) alınan kanal tanımından gelir. Kanal tanımında saklanan değerlerin geçersiz kılınması mümkün değildir. Kanal tanımlama çizelgelerine ilişkin ek bilgi için İstemci kanal tanımlama çizelgesi ve “.NET ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 567 başlıklı konuya bakın.

Her bir durumda, belirtim aşağıdaki biçime sahip bir dizgi biçimini alır:

```
Assembly_name(Class_name)
```

Sınıf_adı , IBM.WMQ.MQSecurityExit, IBM.WMQ.MQSendExit ya da IBM.WMQ.MQReceiveExit arabirimini (uygun olduğu şekilde) uygulayan bir .NET sınıfının ad alanı belirtimi de içinde olmak üzere tam olarak nitelenmiş adıdır. *Assembly_name* , sınıfı barındıran yapıbiriminin dosya uzantısı da içinde olmak üzere

tam olarak nitelenmiş konumdur. MQEnvironment ya da MQQueueManagerözelliklerini kullanıyorsanız, dizginin uzunluğu 999 karakterle sınırlıdır. Ancak, CCDT ' de kanal çıkış adı belirtilirse, ad 128 karakterle sınırlıdır. Gerektiğinde, .NET istemci kodu dizgi belirtimini ayrıştırarak belirtilen sınıfın bir eşgörünümünü yükler ve yaratır.

Kanal çıkışı kullanıcı verilerinin belirtilmesi (yönetilen istemci)

Kanal çıkışlarıyla ilişkilendirilmiş kullanıcı verileri olabilir. MQQueueManager nesnesinizi yaratırken (MQEnvironment ya da MQQueueManager oluşturucusunda) bir kanal adı ve bağlantı adı belirtirseniz, kullanıcı verilerini iki şekilde belirtebilirsiniz.

Öncelik sırasına göre şunlar:

1. MQQueueManager oluşturucusuna MQC.SECURITY_USERDATA_PROPERTY, MQC.SEND_USERDATA_PROPERTY ya da MQC.RECEIVE_USERDATA_PROPERTY hashtable özelliklerinin geçirilmesi.
2. MQEnvironment SecurityUserVerilerinin, SendUserVerilerinin ya da ReceiveUserVeri özelliklerinin ayarlanması.

Bir kanal adı ve bağlantı adı belirlemezseniz, kullanılacak çıkış kullanıcısı veri değerleri, istemci kanal tanımlama çizelgesinden (CCDT) alınan kanal tanımından gelir. Kanal tanımında saklanan değerlerin geçersiz kılınması mümkün değildir. Kanal tanımlama çizelgelerine ilişkin ek bilgi için İstemci kanal tanımlama çizelgesi ve “.NET ile istemci kanal tanımlama çizelgesinin kullanılması” sayfa 567 başlıklı konuya bakın.

Her durumda, belirtim 32 karakterle sınırlı bir dizedir.

.NET içinde otomatik istemci yeniden bağlantısı

İstemcinizin beklenmeyen bir bağlantı kesmesi sırasında kuyruk yöneticisine otomatik olarak yeniden bağlanmasını sağlayabilirsiniz.

Örneğin, kuyruk yöneticisi durursa ya da ağ ya da sunucu başarısız olursa, istemcinin kuyruk yöneticisiyle bağlantısı beklenmedik bir şekilde kesilir.

Otomatik istemci yeniden bağlantısı olmadan, bağlantı başarısız olduğunda bir hata üretilir. Bağlantıyı yeniden kurmanıza yardımcı olması için hata kodunu kullanabilirsiniz.

Otomatik istemci yeniden bağlanma olanağını kullanan bir istemciye yeniden bağlanabilir istemci adı verilir. Yeniden bağlanabilir bir istemci yaratmak için, kuyruk yöneticisine bağlanırken yeniden bağlanma seçenekleri adı verilen bazı seçenekleri belirleyin.

İstemci uygulaması bir IBM MQ .NET istemcisiyse, kuyruk yöneticisi yaratmak için `MQQueueManager` sınıfını kullandığınızda, CONNECT_OPTIONS_PROPERTY için uygun bir değer belirterek otomatik istemci yeniden bağlantısı kurmayı tercih edebilir. CONNECT_OPTIONS_PROPERTY değerlerinin ayrıntıları için bkz. Reconnection options .

İstemci uygulamasının her zaman aynı adı taşıyan bir kuyruk yöneticisine mi, aynı kuyruk yöneticisine mi, yoksa istemci bağlantı çizelgesinde aynı `QMNAME` ile tanımlanan herhangi bir kuyruk yöneticisine mi bağlanacağını seçebilirsiniz (Ayrıntılar için CCDT ' deki Kuyruk Yöneticisi Grupları konusuna bakın).

.NET için Transport Layer Security (TLS) desteği

IBM MQ classes for .NET istemci uygulamaları TLS (Transport Layer Security; İletim Katmanı Güvenliği) şifrelemesini destekler. TLS iletişim kuralı, internet üzerinden iletişim güvenliği sağlar ve istemci/sunucu uygulamalarının gizli ve güvenilir bir şekilde iletişim kurmasına izin verir.

İlgili kavramlar

IBM MQ.NET yönetilen istemci TLS desteği

Şifreleme güvenliği iletişim kuralları: TLS

Yönetilmeyen .NET istemcisi için TLS desteği

Yönetilmeyen .NET istemcisi için TLS desteği C MQI ve IBM Global Security Kit (GSKit)' ye dayalıdır. C MQI, TLS işlemlerini işler ve GSKit , TLS güvenli yuva iletişim kurallarını uygular.

Yönetilmeyen .NET istemcisi için TLS ' yi etkinleştirme
TLS yalnızca istemci bağlantıları için desteklenir. TLS 'yi etkinleştirmek için, kuyruk yöneticisiyle iletişim kurarken kullanılacak CipherSpec ' i belirtmeniz gerekir ve bu, hedef kanalda ayarlanan CipherSpec ile eşleşmelidir.

TLS ' yi etkinleştirmek için, MQEnvironment 'ın SSLCipherSpec statik üye değişkenini kullanarak CipherSpec belirtin. Aşağıdaki örnek, TLS_RSA_WITH_AES_128_CBC_SHA: CipherSpec ile TLS gerektirecek şekilde ayarlanan SECURE.SVRCONN.CHANNELadlı bir SVRCONN kanalına bağlanır.

V 9.3.0 V 9.3.0

```
MQEnvironment.Hostname         = "your_hostname";
MQEnvironment.Channel          = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.SSLCipherSpec    = "TLS_RSA_WITH_AES_128_CBC_SHA256";
MQEnvironment.SSLKeyRepository = "C:\mqm\key.kdb";
MQQueueManager qmgr = new MQQueueManager("your_Q_manager");
```

CipherSpecs'in bir listesi için CipherSpecs' In Belirtilmesi başlıklı konuya bakın.

SSLCipherSpec özelliği, bağlantı özelliklerinin HASH çizelgesinde MQC.SSL_CIPHER_SPEC_PROPERTY kullanılarak da ayarlanabilir.

TLS kullanarak başarıyla bağlanmak için istemci anahtar deposu, kuyruk yöneticisi tarafından sunulan sertifikanın doğrulanabileceği Sertifika Yetkilisi kök sertifikaları zinciriyle ayarlanmalıdır. Benzer şekilde, SVRCONN kanalındaki SSLClientAuth MQSSL_CLIENT_AUTH_REQUIRED olarak ayarlandıysa, istemci anahtar deposu kuyruk yöneticisi tarafından güvenilen tanımlayıcı bir kişisel sertifika içermelidir.

Kuyruk Yöneticisinin Ayırt Edici Adının Kullanılması
Kuyruk yöneticisi kendisini, *Ayırt Edici Ad* (DN) içeren bir TLS sertifikasını kullanarak tanımlar.

Bir IBM MQ .NET istemci uygulaması, doğru kuyruk yöneticisiyle iletişim kurduğundan emin olmak için bu DN ' yi kullanabilir. MQEnvironment 'ın sslPeerAd değişkeni kullanılarak bir DN örüntüsü belirtildi. Örneğin, ayar:

```
MQEnvironment.SSLPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

Yalnızca kuyruk yöneticisi, QMGR. ile başlayan bir Ortak Adı olan bir sertifika sunarsa bağlantının başarılı olmasını sağlar. ve en az iki Kuruluş Birimi adı; bunlardan ilki IBM ve ikincisi WEBSPHERE olmalıdır.

SSLPeerName özelliği, bağlantı özelliklerinin HASH çizelgesinde MQC.SSL_PEER_NAME_PROPERTY kullanılarak da ayarlanabilir. Ayırt Edici Adlar ve eşdüzey adları ayarlamaya ilişkin kurallar hakkında daha fazla bilgi için bkz. IBM MQ güvenliğinin sağlanması.

SSLPeerName ayarlanırsa, bağlantılar yalnızca geçerli bir kalıba ayarlandıysa ve kuyruk yöneticisi eşleşen bir sertifika sunduysa başarılı olur.

TLS kullanılırken hata işleme

TLS kullanılarak bir kuyruk yöneticisine bağlanırken IBM MQ classes for .NET tarafından aşağıdaki neden kodları yayınlanabilir:

MQRC_SSL_NOT_ALLOWED
SSLCipherSpec özelliği ayarlandı, ancak bağ tanımları bağlantısı kullanıldı. Yalnızca istemci bağlantısı TLS ' yi destekler.

MQRC_SSL_PEER_NAME_MISMATCH
SSLPeerName özelliğinde belirtilen DN kalıbı, kuyruk yöneticisi tarafından sunulan DN ile eşleşmedi.

MQRC_SSL_PEER_NAME_ERROR (QRC_SSL_PEER_NAME_ERROR)
SSLPeerName özelliğinde belirtilen DN kalıbı geçerli değil.

V 9.3.0 V 9.3.0 **MQRC_KEY_REPOSITORY_HATA**
Anahtar havuzunun konumu belirtilmedi, geçerli değil ya da erişilemiyor.

Yönetilen .NET istemcisi için TLS desteği

Yönetilen .NET istemcisi, TLS güvenli yuva iletişim kurallarını uygulamak için Microsoft .NET Framework kitaplıklarını kullanır. Microsoft System.Net.SecuritySslStream sınıfı, bağlı TCP yuvaları üzerinden bir akış olarak çalışır ve bu yuva bağlantısı üzerinden veri gönderir ve alır.

Gerekli .NET Framework düzeyi alt sınırı: .NET Framework v3.5. Şifre Algoritması desteğinin düzeyi, uygulamanın kullandığı .NET Framework düzeyine dayalıdır:

- .NET Framework 3.5 ve 4.0düzeylerini temel alan uygulamalar için kullanılabilir güvenli yuva iletişim kuralları SSL 3.0 ve TSL 1.0' dır.
- .NET Framework düzey 4.5tabanlı uygulamalar için kullanılabilir güvenli yuva iletişim kuralları SSL 3.0, TLS 1.1 ve TLS 1.2' dir.

Daha yüksek TLS iletişim kuralı desteği beklenen uygulamaları, .NET Frameworkiçinde Microsoft Güvenlik desteği için tanımlandığı şekilde çerçevenin daha sonraki bir sürümüne taşımanız gerekebilir.

Yönetilen .NET istemcisi için TLS desteğinin ana özellikleri şunlardır:

TLS iletişim kuralı desteği
.NET yönetilen istemcisi için TLS desteği, .NET SSLStream sınıfı aracılığıyla tanımlanır ve uygulamanın kullandığı .NET Framework 'e bağlıdır. Daha fazla bilgi için bkz. “Yönetilen .NET istemcisi için TLS iletişim kuralı desteği” sayfa 574.

CipherSpec desteği
.NET yönetilen istemcisine ilişkin TLS ayarları, Microsoft.NET TLS kirişleri içindir. Daha fazla bilgi için bkz. “Yönetilen .NET istemcisi için CipherSpec desteği” sayfa 575 ve “Yönetilen .NET istemcisi için CipherSpec eşlemeleri” sayfa 576.

Anahtar havuzları
İstemci tarafındaki anahtar havuzu bir Windows anahtar deposunda bulunur. Sunucu tarafı havuzu, bir Şifreleme İletisi Sözdizimi (CMS) havuz tipidir. Daha fazla bilgi için bkz. “Yönetilen .NET istemcisi için temel havuzlar” sayfa 578.

Sertifikalar
Bir istemci ile kuyruk yöneticisi arasında karşılıklı kimlik doğrulaması gerçekleştirmek için kendinden imzalı TLS sertifikalarını kullanabilirsiniz. Daha fazla bilgi için bkz. “Yönetilen .NET istemcisi için sertifikaların kullanılması” sayfa 578.

SSLPEERNAME
.NETiçinde uygulamalar, bir Ayırt Edici Ad (DN) kalıbı belirtmek için isteğe bağlı SSLPEERNAME özniteliğini kullanabilir. Daha fazla bilgi için bkz. “SSLPEERNAME” sayfa 579.

FIPS uyumluluğu
FIPS ' nin programlı olarak etkinleştirilmesi, Microsoft.NET Security kitaplığı tarafından desteklenmez. FIPS etkinleştirmesi, Windows Grup İlkesi ayarı tarafından denetlenir.

NSA Suite B uyumluluğu
IBM MQ , RFC 6460 'ı uygular. NSA B grubu için Microsoft.NET uygulaması 5430 'dur. Bu, .NET Framework 3.5 ' ten itibaren desteklenir.

Gizli anahtar sıfırlama ya da yeniden anlaşma
SSLStream sınıfı, diğer IBM MQ istemcileriyle tutarlılık için gizli anahtar sıfırlamayı ya da yeniden anlaşma yapmayı desteklemese de .NET yönetilen istemcisi, uygulamaların `SSLKeyResetCount` ' yi ayarlamasına izin verir. Daha fazla bilgi için bkz. “Yönetilen .NET istemcisi için gizli anahtar sıfırlama ya da yeniden anlaşma” sayfa 580.

İptal denetimi
SSLStream sınıfı, sertifika zincirleme altyapısı tarafından otomatik olarak yapılan sertifika iptal denetimini destekler. Daha fazla bilgi için bkz. “İptal denetimi” sayfa 580.

IBM MQ güvenlik çıkışı desteği
`SSLStream` sınıfı, IBM MQ güvenlik çıkışları için sınırlı destek sağlar. `SSLPeerNamePtr`(Konu DN 'si) ve `SSLRemCertIssNamePtr` (Sertifika Veren DN 'si) almak için yerel ve uzak sertifikaların sorgulanması mümkündür; bu, Microsoft.NETiçinde desteklenir. Ancak, DNQ, UNSTRUCTUREDNAME

ve UNSTRUCTUREDADDRESS gibi öznitelikleri almak için destek yoktur, bu nedenle bu değerler çıkışlar kullanılarak alınamaz.

Şifreleme donanımı desteği
Yönetilen .NET istemcisi için şifreleme donanımı desteklenmez.

Yönetilen IBM MQ .NET ve XMS .NET istemcilerinde TLS1.3 desteği

V 9.3.2

İşletim sisteminin TLS1.3'ü desteklemesi koşuluyla, IBM MQ 9.3.2, IBM MQ .NET ve XMS .NET istemcileri TLS1.3 ' ü destekler.

Yönetilen .NET istemcisi, TLS güvenli yuva iletişim kurallarını uygulamak için Microsoft .NET Framework kitaplıklarını kullanır. Microsoft `System.Net.SecuritySslStream` sınıfı, bağlı TCP yuvaları üzerinden bir akış olarak çalışır ve bu yuva bağlantısı üzerinden veri gönderir ve alır.

Windowsişletim sistemlerinde .NET , SCHANNEL kullanır ve Linux .NET üzerinde SSL İletişimi için OpenSSL kullanır.

Windows Windowsüzerinde çalışan IBM MQ .NET istemci uygulamaları için

Microsoft , Windows 11 ve Windows Server 2022 ' in TSL1.3 şifrelerini varsayılan olarak desteklediğini duyurmuştu.

TLS_AES_128_GCM_SHA256 ve TLS_AES_256_GCM_SHA384 şifreleme takımları her iki Windowssürümünde de varsayılan olarak etkinleştirilir.

Uyarı:

- TLS_CHACHA20_POLY1305_SHA256 Cipher Suite varsayılan olarak etkinleştirilmez, ancak desteklenir.
- TLS1.3 etkinleştirilmiş bir IBM MQ .NET istemcisi için, bir kuyruk yöneticisine başarıyla bağlanmak için, IBM Global Security Kit (GSKit) sürüm 8.0.55.29 kuyruk yöneticisi tarafında gerekli olan en düşük sürümdür.

Linux Linuxüzerinde çalışan IBM MQ .NET istemci uygulamaları için

.NET , SSL Communication için Linux üzerinde OpenSSL kullandığından, TLS1.3'ü kullanmak için en düşük gereksinim OpenSSL v1.1.1 ' dir.

Ayrıca, .NET Linuxüzerinde OpenSSL kullandığından, OpenSSL tarafından desteklenen tüm şifrelemeler .NET için de çalışmalıdır.

OpenSSL , TLS1.3: için aşağıdakileri destekler: CipherSpecs

- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_GCM_SHA256
- TLS_AES_128_CCM_8_SHA256
- TLS_AES_128_CCM_SHA256

İlgili kavramlar
“Yönetilen .NET istemcisi için CipherSpec eşlemeleri” sayfa 576
IBM MQ.NET arabirimi, yönetilen istemcinin bir kuyruk yöneticisiyle güvenli bir bağlantı kurmak için kullanması gereken TLS iletişim kuralının sürümünü belirlemek için kullanılan bir IBM MQ - Microsoft.NET eşleme tablosu sağlar.

Yönetilen .NET istemcisi için TLS iletişim kuralı desteği
IBM MQ.NET TLS desteği, .NET SSLStream sınıfına dayalıdır.

Not: Yönetilen .NET istemcisi için TLS iletişim kuralı desteği, uygulamanın kullandığı .NET Framework düzeyine bağlıdır. Daha fazla bilgi için “Yönetilen .NET istemcisi için TLS desteği” sayfa 573 başlıklı konuya bakın.

Microsoft.NET SSLStream sınıfının TLS ' yi başlatması ve kuyruk yöneticisiyle el sıkışması gerçekleştirmesi için, ayarlamanız gereken parametrelerden biri **SSLProtocol**olup burada aşağıdaki değerlerden biri olması gereken TLS sürüm numarasını belirtmeniz gerekir:

- SSL3.0
- TLS1.0
- TLS1.2

Bu parametrenin değeri, tercih edilen CipherSpec öğesinin ait olduğu Protokol ailesiyle sıkıca bağlantılıdır. SSLStream, sunucuyla (kuyruk yöneticisi) TLS anlaşması başlattığında, anlaşma için kullanılacak CipherSpecs listesini tanımlamak için **SSLProtocol** içinde belirtilen TLS sürümünü kullanır.

IBM MQ.NET , uygulamaların bu değeri ayarlamak için kullanabileceği hiçbir özelliği kullanıma sunmaz. Bunun yerine IBM MQ , CipherSpec kümesini Protokol ailesiyle dahili olarak eşlemek için bir eşleme tablosu kullanır ve kullanılacak SSLProtocol sürümünü tanımlar. Bu çizelge, Microsoft.NET ile IBM MQarasında desteklenen CipherSpec ' in her birinin eşlenmesini ve bunların ait olduğu Protokol sürümünü gösterir. Daha fazla bilgi için “Yönetilen .NET istemcisi için CipherSpec eşlemeleri” sayfa 576 başlıklı konuya bakın.

Yönetilen .NET istemcisi için CipherSpec desteği
Bir uygulamaya ilişkin CipherSpec ayarları, sunucuyla yapılan anlaşma sırasında kullanılır.

IBM MQ istemcileri, kuyruk yöneticisiyle tokalaşma sırasında kullanılan bir CipherSpec değeri ayarlamanızı sağlar. IBM MQ istemcileri, Windows grup ilkesinde belirtilen CipherSpec ' i kurmak üzere güvenli bağlantı için geçerli bir CipherSpec ayarlamalıdır. Bu alanın boş bırakılması, yuvalar üzerinde güvenlik olmadan düz metin kanalını gösterir.

IBM MQ.NET yönetilen istemcisi için TLS ayarları, Microsoft.NET SSLStream sınıfı içindir. SSLStream için, CipherSpecya da CipherSpecstercih listesi, yalnızca bilgisayar çapında bir ayar olan Windows grup ilkesinde ayarlanabilir. Daha sonra SSLStream, sunucuyla tokalaşma sırasında belirtilen CipherSpec ya da tercih listesini kullanır. Diğer IBM MQ istemcileri durumunda, IBM MQ kanal tanımlamasındaki uygulamada CipherSpec özelliği ayarlanabilir ve TLS anlaşması için aynı ayar kullanılır. Bu sınırlamanın sonucu olarak TLS el sıkışması, IBM MQ kanal yapılandırmasında belirtilenlerden bağımsız olarak desteklenen herhangi bir CipherSpec ile ilişki kurabilir. Bu nedenle, bunun kuyruk yöneticisinde AMQ9631 hatasına neden olması olasıdır. Bu hatayı önlemek için, Windows grup ilkesinde TLS yapılandırması olarak uygulamada ayarladığınız CipherSpec değerini ayarlayın.

Yeni IBM MQ.NET TLS istemci kodu yalnızca doğru protokol sürümünün kararlaştırılıp kararlaştırılmadığını denetler. TLS iletişim kuralı sürümü, uygulamanın ayarladığı CipherSpec ' den türetilir ve sunucuyla TLS anlaşması için kullanılır (kuyruk yöneticisi). Bu nedenle, IBM MQ.NET yönetilen istemci uygulamasında CipherSpec ' i ayarlamak için tasarım gereği gereklidir. IBM MQ istemcisi tarafından ayarlanan CipherSpec SSL 3.0, TLS 1.0 ve TLS 1.2 iletişim kurallarından başka bir şeyse, IBM MQ yönetilen .NET istemcisi varsayılan olarak SSL 3.0 ya da TLS 1.0 iletişim kurallarından herhangi biriyle ilişki kurar ve bir hata bildirmez.

Not: Uygulama tarafından sağlanan CipherSpec değeri IBM MQtarafından bilinen bir CipherSpec değilse, IBM MQ yönetilen .NET istemcisi bunu göz önünde bulunmaz ve Windows sisteminin grup ilkesine dayalı olarak bağlantıyı kararlaştırır.

CipherSpec ayarlanması

CipherSpec' i ayarlamanın üç yolu vardır:

MQEnvironment .NET sınıfı
Aşağıdaki örnek, MQEnvironment sınıfıyla bir CipherSpec ' in nasıl ayarlanacağını göstermektedir.

```
MQEnvironment.SSLKeyRepository = "*USER";
```

```
MQEnvironment.ConnectionName = connectionName;
MQEnvironment.Channel = channelName;
MQEnvironment.properties.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED);
MQEnvironment.SSLCipherSpec = "TLS_RSA_WITH_AES_128_CBC_SHA";
```

TLS CipherSpec özelliği

Aşağıdaki örnek, MQQueueManager oluşturucusuna bir hashtable değiştirgesi ekleyerek CipherSpec ' in nasıl ayarlanacağını göstermektedir.

```
properties = new Hashtable();
properties.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED);
properties.Add(MQC.HOST_NAME_PROPERTY, hostName);
properties.Add(MQC.PORT_PROPERTY, port);
properties.Add(MQC.CHANNEL_PROPERTY, channelName);
properties.Add(MQC.SSL_CERT_STORE_PROPERTY, sslKeyRepository);
properties.Add(MQC.SSL_CIPHER_SPEC_PROPERTY, cipherSpec);
properties.Add(MQC.SSL_PEER_NAME_PROPERTY, sslPeerName);
properties.Add(MQC.SSL_RESET_COUNT_PROPERTY, keyResetCount);
queueManager = new MQQueueManager(queueManagerName, properties);
```

Windows grup ilkesi

Windows grup ilkesi yönetim konsolu aracılığıyla bir şifreleme takımı listesi yapılandırıldığında, SVRCONN kanal tanımı eşleşen bir CipherSpecbelirtmelidir. Eşleşen bir CipherSpec , "ANY_TLS12_OR_HIGHER" gibi genel bir değer ya da sıralı listeden kararlaştırılacak en yüksek şifreleme takımıyla eşlenen belirli bir değer olabilir. İstemci listesinin sırası değişirse SVRCONN CipherSpec yapılanışını değiştirmesi gerekmediğinden, .NET istemcileriyle kullanılmak üzere soysal CipherSpec değerlerinin kullanılması önerilir.

CCDT kullanımı

IBM MQ.NET yalnızca yerel bilgisayardaki İstemci Kanal Tanımlama Çizelgelerini (.TAB dosyaları) destekler. CipherSpec değeri ayarlanmış var olan CCDT dosyaları IBM MQ.NET bağlantıları için kullanılabilir. Ancak, istemci bağlantı kanalında ayarlanan CipherSpec değeri TLS iletişim kuralı sürümünü belirler ve Windows grup ilkesinde ayarlanan CipherSpec ile eşleşmelidir.

İlgili kavramlar

“IBM MQ ortamının ayarlanması” sayfa 563
Bir kuyruk yöneticisine bağlanmak için istemci bağlantısını kullanmadan önce IBM MQ ortamını ayarlamanız gerekir.

“Yönetilen .NET istemcisi için TLS desteği” sayfa 573
Yönetilen .NET istemcisi, TLS güvenli yuva iletişim kurallarını uygulamak için Microsoft .NET Framework kitaplıklarını kullanır. Microsoft System.Net.SecuritySslStream sınıfı, bağlı TCP yuvaları üzerinden bir akış olarak çalışır ve bu yuva bağlantısı üzerinden veri gönderir ve alır.

İlgili görevler

CipherSpecs ' nin Belirtilmesi

İlgili başvurular

MQEnvironment .NET sınıfı

Yönetilen .NET istemcisi için CipherSpec eşlemeleri

IBM MQ.NET arabirimi, yönetilen istemcinin bir kuyruk yöneticisiyle güvenli bir bağlantı kurmak için kullanması gereken TLS iletişim kuralının sürümünü belirlemek için kullanılan bir IBM MQ - Microsoft.NET eşleme tablosu sağlar.

SVRCONN kanalında bir CipherSpec belirtilirse, TLS anlaşması tamamlandıktan sonra kuyruk yöneticisi, CipherSpec öğesini istemci uygulamasının kullandığı kararlaştırılan CipherSpec ile eşleştirmeyi dener. Kuyruk yöneticisi eşleşen bir CipherSpecbulamazsa, iletişim AMQ9631hatasıyla başarısız olur.

IBM MQ.NET arabirimi, IBM MQ - Microsoft.NET CipherSpec eşleme tablosunu sağlar. Bu tablo, istemcinin kuyruk yöneticisiyle güvenli yuva bağlantısı kurmak için kullanmak istediği TLS iletişim kuralı sürümünü belirlemek için kullanılır. SSLCipherSpec değerine dayalı olarak SSLProtocol sürümü, kullandığınız Microsoft.NET Framework sürümüne bağlı olarak TLS 1.0ya da TLS 1.2olabilir.

Yanlış bir değer belirtilmesi SSL 3.0 ya da TLS 1.0 iletişim kurallarının kullanılmasıyla sonuçlanabileceği için doğru SSLCipherSpec değerini sağladığınızdan emin olun.

Çizelge 78. IBM MQ ve Microsoft.NET eşleme tablosu

IBM MQ CipherSpec	**Microsoft.NET CipherSpec**	**TLS sürümü**
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0
TLS_RSA_WITH_3DES_EDE_CBC_SHA [1]	TLS_RSA_WITH_3DES_EDE_CBC_SHA [1]	TLS 1.0
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256	TLS 1.2
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P384	TLS 1.2
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P384	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384	TLS 1.2
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P521	TLS 1.2
ECDHE_RSA_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2
ECDHE_RSA_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P384	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384	TLS 1.2
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P521	TLS 1.2

Çizelge 78. IBM MQ ve Microsoft.NET eşleme tablosu (devamı var)		
IBM MQ CipherSpec	**Microsoft.NET CipherSpec**	**TLS sürümü**
V 9.3.2 TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLS 1.3
V 9.3.2 TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLS 1.3
V 9.3.2 TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLS 1.3
V 9.3.2 TLS_AES_128_CCM_8_SHA256	TLS_AES_128_CCM_8_SHA256	TLS 1.3
V 9.3.2 TLS_AES_128_CCM_SHA256	TLS_AES_128_CCM_SHA256	TLS 1.3

Notlar:

1. Deprecated Bu CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA kullanımdan kaldırılmıştır. Ancak, bağlantı AMQ9288hatasıyla sonlandırılmadan önce 32 GB ' ye kadar veri aktarmak için kullanılabilir. Bu hatayı önlemek için üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckullanırken gizli anahtar sıfırlamasını etkinleştirmeniz gerekir.

İlgili kavramlar

“Yönetilen .NET istemcisi için TLS desteği” sayfa 573

Yönetilen .NET istemcisi, TLS güvenli yuva iletişim kurallarını uygulamak için Microsoft .NET Framework kitaplıklarını kullanır. Microsoft System.Net.SecuritySslStream sınıfı, bağlı TCP yuvaları üzerinden bir akış olarak çalışır ve bu yuva bağlantısı üzerinden veri gönderir ve alır.

Yönetilen .NET istemcisi için temel havuzlar

Yönetilen .NET istemcileri tarafından kullanılan anahtar havuzu Windows anahtar depodur. TLS anlaşması sırasında hem kimlik hem de güven için istemci uygulaması tarafından kullanılabilmesi için sertifikaların ve özel anahtarların kullanıcı ya da sistem anahtar deposunda bulunması gerekir.

İstemci tarafı

Uygulamada, anahtar havuzu için aşağıdaki değerlerden birini ayarlayabilirsiniz:

- "*USER": IBM MQ.NET , istemci sertifikalarını almak için yürürlükteki kullanıcının sertifika deposuna erişir.
- "*SYSTEM": IBM MQ.NET , sertifikaları almak için yerel bilgisayar hesabına erişir.

İstemcinin sertifikaları, kullanıcı ya da bilgisayar hesabının Sertifika Deposunda saklanmalıdır. Tüm sunucu (CA) sertifikaları, sertifika deposunun kök dizininde saklanmalıdır.

Not: Birden çok sertifikayı aşağıdaki biçimlerde tek bir dosyada saklayabilirsiniz:

- Kişisel Bilgi Alışverişi-PKCS #12 (.PFX, .P12)
- Şifreleme Iletisi Sözdizimi Standardı-PKCS #7 Sertifikaları (.P7B)
- Microsoft Diziselleştirilmiş Sertifika Deposu (.SST)

Yönetilen .NET istemcisi için sertifikaların kullanılması

İstemci sertifikaları için IBM MQ yönetilen .NET istemci, Windows anahtar deposuna erişir ve istemcinin sertifika etiketiyle eşleşen ya da dizgiyle eşleşen tüm sertifikalarını yükler.

Kullanılacak bir sertifika seçerken, IBM MQ yönetilen .NET istemci her zaman SSLStream TLS el sıkışması için ilk eşleşen sertifikayı kullanır.

Sertifika etiketine göre eşleşen sertifikalar

Sertifika etiketini ayarlarsanız, IBM MQ yönetilen .NET istemci, istemci sertifikasını tanımlamak için belirtilen etiket adıyla Windows sertifika deposunda arama yapar. Eşleşen tüm sertifikaları yükler ve listedeki ilk sertifikayı kullanır. Sertifika etiketini ayarlamak için iki seçenek vardır:

- Sertifika etiketi, MQEnvironment.CertificateLabelöğesine erişen MQEnvironment sınıfında ayarlanabilir.
- Sertifika etiketi, aşağıdaki örnekte gösterildiği gibi, MQQueueManager oluşturucusuyla giriş değiştirgesi olarak verilen bir HASH çizelgesi özelliklerinde de ayarlanabilir.

```
Hashtable properties = new Hashtable();
properties.Add("CertificateLabel", "mycert");
```

Ad ("CertificateLabel") ve değer büyük ve küçük harfe duyarlıdır.

Sertifikaları dizgiye göre eşleştirme

Sertifika etiketi ayarlanmazsa, "`ibmwebspheremq`" dizgisiyle eşleşen sertifika ve oturum açmış olan yürürlükteki kullanıcı (küçük harfli) aranır ve kullanılır.

İlgili görevler

İstemcinin kuyruk yöneticisine güvenli bir şekilde bağlanması

İlgili başvurular

MQEnvironment .NET sınıfı

SSLPEERNAME

SSLPEERNAME özniteliği, eşdüzey kuyruk yöneticisinden sertifikanın Ayırt Edici Adını (DN) denetlemek için kullanılır.

IBM MQ.NETuygulamasında uygulamalar, aşağıdaki örnekte gösterildiği gibi bir ayırt edici ad kalıbı belirtmek için SSLPEERNAME özelliğini kullanabilir.

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Diğer IBM MQ istemcilerine gelince, SSLPEERNAME isteğe bağlı bir parametredir.

SSLPEERNAME değeri ayarlanmazsa, IBM MQ.NET yönetilen istemcisi Uzak (Sunucu) sertifika doğrulaması yapmaz ve yönetilen istemci Uzak (/sunucu) sertifikasını olduğu gibi kabul eder.

SSLPEERNAME değerini ayarlama şekliniz, kullandığınız IBM MQ yığın ürünlerinden hangisini kullandığınıza bağlıdır.

IBM MQ classes for .NET

Aşağıdaki gibi üç seçenek vardır.

1. MQEnvironment sınıfında MQEnvironment.SSLPeerName öğesini ayarlayın.
2. `MQEnvironment.properties.Add(MQC.SSL_PEER_NAME_PROPERTY, value)`
3. `MQQueueManager (String queueManagerName, Hashtable properties)`kuyruk yöneticisi oluşturucusunu kullanın. Seçenek 2 için `Hashtable properties` içinde SSLPEERNAME değerini sağlayın.

XMS .NET

Bağlantı üreticisinde SSL eşdüzey adını ayarlayın:

```
ConnectionFactory.SetStringProperty(XMSC.WMQ_SSL_PEER_NAME, value);
```

WCF
URI ' ye noktalı virgülle ayrılmış alan olarak SslPeeradını ekleyin.

İlgili başvurular
MQEnvironment .NET sınıfı

Yönetilen .NET istemcisi için gizli anahtar sıfırlama ya da yeniden anlaşma
SSLStream sınıfı, gizli anahtar yeniden ayarlaması/yeniden anlaşma özelliğini desteklemiyor. Ancak, diğer IBM MQ istemcileriyle tutarlı olması için IBM MQ yönetilen .NET istemcisi, uygulamaların **SSLKeyResetCount**öğesini ayarlamasına izin verir.

Sınıra ulaşıldığında, IBM MQ.NET kuyruk yöneticisiyle bağlantıyı keser ve uygulamaya neden kodu olarak MQRC_CONNECTION_BROKEN kural dışı durumu bildirilir. Uygulamalar kural dışı durumu işlemeyi ve bağlantıları yeniden kurmayı seçebilir ya da IBM MQ.NET için MQCNO_RECONNECT seçeneğinin kuyruk yöneticisine otomatik olarak yeniden bağlanmasını etkinleştirebilir.

Otomatik istemci yeniden bağlanma olanağının etkinleştirilmesi, anahtar ilk duruma getirme sayısına ulaşıldığında var olan tüm bağlantıların devre dışı olduğu ve IBM MQ.NET istemcisinin tüm bağlantıları yeniden yarattığı anlamına gelir. Otomatik istemci yeniden bağlantısıyla ilgili ek bilgi için Otomatik istemci yeniden bağlantısıbaşlıklı konuya bakın.

İlgili kavramlar
SSL ve TLS gizli anahtarlarını sıfırlama

İptal denetimi
SSLStream sınıfı, sertifika iptal denetimini destekler.

İptal denetimi, sertifika zincirleme altyapısı tarafından otomatik olarak gerçekleştirilir. Bu, hem Online Certificate Status Protocol (OCSP) hem de Certificate Revocation (CRL) listeleri için geçerlidir. SSLStream sınıfı, yalnızca sertifikada belirtilen sunucuyu kullanan, yani sunucunun sertifikanın kendisi tarafından dikte edilen sertifikayı iptal eden sertifikayı kullanır. HTTP CDP uzantıları ve OCSP HTTP isteklerinin HTTP yetkili sunucusu aracılığıyla yetkili sunucu olarak kullanılması mümkündür.

İptal denetimini ayarlama şekliniz, kullandığınız IBM MQ yığın ürünlerinden hangisini kullandığınıza bağlıdır.

IBM MQ.NET
İptal denetimi, `MQEnvironment.cs` sınıf dosyasındaki **MQEnvironment.SSLCertRevocationCheck** özelliğine erişilerek ayarlanabilir.

XMS .NET
İptal denetimi, bağlantı üreticisi özelliği bağlamında aşağıdaki örnekte gösterildiği gibi ayarlanabilir.

```
ConnectionFactory.SetBooleanProperty(XMSC.WMQ_SSL_CERT_REVOCATION_CHECK, true);
```

WCF
İptal denetimi, aşağıdaki adlandırma kuralı kullanılarak URI üzerinde ayarlanabilir.

```
"SslCertRevocationCheck=true"
```

Yönetilen IBM MQ .NET için TLS ' nin yapılandırılması
Yönetilen IBM MQ .NET için TLS ' nin yapılandırılması, imzalayıcı sertifikalarının oluşturulmasından, ardından sunucu tarafının, istemci tarafının ve uygulama programının yapılandırılmasından oluşur.

Bu görev hakkında

TLS ' yi yapılandırmak için önce uygun imzalayıcı sertifikalarını oluşturmanız gerekir. İmzalayıcı sertifikaları, kendinden imzalı ya da bir sertifika yetkilisi tarafından sağlanan sertifikalar olabilir. Kendinden imzalı sertifikalar bir geliştirme, test veya üretim öncesi sistemde kullanılabilse de, bunları üretim sisteminde kullanmayın. Bir üretim sisteminde, güvenilen bir dış sertifika kuruluşundan (CA) edindiğiniz sertifikaları kullanın.

Yordam

1. İmzalayıcı sertifikalarını oluşturun.
   a) Kendinden onaylı sertifikalar oluşturmak için IBM MQ ile birlikte sağlanan aşağıdaki araçlardan birini kullanın:

      strmqikm GUI 'sini kullanın ya da komut satırından **runmqckm** ya da **runmqakm** komutunu kullanın. Bu araçları kullanma hakkında daha fazla bilgi için bkz. **runmqckm**, **runmqakm**ve **strmqikm** dijital sertifikaları yönetmek için.

   b) Kuyruk yöneticisine ve istemcilere ilişkin sertifikaları bir sertifika yetkilisinden (CA) almak için Sertifika yetkilisinden kişisel sertifika almabaşlıklı konudaki yönergeleri izleyin.
2. Sunucu tarafını yapılandırın.
   a) İstemcinin bir kuyruk yöneticisine güvenli bir şekilde bağlanmasıbaşlıklı konuda açıklandığı gibi IBM Global Security Kit (GSKit)kullanarak kuyruk yöneticisinde TLS ' yi yapılandırın.
   b) SVRCONN kanalı TLS özniteliklerini ayarla:
      - **SSLCAUTH** değerini "REQUIRED/OPTIONAL"olarak ayarlayın.
      - **SSLCIPH** ayarını uygun bir CipherSpecdeğerine ayarlayın.

      Daha fazla bilgi için bkz.“Yönetilmeyen .NET istemcisi için TLS ' yi etkinleştirme” sayfa 572.
3. İstemci tarafını yapılandırın.
   a) İstemci sertifikalarını Windows sertifika deposuna (Kullanıcı/Bilgisayar hesabı altında) aktarın.

      IBM MQ .NET , Windows sertifika deposundan istemci sertifikalarına erişir; bu nedenle, IBM MQ ile güvenli yuva bağlantısı kurmak için sertifikalarınızı Windows sertifika deposuna aktarmanız gerekir. Windows anahtar deposuna nasıl erişileceği ve istemci tarafı sertifikalarının nasıl içe aktarılacağı hakkında daha fazla bilgi için Sertifikaları ve özel anahtarları içe aktarma ya da dışa aktarmabaşlıklı konuya bakın.
   b) CertificateLabel öğesini, İstemcinin kuyruk yöneticisine güvenli bir şekilde bağlanmasıkonusunda açıklandığı gibi sağlayın.
   c) Gerekirse, CipherSpec' i ayarlamak için Windows Grup İlkesini düzenleyin ve Windows Grup İlkesi güncellemelerinin yürürlüğe girmesi için bilgisayarı yeniden başlatın.
4. Uygulama programını yapılandırın.
   a) Bağlantıyı güvenli bağlantı olarak göstermek için MQEnvironment ya da SSLCipherSpec değerini ayarlayın.

      Belirlediğiniz değer, kullanılmakta olan protokolü (TLS) tanımlamak için kullanılır. CipherSpec kümesi, desteklenen SSLProtocol sürümünün CipherSpecs ' inden biri olmalıdır ve tercihen Windows Grup İlkesinde belirtilenle aynı olabilir. (Desteklenen SSLProtocol sürümü, kullanılan .NET çerçevesine bağlıdır. SSLProtocol sürümü, kullandığınız Microsoft .NET Framework sürümüne bağlı olarak TLS 1.0ya da TLS 1.2olabilir.)

      Not: Uygulama tarafından sağlanan CipherSpec değeri, IBM MQtarafından bilinen bir CipherSpec değilse, IBM MQ yönetilen .NET istemcisi bunu göz önünde bulunmaz ve Windows sisteminin grup ilkesine dayalı olarak bağlantıyı kararlaştırır.
   b) SSLKeyRepository özelliğini "*SYSTEM" ya da "*USER"olarak ayarlayın.
   c) İsteğe bağlı: SSLPEERNAME değerini, sunucu sertifikasının ayırt edici adına (DN) ayarlayın.
   d) CertificateLabel öğesini, İstemcinin kuyruk yöneticisine güvenli bir şekilde bağlanmasıkonusunda açıklandığı gibi sağlayın.
   e) KeyResetCount, CertificationRevocationCheck gibi, gerek duyduğunuz diğer isteğe bağlı parametreleri ayarlayın ve FIPS ' yi etkinleştirin.

TLS iletişim kuralı ve TLS anahtar havuzunun nasıl ayarlanacağına ilişkin örnekler

Temel .NETiçin, aşağıdaki örnekte gösterildiği gibi, MQEnvironment sınıfı aracılığıyla TLS iletişim kuralı ve TLS anahtar havuzunu ayarlayabilirsiniz:

```
MQEnvironment.SSLCipherSpec = "TLS_RSA_WITH_AES_128_CBC_SHA256";
MQEnvironment.SSLKeyRepository = "*USER";

MQEnvironment.properties.Add(MQC.SSL_CIPHER_SPEC_PROPERTY, "TLS_RSA_WITH_AES_128_CBC_SHA256")
```

Diğer bir seçenek olarak, aşağıdaki örnekte gösterildiği gibi, MQQueueManager oluşturucusunun bir parçası olarak bir hashtable belirterek TLS iletişim kuralı ve TLS anahtar havuzunu ayarlayabilirsiniz.

```
Hashtable properties = new Hashtable();
properties.Add(MQC.SSL_CERT_STORE_PROPERTY, sslKeyRepository);
properties.Add(MQC.SSL_CIPHER_SPEC_PROPERTY, "TLS_RSA_WITH_AES_128_CBC_SHA256")
```

Sonraki adım

IBM MQ .NET yönetilen TLS uygulamalarını geliştirmeye başlama hakkında daha fazla bilgi için bkz.“Basit bir uygulama yazılması” sayfa 582.

İlgili başvurular

MQEnvironment .NET sınıfı

KeyResetSayısı (MQLONG)

AIX, Linux, and Windows için Federal Bilgi İşleme Standartları (FIPS)

Basit bir uygulama yazılması

Bağlantı üreticileri için SSL özelliklerini ayarlama, kuyruk yöneticisi eşgörünümü yaratma, bağlantı, oturum ve hedef yaratma ve sınama iletisi gönderme örnekleri de içinde olmak üzere basit bir IBM MQ yönetilen .NET TLS uygulaması yazmaya ilişkin ipuçları.

Başlamadan önce

Öncelikle “Yönetilen IBM MQ .NET için TLS ' nin yapılandırılması” sayfa 580içinde açıklandığı gibi yönetilen IBM MQ.NET için TLS ' yi yapılandırmanız gerekir.

Temel .NETiçindeki uygulama programı yapılandırması için, MQEnvironment sınıfını kullanarak ya da MQQueueManager oluşturucusunun bir parçası olarak bir hashtable belirterek SSL özelliklerini ayarlayın.

XMS .NETiçindeki uygulama programı yapılandırması için, bağlantı üreticileri özellik bağlamında SSL özelliklerini ayarlarsanız.

Yordam

1. Bağlantı üreticileri için SSL özelliklerini aşağıdaki örneklerde gösterildiği gibi ayarlayın.

   Örnek: IBM MQ.NET

   ```
   properties = new Hashtable();
   properties.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED);
   properties.Add(MQC.HOST_NAME_PROPERTY, hostName);
   properties.Add(MQC.PORT_PROPERTY, port);
   properties.Add(MQC.CHANNEL_PROPERTY, channelName);
   properties.Add(MQC.SSL_CERT_STORE_PROPERTY, sslKeyRepository);
   properties.Add(MQC.SSL_CIPHER_SPEC_PROPERTY, cipherSpec);
   properties.Add(MQC.SSL_PEER_NAME_PROPERTY, sslPeerName);
   properties.Add(MQC.SSL_RESET_COUNT_PROPERTY, keyResetCount);
   properties.Add("CertificateLabel", "ibmwebspheremq");
   MQEnvironment.SSLCertRevocationCheck = sslCertRevocationCheck;
   ```

XMS .NET örneği

```
cf.SetStringProperty(XMSC.WMQ_SSL_KEY_REPOSITORY, "sslKeyRepository");
cf.SetStringProperty(XMSC.WMQ_SSL_CIPHER_SPEC, cipherSpec);
cf.SetStringProperty(XMSC.WMQ_SSL_PEER_NAME, sslPeerName);
cf.SetIntProperty(XMSC.WMQ_SSL_KEY_RESETCOUNT, keyResetCount);
cf.SetBooleanProperty(XMSC.WMQ_SSL_CERT_REVOCATION_CHECK, true);
```

2. Aşağıdaki örneklerde gösterildiği gibi, kuyruk yöneticisi yönetim ortamını, bağlantılarını, oturumu ve hedefi yaratın.

   MQ .NET örneği

   ```
   queueManager = new MQQueueManager(queueManagerName, properties);
   Console.WriteLine("done");

   // accessing queue
   Console.Write("Accessing queue " + queueName + ".. ");
   queue = queueManager.AccessQueue(queueName, MQC.MQOO_OUTPUT +
   MQC.MQOO_FAIL_IF_QUIESCING);
   Console.WriteLine("done");
   ```

   XMS .NET örneği

   ```
   connectionWMQ = cf.CreateConnection();
   // Create session
   sessionWMQ = connectionWMQ.CreateSession(false, AcknowledgeMode.AutoAcknowledge);

   // Create destination
   destination = sessionWMQ.CreateQueue(destinationName);

   // Create producer
   producer = sessionWMQ.CreateProducer(destination);
   ```

3. Aşağıdaki örneklerde gösterildiği gibi bir ileti gönderin.

   MQ .NET örneği

   ```
   // creating a message object
   message = new MQMessage();
   message.WriteString(messageString);

   // putting messages continuously
   for (int i = 1; i <= numberOfMsgs; i++)
   {
   Console.Write("Message " + i + " <" + messageString + ">.. ");
   queue.Put(message);
   Console.WriteLine("put");
   }
   ```

   XMS .NET örneği

   ```
   textMessage = sessionWMQ.CreateTextMessage();
   textMessage.Text = simpleMessage;
   producer.Send(textMessage);
   ```

4. TLS bağlantısını doğrulayın.

   TLS bağlantısının kurulduğunu ve düzgün çalıştığını doğrulamak için kanal durumunu denetleyin.

SSLStream için izleme yapılandırılıyor
SSLStream sınıfıyla ilgili izleme olaylarını ve iletileri yakalamak için, uygulamanıza ilişkin uygulama yapılandırma dosyasına sistem tanılamaları için bir yapılandırma bölümü eklemeniz gerekir.

Bu görev hakkında

Not:

Bu görev yalnızca IBM MQ classes for .NET Framework için geçerlidir. Uygulama yapılandırma dosyası IBM MQ classes for .NET (.NET Standard ve .NET 6 kitaplıkları) içinde desteklenmez.

Uygulama yapılandırma dosyasına sistem tanılamaları için bir yapılandırma bölümü eklemezseniz, IBM MQ yönetilen .NET istemci TLS ve SSLStream sınıfıyla ilgili herhangi bir olay, izleme ya da hata ayıklama noktası yakalamaz.

Not: IBM MQ izlemenin **strmqtrc** kullanılarak başlatılması, gerekli tüm TLS izlemesini yakalamamaktadır.

Yordam

1. Uygulama projeniz için bir uygulama yapılandırma (App.Config) dosyası oluşturun.
2. Aşağıdaki örnekte gösterildiği gibi bir sistem tanılama yapılandırması bölümü ekleyin.

```
<system.diagnostics>
           <sources>
                 <source name="System.Net" tracemode="includehex">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
                 <source name="System.Net.Sockets">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
                 <source name="System.Net.Cache">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
                 <source name="System.Net.Security">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
                 <source name="System.Security">
                       <listeners>
                             <add name="ExternalSourceTrace"/>
                       </listeners>
                 </source>
           </sources>
           <switches>
                 <add name="System.Net" value="Verbose"/>
                 <add name="System.Net.Sockets" value="Verbose"/>
                 <add name="System.Net.Cache" value="Verbose"/>
                 <add name="System.Security" value="Verbose"/>
                 <add name="System.Net.Security" value="Verbose"/>
           </switches>

           <sharedListeners>
                 <add name="ExternalSourceTrace" type="IBM.WMQ.ExternalSourceTrace,
amqmdnet, Version=n.n.n.n, Culture=neutral, PublicKeyToken=dd3cb1c9aae9ec97" />
            </sharedListeners>
          <trace autoflush="true"/>
     </system.diagnostics>
```

Uyarı: add name girişinin `Version` alanının, kullanılmakta olan .net `amqmdnet.dll` dosyasının hangi sürümü olması gerekiyorsa o olması gerekir.

İlgili görevler

Uygulama yapılandırma dosyası kullanarak IBM MQ classes for .NET Framework istemcilerini izleme

Yönetilen .NET içinde TLS uygulanması için örnek uygulamalar

WCF için IBM MQ classes for .NET, XMS .NET ve IBM MQ özel kanalında yönetilen .NET için TLS uygulamasını göstermek üzere örnek uygulamalar sağlanır.

Aşağıdaki çizelge, örnek uygulamaların yerini göstermektedir. *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Çizelge 79. Yönetilen .NET içinde TLS uygulamak için örnek uygulamaların konumu	
IBM MQ.NET yığın ürünü	**Örneklerin konumu**
Taban.NET	*MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\base\SimplePut\SimplePut.cs *MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\base\SimpleGet\SimpleGet.cs
XMS .NET	*MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\xms\simple\wmq\SimpleProducer\SimpleProducer.cs *MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\xms\simple\wmq\SimpleConsumer\SimpleConsumer.cs
WCF için IBM MQ özel kanalı	*MQ_INSTALLATION_PATH*\Tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\MQMessagingOneWayService.cs

Windows .NET Monitor Olanağının Kullanılması

.NET Monitor, IBM MQ tetikleyici izleme programına benzer bir uygulamadır.

Önemli: Önemli bilgiler için bkz. Yalnızca Windows üzerinde birincil kuruluşla kullanılabilen özellikler .

İzlenen bir kuyrukta bir ileti alındığında ve daha sonra bu iletiyi işleyen .NET bileşenleri yaratabilirsiniz. .NET Monitor, **runmqdnm** komutu tarafından başlatılır ve **endmqdnm** komutu tarafından durdurulur. Bu komutların ayrıntıları için bkz. runmqdnm ve endmqdnm.

.NET Monitor 'u kullanmak için, amqmdnm.dlliçinde tanımlanan IMQObjectTrigger arabirimini gerçekleştiren bir bileşen yazın.

Bileşenler işlemsel ya da hareket dışı olabilir. Bir hareket bileşeni System.EnterpriseServices.ServicedComponent öğesinden edinmeli ve RequiresTransaction ya da SupportsTransactionolarak kaydedilmelidir. .NET Monitor zaten bir hareket başlatmış olduğu için RequiresNew olarak kaydedilmemelidir.

Bileşen, **runmqdnm**içinden MQQueueManager, MQQueue ve MQMessage nesnelerini alır. runmqdnm başlatıldığında, *-u* komut satırı seçeneği kullanılarak belirtildiyse bir Kullanıcı Parametresi dizgisi de alabilir. Bileşeninizin, bir MQMessage nesnesinde izlenen kuyruğa gelen bir iletinin içeriğini aldığını unutmayın. Kuyruk yöneticisine bağlanmak, kuyruğu açmak ya da iletinin kendisini almak zorunda değildir. Bundan sonra, bileşen iletiyi uygun şekilde işlemeli ve denetimi .NET Monitor 'a döndürmelidir.

Bileşeniniz bir hareket bileşeni olarak yazıldıysa, System.EnterpriseServices.ServicedComponenttarafından sağlanan olanakları kullanarak hareketi kesinleştirmek ya da geriye işlemek için kayıt olur.

Bileşen, MQQueueManager ve MQQueue nesnelerini ve iletiyi aldıkça, o iletiye ilişkin tam bağlam bilgileri içerir ve örneğin, IBM MQile ayrı olarak bağlantı kurmasına gerek kalmadan aynı kuyruk yöneticisinde başka bir kuyruk açabilir.

Windows *Örnek kod parçaları*

Bu konuda, .NET Monitor 'dan ileti alan ve bu iletiyi yazdıryan bileşenlere ilişkin iki örnek yer alır; bu bileşenlerden biri işlemsel işlemeyi, diğeri işlemsel olmayan işlemeyi kullanır. Üçüncü bir örnek, her iki örnek için de geçerli olan ortak yardımcı program yordamlarını gösterir. Tüm örnekler C# ' de.

Örnek 1: İşlemsel işleme

```
/********************************************************************/
/* 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;
    }
  }
}
```

Örnek 2: Hareket dışı işleme

```
/*********************************************************************/
/* 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());
      }
    }
  }
}
```

Örnek 3: Ortak yordamlar

```
/*********************************************************************/
/* 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;
    }
  }
}
```

IBM MQ .NET programlarının derlenmesi

Çeşitli dillerde yazılan .NET uygulamalarını derlemek için örnek komutlar.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

IBM MQ classes for .NETkullanarak bir C# uygulaması oluşturmak için şu komutu kullanın:

```
csc /t:exe /r:System.dll /r:amqmdnet.dll /lib: MQ_INSTALLATION_PATH\bin /out:MyProg.exe
MyProg.cs
```

IBM MQ classes for .NETkullanarak bir Visual Basic uygulaması oluşturmak için aşağıdaki komutu kullanın:

```
vbc /r:System.dll /r: MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:MyProg.exe MyProg.vb
```

IBM MQ classes for .NETkomutunu kullanarak bir Yönetilen C++ uygulaması oluşturmak için aşağıdaki komutu kullanın:

```
cl /clr MQ_INSTALLATION_PATH\bin Myprog.cpp
```

Diğer diller için, dil sağlayıcısı tarafından sağlanan belgelere bakın.

Bağımsız IBM MQ .NET istemcisinin kullanılması

IBM MQ .NET istemcisi, uygulamalarınızı çalıştırmak için üretim sistemlerinde tam IBM MQ istemci kuruluşunu kullanmanız gerekmeden bir IBM MQ .NET düzeneğini paketleme ve devreye alma yeteneği sunar.

Başlamadan önce

V 9.3.1 IBM MQ 9.3.1'den varsayılan konuma kurulan `amqmdnetstd.dll` istemci kitaplığı .NET 6' e dayalıdır. .NET Standard tabanlı `amqmdnetstd.dll` istemci kitaplığı, IBM MQ istemcisi kuruluş paketinde yeni bir konuma taşındı ve artık aşağıdaki konumlarda kullanılabilir:

- Windows Windowssistemlerinde: *MQ_INSTALLATION_PATH*`\bin\netstandard2.0`
- Linux Linuxsistemlerinde: *MQ_INSTALLATION_PATH*`\lib64\netstandard2.0`

LTS IBM MQ 9.3.0 Long Term Supportiçin `amqmdnetstd.dll` kitaplığı aşağıdaki yerlerde bulunur:

- Windows Windowssistemlerinde: *MQ_INSTALLATION_PATH*\bin. Örnek uygulamalar *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/baseiçine kurulur.
- Linux Linuxsistemlerinde: *MQ_INSTALLATION_PATH*/lib64 path. .NET örnekleri *MQ_INSTALLATION_PATH*/samp/dotnet/samples/cs/core/baseiçinde yer almaktadır.

LTS Stabilized amqmdnet.dll kitaplığı sağlanmaya devam eder, ancak bu kitaplık dengelenir; başka bir değişle, kitaplığa yeni özellikler eklenmez. En son özelliklerden herhangi biri için amqmdnetstd.dll kitaplığına geçmeniz gerekir. Ancak, IBM MQ 9.1 Long Term Support ya da Continuous Delivery yayın düzeylerinde amqmdnet.dll kitaplığını kullanmaya devam edebilirsiniz.

Bu görev hakkında

IBM MQ .NET uygulamalarınızı, tam IBM MQ istemcisinin kurulu olduğu bir makinede oluşturabilir ve daha sonra uygulamanızla birlikte IBM MQ .NET düzeneğini amqmdnetstd.dllpaketleyebilir ve üretim sistemlerinde konuşlandırabilirsiniz.

Oluşturup devreye aldığınız uygulamalar, geleneksel .NET uygulamaları, Hizmetler ya da Microsoft Azure Web/Worker uygulamaları olabilir

Bu tür konuşlandırmalarda, IBM MQ .NET istemcisi yalnızca bir kuyruk yöneticisine ilişkin yönetilen bağlantı kipini destekler. Bu iki kip tam bir IBM MQ istemcisi kuruluşu gerektirdiğinden, sunucu bağ tanımları ve yönetilmeyen istemci kipi bağlanırlığı kullanılamaz. Diğer iki kipi kullanma girişimi bir uygulama kural dışı durumuyla sonuçlanır.

Yordam

Uygulamalarda IBM MQ .NET istemci düzeneğine başvurma

- Uygulamanızdaki amqmdnetstd.dll düzeneğine, önceki yayınlarda olduğu gibi başvuruda bulunun.

  amqmdnetstd.dll düzeneğinin uygulamanın bin dizinine kopyalandığından emin olmak için amqmdnetstd.dll düzeneğinin **CopyLocal** özelliğini True olarak ayarlayın. Bu özelliğin ayarlanması, uygulama paketleme aracının üretim sistemlerinde devreye alınması için gerekli ikili dosyaları ve Microsoft Azure PaaS bulut ortamlarını paketlemesine de yardımcı olur.

Genel hareket desteği eklenmesi

- Uygulamanızın WMQDotnetXAMonitor Monitor uygulamasını uygulamanın kendisiyle birlikte makinede konuşlandırdığını doğrulayın.

  Bir uygulama IBM MQ .NET tarafından yönetilen genel hareket özelliğini kullanıyorsa, uygulamanın kendisiyle birlikte makineye WMQDotnetXAMonitor ' ı da konuşlandırmalıdır. Belirsiz hareketlerin kurtarılması için bu yardımcı program gereklidir.

İzleme başlatılıyor ve durduruluyor

- Yalnızca IBM MQ classes for .NET Framework için, uygulama yapılandırma dosyasını ve IBM MQ ' e özgü bir izleme yapılandırma dosyasını kullanarak izlemeyi başlatmak ve durdurmak için Bir IBM MQ classes for .NET Framework istemcisinin bir uygulama yapılandırma dosyasını kullanarak izlenmesibaşlıklı konuya bakın.

  Tam IBM MQ istemcisi kuruluşu olmadığından, izlemeyi başlatmak ve durdurmak için kullanılan standart araçlar (**strmqtrc** ve **endmqtrc**) kullanılamadığından, uygulama yapılandırma dosyasını ve IBM MQ ' a özgü bir izleme yapılandırma dosyasını kullanmanız gerekir.

  Notlar:

  – İzleme oluşturmanın bu yolu, .NET yeniden dağıtılabilir yönetilen istemcinin yanı sıra bağımsız .NET istemcisi için de geçerlidir. Bkz. .NET uygulama yürütme ortamı- Windows yalnızca.

  – Uygulama yapılandırma dosyası IBM MQ classes for .NET (.NET Standard ve .NET 6 kitaplıkları) içinde desteklenmez. IBM MQ classes for .NET (.NET Standard ve .NET 6 kitaplıkları) için izlemeyi etkinleştirmek üzere **MQDOTNET_TRACE_ON** ortam değişkenini kullanırsınız. Ortam değişkenlerini kullanarak IBM MQ .NET uygulamalarının izlenmesibaşlıklı konuya bakın.

- V 9.3.3
  `mqclient.ini` dosyasını kullanarak ve İzleme kısmı için uygun özellikleri ayarlayarak izlemeyi başlatın ve durdurun.

  Bkz. mqclient.ini ile IBM MQ .NET uygulamalarını izleme.

  IBM MQ 9.3.3' den `mqclient.ini` dosyasını kullanarak ve İzleme kısmına ilişkin uygun özellikleri ayarlayarak izlemeyi yapılandırabilirsiniz. Ayrıca, `mqclient.ini` dosyasıyla izlemeyi dinamik olarak etkinleştirebilir ve devre dışı bırakabilirsiniz.

Uygulama yapılandırma dosyasında bağ tanımı yeniden yönlendirmesinin etkinleştirilmesi

- IBM MQ .NET yapıbiriminin derleme zamanı bağlama başvurusunu yapıbiriminin daha sonraki bir sürümüne etkinleştirmek için, <dependentAssembly> özelliğini uygulama yapılandırma dosyasına ekleyin.

  `app.config` dosyasındaki aşağıdaki örnek parçacık, IBM MQ .NET yapıbiriminin IBM MQ 8.0.0 Fix Pack 2 (`8.0.0.2`) sürümü kullanılarak derlenen bir uygulamayı yeniden yönlendirir, ancak daha sonra bir düzeltme paketi (IBM MQ 8.0.0 Fix Pack 3), IBM MQ.NET düzeneğini `8.0.0.3`olarak güncelleyen bir düzeltme paketi uygulandı.

```
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <!-- amqmdnet related binding redirect -->
    <dependentAssembly>
      <assemblyIdentity name="amqmdnet"
                        publicKeyToken="dd3cb1c9aae9ec97"
                        culture="neutral" />
      <codeBase version="8.0.0.2"
            href="file:///amqmdnet.dll"/>
      <bindingRedirect oldVersion="1.0.0.3-8.0.0.2"
                       newVersion="8.0.0.3"/>
      <publisherPolicy apply="no" />
    </dependentAssembly>
  </assemblyBinding>
</runtime>
```

İlgili kavramlar

“kurmaIBM MQ classes for .NET” sayfa 534
Örnekler de içinde olmak üzere IBM MQ classes for .NET, Windows ve Linux üzerinde IBM MQ ile birlikte kurulur.

Yeniden dağıtılabilen istemciler

.NET uygulama yürütme ortamı-yalnızca Windows

İlgili görevler

“WMQDotnetXAMonitor uygulamasının kullanılması” sayfa 553
IBM MQ .NET istemcisi, tamamlanmamış dağıtılmış hareketleri kurtarmak için kullanabileceğiniz bir XA Monitor uygulaması (WmqDotnetXAMonitor) sağlar. WmqDotnetXAMonitor uygulaması, hareketlerin belirsiz olduğu kuyruk yöneticisiyle bağlantı kurar ve ayarladığınız değiştirgelere dayalı olarak hareketi çözer.

IBM MQ .NET uygulamalarını izleme

V 9.3.0 OutboundSNI özellik

Bir özellik ya da ortam değişkeni kullanarak bir uygulamada **OutboundSNI** özelliğini ayarlayabilirsiniz.

IBM MQ 9.3.0'den MQC.OUTBOUND_SNI_PROPERTY , kuyruk yöneticisine bağlanmak için `MQQueueManager` sınıfı kullanılırken bir HASH çizelgesi kullanılıyor.

MQC.OUTBOUND_SNI_PROPERTY aşağıdaki değerleri alır:

- MQC.OUTBOUND_SNI_CHANNEL
- MQC.OUTBOUND_SNI_HOSTNAME, "HOSTNAME" ile eşlenir
- MQC.OUTBOUND_SNI_ASTERISK, "*" ile eşlenir

Buna ek olarak, aşağıdaki değerleri alan MQOUTBOUND_SNI ortam değişkenini kullanarak **OutboundSNI** özelliğini ayarlayabilirsiniz:

- Kanal
- ANASİSTEM ADI
- *

ve diğer `mqclient.ini` özelliklerinde olduğu gibi, `App.config` dosyasında **OutboundSNI** değerini ayarlayın.

Not: Özellik varsayılan olarak MQC.OUTBOUND_SNI_CHANNEL OUTBOUND_SNI_CHANNEL.

Yönetilen düğümde **OutboundSNI** özelliğini ayarlamak için öncelik sırası:

1. Uygulama düzeyi özelliği
2. Ortam değişkeni

Yönetilmeyen düğümdeki **OutboundSNI** özelliği için yalnızca `mqclient.ini` desteklenir.

`App.config` dosyasında ayarlanan özellikler yalnızca .NET Framework uygulamaları için geçerlidir.

Uygulama düzeyinde ya da `App.config` dosyasında geçerli olmayan bir değer sağlarsanız, MQRC_OUTBOUND_SNI_NOT_VALID dönüş kodu yayınlanır.

Geçerli olmayan bir ortam değişkeni ayarlarsanız ya da `mqclient.ini` dosyasında geçerli olmayan bir değer sağlarsanız, CHANNEL varsayılan değeri kullanılır.

OutboundSNI ve birden çok sertifika

IBM MQ , birden çok sertifika işlevselliği sağlamak için SNI üstbilgisini kullanır. Bir uygulama CERTLABL alanı aracılığıyla farklı bir sertifika kullanacak şekilde yapılandırılmış bir IBM MQ kanalına bağlanıyorsa, uygulamanın **OutboundSNI** CHANNEL ayarıyla bağlanması gerekir.

OutboundSNI ayarı CHANNEL dışında bir uygulama yapılandırılmış bir sertifika etiketiyle bir kanala bağlanırsa, uygulama MQRC_SSL_INITIALIZATION_ERROR ile reddedilir ve kuyruk yöneticisi hata günlüklerine AMQ9673 iletisi yazdırılır.

IBM MQ ' ın birden çok sertifika işlevini nasıl sağladığı hakkında daha fazla bilgi için bkz. Nasıl IBM MQ birden çok sertifika yeteneği sağlar .

XMS .NET uygulamalarının geliştirilmesi

IBM MQ Message Service Client (XMS) for .NET (XMS .NET), Java Message Service (JMS) ile aynı arabirim kümesine sahip XMS adlı bir uygulama programlama arabirimi (API) sağlar. API. IBM MQ Message Service Client (XMS) for .NET , herhangi bir .NET uyumlu dil tarafından kullanılabilen, tam olarak yönetilen bir XMSuygulamasını içerir.

Bu görev hakkında

XMS Destek:

- Noktadan noktaya ileti sistemi
- İleti alışverişi yayınla/abone ol
- Zamanuyumlu ileti teslimi
- Zamanuyumsuz ileti teslimi

Bir XMS uygulaması, aşağıdaki uygulama tipleriyle ileti değiş tokuşu yapabilir:

- XMS uygulaması
- IBM MQ classes for JMS uygulaması
- Yerel bir IBM MQ uygulaması
- IBM MQ varsayılan ileti alışverişi sağlayıcısını kullanan bir JMS uygulaması

Bir XMS uygulaması, aşağıdaki ileti sistemi sunucularından herhangi birine bağlanabilir ve bunları kullanabilir:

IBM MQ Kuyruk Yöneticisi
Uygulama bağ tanımlarında ya da istemci kipinde bağlanabilir.

WebSphere Application Server service integration bus
Uygulama doğrudan TCP/IP bağlantısı kullanabilir ya da TCP/IP üzerinden HTTP kullanabilir.

IBM Integration Bus
İletiler, WebSphere MQ Real-Time Transportkullanılarak uygulama ile aracı arasında aktarılır. İletiler, WebSphere MQ Multicast Transportkullanılarak uygulamaya teslim edilebilir.

Bir IBM MQ kuyruk yöneticisine bağlanarak, bir XMS uygulaması IBM Integration Busile iletişim kurmak için WebSphere MQ Enterprise Transport komutunu kullanabilir. Alternatif olarak, bir XMS uygulaması IBM MQile bağlantı kurarak yayınlayabilir ve abone olabilir.

IBM MQ 9.1.1'den IBM MQ , Windows ortamlarındaki uygulamalar için .NET Core ' yi destekler. Daha fazla bilgi için bkz “kurmaIBM MQ classes for XMS .NET” sayfa 595.

IBM MQ 9.1.2'den IBM MQ , Linux ortamlarındaki uygulamalar için .NET Core ' yi destekler.

IBM MQ 9.1.4' den XMS .NET yönetilen uygulamalar, kümelenmiş kuyruk yöneticileri arasındaki bağlantıları otomatik olarak dengeleyebilirler. Hem .NET Framework hem de .NET Standard kitaplıkları desteklenir. Daha fazla bilgi için bkz. Tek biçimli kümeler hakkında ve Otomatik uygulama dengeleme.

İlgili görevler
IBM Desteği ile iletişim kurulması
XMS .NETproblems sorunlarının giderilmesi

XMS tarafından desteklenen ileti sistemi stilleri

XMS , noktadan noktaya iletişim ve yayınlama/abone olma ileti sistemi stillerini destekler.

İleti sistemi stillerine ileti sistemi etki alanları da denir.

Noktadan noktaya ileti sistemi

Sık kullanılan bir noktadan noktaya ileti sistemi biçimi kuyruğa alma özelliğini kullanır. En basit durumda, bir uygulama hedef kuyruğu tanımlayarak, örtük ya da belirtik olarak başka bir uygulamaya ileti gönderir. Temel ileti sistemi ve kuyruğa alma sistemi, iletiyi gönderen uygulamadan alır ve iletiyi hedef kuyruğuna yönlendirir. Bundan sonra, alan uygulama iletiyi kuyruktan alabilir.

Temel ileti sistemi ve kuyruğa alma sistemi IBM Integration Busiçeriyorsa, IBM Integration Bus bir iletiyi eşleyebilir ve iletinin kopyalarını farklı kuyruklara yönlendirebilir. Sonuç olarak, iletiyi birden çok uygulama alabilir. IBM Integration Bus bir iletiyi dönüştürebilir ve iletiye veri ekleyebilir.

Noktadan noktaya ileti sisteminin temel özelliği, bir uygulamanın ileti gönderdiğinde iletiyi yerel bir kuyruğa koymasıdır. Temel ileti sistemi ve kuyruğa alma sistemi, iletinin gönderileceği hedef kuyruğu belirler. Alan uygulama iletiyi hedef kuyruktan alır.

İleti alışverişi yayınla/abone ol

Yayınlama/abone olma ileti sisteminde iki tip uygulama vardır: yayınlayıcı ve abone.

Yayınlayıcı , yayın iletileri biçiminde bilgi sağlar. Bir yayınlayıcı bir iletiyi yayınladığında, ileti içindeki bilgilerin konusunu tanımlayan bir konuyu belirtir.

Abone , yayınlanan bilgilerin tüketicisidir. Bir abone, abonelikler oluşturarak ilgilendiği konuları belirtir.

Yayınlama/abone olma sistemi, yayıncılardan ve abonelerden yayınlar alır. Yayınları abonelere yönlendirir. Bir abone, yalnızca abone olduğu konularla ilgili yayınları alır.

Yayınlama/abone olma ileti sisteminin temel özelliği, bir yayıncının bir iletiyi yayınlarken bir konuyu belirlemektir. Aboneleri tanımlamaz. Abonesi olmayan bir konuda ileti yayınlandıysa, ileti hiçbir uygulama tarafından alınmaz.

Bir uygulama hem yayınlayıcı hem de abone olabilir.

XMS nesne modeli

XMS API, nesne yönelimli bir arabirimdir. XMS nesne modeli, JMS 1.1 nesne modelini temel alır.

Ana XMS sınıfları

Ana XMS sınıfları ya da nesne tipleri şunlardır:

ConnectionFactory
`ConnectionFactory` nesnesi, bir bağlantıya ilişkin parametre kümesini içerir. Bir uygulama, bağlantı yaratmak için `ConnectionFactory` kullanır. Bir uygulama, çalıştırma zamanında parametreleri sağlayabilir ve bir `ConnectionFactory` nesnesi oluşturabilir. Diğer bir seçenek olarak, bağlantı değiştirgeleri yönetilen nesnelerin bulunduğu bir havuzda saklanabilir. Bir uygulama havuzdan bir nesne alabilir ve havuzdan bir `ConnectionFactory` nesnesi yaratabilir.

Bağlantı
`Connection` nesnesi, bir uygulamadan ileti sistemi sunucusuna etkin bir bağlantıyı kapsüller. Bir uygulama, oturum yaratmak için bir bağlantı kullanır.

Hedef
Bir uygulama, `Destination` nesnesini kullanarak ileti gönderir ya da alır. Yayınlama/abone olma etki alanında, `Destination` nesnesi bir konuyu kapsüller ve noktadan noktaya iletişim etki alanında `Destination` nesnesi bir kuyruğu kaplar. Bir uygulama, yürütme sırasında `Destination` nesnesi yaratmak için gereken değiştirgeleri sağlayabilir. Diğer bir seçenek olarak, yönetilen nesneler havuzunda saklanan bir nesne tanımlamasından `Destination` nesnesi yaratabilir.

Oturum
`Session` nesnesi, ileti göndermek ve almak için tek iş parçacıklı bir bağlamdır. Bir uygulama, `Message`, `MessageProducer`ve `MessageConsumer` nesneleri oluşturmak için bir `Session` nesnesi kullanır.

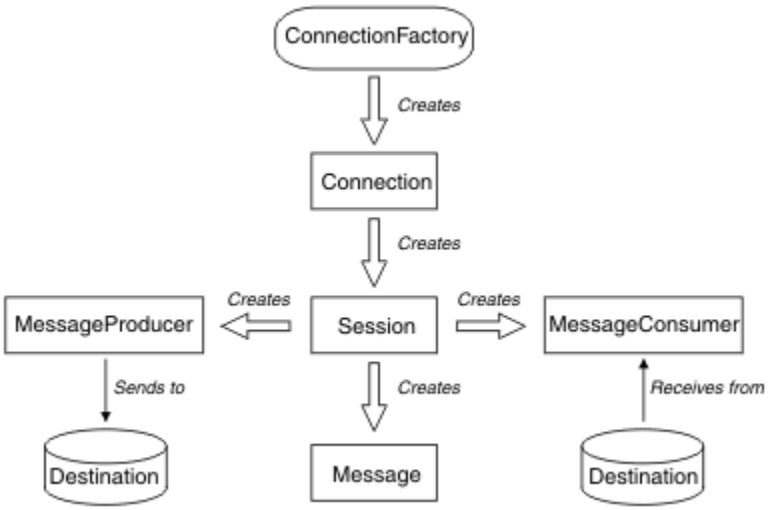
İleti
`Message` nesnesi, bir uygulamanın `MessageProducer` nesnesi kullanarak gönderdiği ya da `MessageConsumer` nesnesi kullanılarak aldığı `Message` nesnesini içerir.

MessageProducer
`MessageProducer` nesnesi, bir uygulama tarafından bir hedefe ileti göndermek için kullanılır.

MessageConsumer
`MessageConsumer` nesnesi, bir uygulama tarafından bir hedefe gönderilen iletileri almak için kullanılır.

XMS nesneleri ve ilişkileri

Şekil 52 sayfa 594 , XMS nesnesinin ana tiplerini gösterir: ConnectionFactory, Connection, Session, MessageProducer, MessageConsumer, Message ve Destination. Bir uygulama, bağlantı yaratmak için bir bağlantı üreticisi kullanır ve oturum yaratmak için bir bağlantı kullanır. Uygulama daha sonra ileti, ileti üreticileri ve ileti tüketicileri yaratmak için bir oturum kullanabilir. Uygulama, bir hedefe ileti göndermek için bir ileti üreticisi kullanır ve hedefe gönderilen iletileri almak için bir ileti tüketicisi kullanır.



Şekil 52. XMS nesneleri ve ilişkileri

XMS .NETiçinde, XMS sınıfları bir .NET arabirimleri kümesi olarak tanımlanır. XMS .NET uygulamalarını kodlarken, yalnızca bildirilmiş arabirimlere gereksinim duyarsınız.

XMS nesne modeli, Java Message Service Belirtim, Sürüm 1.1içinde açıklanan etki alanından bağımsız arabirimlere dayalıdır. `Topic`, `TopicPublisher`ve `TopicSubscriber`gibi etki alanına özgü sınıflar sağlanmaz.

XMS nesnelerinin öznitelikleri ve özellikleri

Bir XMS nesnesi, farklı şekillerde uygulanan, nesnenin özellikleri olan özniteliklere ve özelliklere sahip olabilir:

Öznitelikler
Özniteliğin bir değeri olmasa da, her zaman var olan ve depolamayı kaplayan bir nesne özelliği. Bu bakımdan, bir öznitelik, değişmez uzunluklu veri yapısındaki bir alana benzer. Özniteliklerin ayırt edici özelliği, her özniteliğin değerini ayarlamak ve almak için kendi yöntemlerine sahip olmaktır.

Özellikler
Bir nesnenin özelliği vardır ve yalnızca değeri ayarlandıktan sonra depolamayı kaplar. Bir özellik silinemez ya da değeri ayarlandıktan sonra depolama alanı kurtarılamaz. Değerini değiştirebilirsiniz. XMS , özellik değerlerini ayarlamak ve almak için bir soysal yöntemler kümesi sağlar.

Denetlenen nesneler

Yönetilen nesneleri kullanarak, merkezi bir havuzdan denetlenecek istemci uygulamaları tarafından kullanılan bağlantı ayarlarını denetleyebilirsiniz. Bir uygulama, nesne tanımlamalarını merkezi havuzdan alır ve bunları `ConnectionFactory` ve `Destination` nesnelerini yaratmak için kullanır. Yönetilen nesneleri kullanarak, uygulamaları yürütme sırasında kullandıkları kaynaklardan ayırabilirsiniz.

Örneğin, XMS uygulamaları, bir test ortamındaki bağlantı ve hedef kümesine başvuran yönetilen nesnelerle yazılabilir ve sınanabilir. Uygulamalar konuşlandırıldığında, uygulamaları üretim ortamındaki bağlantılara ve hedeflere başvuracak şekilde yapılandırmak için yönetilen nesneler değiştirilebilir.

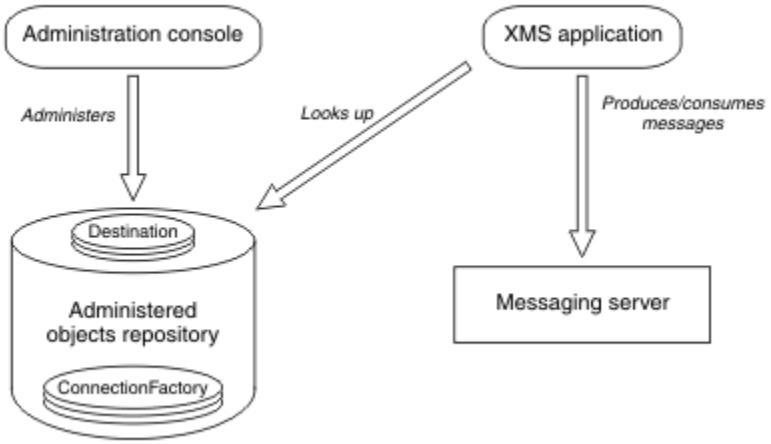
XMS , iki tip denetlenen nesneyi destekler:

- Sunucuyla ilk bağlantıyı kurmak için uygulamalar tarafından kullanılan bir `ConnectionFactory` nesnesi.
- Gönderilmekte olan iletilerin hedefini ve alınmakta olan iletilerin kaynağını belirtmek için uygulamalar tarafından kullanılan bir `Destination` nesnesi. Hedef, bir uygulamanın bağlandığı sunucudaki bir konu ya da kuyruktur.

JMSAdmin yönetim aracı IBM MQile birlikte sağlanır. Yönetilen nesnelerin merkezi bir havuzunda yönetilen nesneleri yaratmak ve yönetmek için kullanılır.

Havuzdaki yönetilen nesneler, IBM MQ classes for JMS ve XMS uygulamaları tarafından kullanılabilir. XMS uygulamaları bir IBM MQ kuyruk yöneticisinebağlanmak için `ConnectionFactory` ve `Destination` nesnelerini kullanabilir. Bir denetimci, uygulama kodunu etkilemeden havuzda tutulan nesne tanımlamalarını değiştirebilir.

Aşağıdaki çizgede, bir XMS uygulamasının genellikle yönetilen nesneleri nasıl kullandığı gösterilmektedir. Çizgenin sol tarafında, yönetim konsolu kullanılarak yönetilen ConnectionFactory ve Hedef nesne tanımlamalarını içeren bir havuz gösterilir. Çizgenin sağ tarafında, havuzdaki nesne tanımlamalarına bakan bir XMS uygulaması gösterilir ve daha sonra, bir ileti sistemi sunucusuna bağlanırken bu nesne tanımlamalarını kullanır.



Şekil 53. XMS uygulaması tarafından yönetilen nesnelerin tipik kullanımı

XMS ileti modeli

XMS ileti modeli, IBM MQ classes for JMS ileti modeliyle aynıdır.

Özellikle XMS , IBM MQ classes for JMS ' in uyguladığı aynı ileti üstbilgisi alanlarını ve ileti özelliklerini uygular:

- JMS üstbilgi alanları. Bu alanların JMS önekiyle başlayan adları vardır.
- JMS tanımlı özellikler. Bu alanların adları JMSX önekiyle başlayan özellikleri vardır.
- IBM tanımlı özellikler. Bu alanların adları JMS_IBM_ önekiyle başlayan özellikleri vardır.

Sonuç olarak, XMS uygulamaları IBM MQ classes for JMS uygulamalarıyla ileti değiş tokuşu yapabilir. Her iletide, bazı üstbilgi alanları ve özellikleri uygulama tarafından ayarlanır ve diğerleri XMS ya da IBM MQ classes for JMStarafından ayarlanır. XMS ya da IBM MQ classes for JMS tarafından ayarlanan alanlardan bazıları ileti gönderildiğinde, diğerleri alındığında ayarlanır. Üstbilgi alanları ve özellikleri, uygun olduğu durumlarda bir ileti sistemi sunucusu aracılığıyla iletiyle birlikte yayılır. Bu iletiler, iletiyi alan herhangi bir uygulamanın kullanımına sunulur.

İlgili kavramlar

IBM MQ classes for JMS

Linux Windows kurmaIBM MQ classes for XMS .NET

Örnekler de içinde olmak üzere IBM MQ classes for XMS .NET, Windows ve Linuxüzerinde IBM MQ ile birlikte kurulur.

IBM MQ 9.2.0' den Microsoft.NET Core 3.1 , IBM MQ classes for XMS .NET Standardürününü çalıştırmak için gerekli en düşük sürümdür.

V 9.3.0 V 9.3.0 IBM MQ 9.3.0' den IBM MQ , IBM MQ classes for XMS .NET Standardkullanan .NET 6 uygulamalarını destekler. Bir .NET Core 3.1 uygulaması kullanıyorsanız, bu uygulamayı `csproj` dosyasında küçük bir düzenleme ile çalıştırabilirsiniz; bu işlem `targetframeworkversion` değerini `"net6.0"`olarak ayarlayabilir ve yeniden derleme gerektirmez.

V 9.3.1 IBM MQ 9.3.1 , hedef çerçeve olarak .NET 6 için oluşturulmuş bir XMS .NET istemci kitaplığı sağlar. IBM MQ 9.3.1' den Microsoft .NET 6.0 , hedef çerçeve olarak .NET 6 kullanılarak oluşturulan IBM MQ kitaplıklarını kullanan uygulamaların çalıştırılması için gerekli en düşük sürümdür.

V 9.3.1 IBM MQ 9.3.1içinden, .NET Standard kullanılarak oluşturulan XMS .NET istemci kitaplığı, `netstandard2.0` yeni bir klasör altında bulunur ve hedef çerçeve olarak .NET 6 kullanılarak oluşturulan XMS .NET istemci kitaplığı, Windows üzerinde *MQ_INSTALLATION_PATH*`/bin` altında ve Linuxüzerinde *MQ_INSTALLATION_PATH*`/lib64` altında bulunur.

`amqmxmsstd.dll` kitaplık

V 9.3.1 IBM MQ 9.3.1' den `amqmxmsstd.dll` kitaplığını aşağıdaki yerlerde bulabilirsiniz:

Hedef çerçeve olarak .NET Standard 2.0 kullanılarak oluşturulan kitaplık

- Windows Windowssistemlerinde: *MQ_INSTALLATION_PATH*`\bin\netstandard2.0`.
- Linux Linuxsistemlerinde: *MQ_INSTALLATION_PATH*`\lib64\netstandard2.0`.

Deprecated Bu kitaplıklar kullanımdan kaldırılmıştır ve IBM ilerideki yayınlarda bu kitaplıkları kaldırmayı planlamaktadır.

Hedef çerçeve olarak .NET 6 kullanılarak oluşturulan kitaplık

- Windows Windowssistemlerinde: *MQ_INSTALLATION_PATH*`\bin`. Örnek uygulamalar *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/base`içine kurulur.
- Linux Linuxsistemlerinde: *MQ_INSTALLATION_PATH*`\lib64`. .NET örnekleri *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/base`içinde yer almaktadır.

LTS IBM MQ 9.3.0 Long Term Supportiçin, IBM MQ classes for XMS .NET Standard kitaplığı (`amqmxmsstd.dll`) aşağıdaki yerlerde bulunur:

- Windows Windowssistemlerinde: *MQ_INSTALLATION_PATH*`\bin`. Örnek uygulamalar *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/xms`içine kurulur.
- Linux Linuxsistemlerinde: *MQ_INSTALLATION_PATH*`/lib64 path`. .NET örnekleri *MQ_INSTALLATION_PATH*`/samp/dotnet/samples/cs/core/xms`içinde yer almaktadır.

Daha fazla bilgi için bkz “kurmaIBM MQ classes for .NET” sayfa 534.

Uyarı: V 9.3.1 Deprecated IBM MQ 9.3.1olanağından, hedef çerçeve olarak .NET Standard 2.0 kullanılarak oluşturulan IBM MQ .NET istemci kitaplıkları kullanımdan kaldırılmıştır ve bu kitaplıklara gönderme yapan uygulamalar derleme sırasında CS0618 uyarısı verilir.

Stabilized Tüm `IBM.XMS.*` kitaplıkları sağlanmaya devam eder, ancak bu kitaplıklar dengelenir; yani, bu kitaplıklara yeni özellikler eklenmez. En son özelliklerden herhangi biri için `amqmxmsstd.dll` kitaplığına geçmeniz gerekir. Ancak, IBM MQ 9.1 Long Term Support ya da Continuous Delivery yayın düzeylerinde var olan kitaplıkları kullanmaya devam edebilirsiniz.

V 9.3.1 Bir .NET Framework uygulaması, IBM MQ 9.3.1 sürümünden daha düşük bir sürümden `amqmdnetstd.dll` ya da `amqmxmsstd.dll` kullanılarak derlenirse ve aynı uygulama .NET 6 tabanlı IBM

MQ istemci kitaplıkları kullanılarak çalıştırılırsa, .NETtarafından şu `FileLoadException` tipi kural dışı durum yayınlanır:

```
Kural dışı durum saptandı: System.IO.FileLoadException: Dosya ya da yapıbirimi yüklenemedi
'amqmdnetstd, Version =x.x.x.x, Culture=nötr, PublicKeyToken=23d6cb914eeaac0e' ya da
Bağımlılıklarından biri. Bulunan yapıbiriminin bildirge tanımlaması,
düzenek referansı. (HRESULT kural dışı durumu: 0x80131040)

Dosya adı: ' amqmdnetstd, Sürüm =x.x.x.x, Culture=nötr,
PublicKeyToken=23d6cb914eeaac0e'
```

Bu hatayı çözmek için, *MQ_INSTALLATION_PATH*`/bin/netstandard2.0` içinde bulunan kitaplıkların .NET Framework uygulamasının çalıştığı dizine kopyalanması gerekir.

IBM MQ 9.2.0' den IBM MQ classes for XMS .NET Standard , NuGet havuzundan karşıdan yüklenebilir. NuGet paketi hem `amqmxmsstd.dll` kitaplığını hem de `amqmdnetstd.dll` kitaplığını içerir. `amqmxmsstd.dll` , `amqmdnetstd.dll` ' e bağlıdır ve XMS .NET Core uygulaması paketlenirken, hem `amqmxmsstd.dll` hem de `amqmdnetstd.dll` , XMS .NET Core uygulamasıyla birlikte paketlenmelidir. Daha fazla bilgi için bkz "IBM MQ classes for XMS .NET dosyasını NuGet havuzundan yükleme" sayfa 598.

`dspmqver` DELETE ...

.NET Core bileşenine ilişkin sürüm ve oluşturma bilgilerini görüntülemek için **`dspmqver`** komutunu kullanabilirsiniz.

IBM MQ classes for XMS .NET Framework ve IBM MQ classes for XMS .NET (.NET Standard ve .NET 6 kitaplıkları) arasında karşılaştırma

Aşağıdaki çizelgede, IBM MQ classes for XMS .NET (.NET Standard ve .NET 6 kitaplıkları) özellikleriyle karşılaştırıldığında IBM MQ classes for XMS .NET Framework özellikleri listelenmektedir.

Çizelge 80. IBM MQ classes for XMS .NET Framework ve IBM MQ classes for XMS .NET arasındaki farklar (.NET Standard ve .NET 6 kitaplıkları)

Özellik	**IBM MQ classes for XMS .NET Framework**	**IBM MQ classes for XMS .NET (.NET Standard ve .NET 6 kitaplıkları)**
Sınıf Adları (API ' ler)	Tüm sınıflar her ağda aynı kalır.	Tüm sınıflar her ağda aynı kalır.
İşletim Sistemi	Windows	Windows Dockerized konteynerleri Linux macOS
`app.config` dosyası (Yeniden dağıtılabilir istemcide İzlemeyi etkinleştirmek için yapılandırma dosyası)	`app.config` dosyası, yeniden dağıtılabilir pakete ilişkin izlemeyi etkinleştirmek için kullanılır.	`app.config` desteklenmiyor. Ortam değişkenlerini kullanın.
Takip edin	XMS .NET istemcisini izlemek için, izlemeyi etkinleştirmek için kullanılan **`XMS_TRACE_ON`** ortam değişkeni gibi var olan ortam değişkenlerini kullanabilirsiniz. Daha fazla bilgi için XMS ortam değişkenlerini kullanarak XMS .NET izlemesini yapılandırmabaşlıklı konuya bakın. Yeniden dağıtılabilen istemciler için, izlemeyi etkinleştirmek üzere `app.config` kütüğü kullanılabilir.	XMS .NET istemcisini izlemek için, izlemeyi etkinleştirmek için kullanılan **`XMS_TRACE_ON`** ortam değişkeni gibi var olan ortam değişkenlerini kullanabilirsiniz. Daha fazla bilgi için XMS ortam değişkenlerini kullanarak XMS .NET izlemesini yapılandırmabaşlıklı konuya bakın.

Çizelge 80. IBM MQ classes for XMS .NET Framework ve IBM MQ classes for XMS .NET arasındaki farklar (.NET Standard ve .NET 6 kitaplıkları) (devamı var)

Özellik	IBM MQ classes for XMS .NET Framework	IBM MQ classes for XMS .NET (.NET Standard ve .NET 6 kitaplıkları)
İletim Kipleri	Yönetilen, Yönetilmeyen ve Bağ Tanımları	Yönetilen
TLS	Windows anahtar deposu, sertifikaları saklamak için kullanılır.	Windowsüzerinde, sertifikaları saklamak için anahtar deposu kullanılmalıdır. İzin verilen değerler şunlardır: *USER ya da *SYSTEM. Girişe dayalı olarak, IBM MQ .NET istemcisi yürürlükteki kullanıcının Windows anahtar deposuna ya da sistem genişliğine bakar. Linuxüzerinde, sertifikaları kurmak için X509Store sınıfını kullanmanız ve .NET Core sertifikalarını aşağıdaki konuma kurmanız önerilir: "`.dotnet/corefx/cryptography/x509stores`".
CCDT	Desteklenenler:	Desteklenir ve CCDT yolunun ayarları .NET Framework sınıflarıyla aynıdır.
İstemci otomatik yeniden bağlanma	Desteklenenler:	Desteklenenler:
Dağıtılmış hareketler	Desteklenenler:	Desteklenmiyor
Dinamik bağlantılı kitaplıkların (dll) genel derleme önbelleğine (GAC) kurulması	Dll, IBM MQ kuruluşunun bir parçası olarak GAC ' ye kurulur.	Dll, IBM MQ kuruluşunun bir parçası olarak GAC ' ye kurulmaz.
WMQ, WPM ve RTT bağlantı tipleri için destek	WMQ, WPM ve RTT bağlantı tiplerini destekler	Yalnızca WMQ için destek
JNDI denetimli nesneler	LDAP ve FileSystem ' ı destekler	Yalnızca FileSystem ' i destekler

V 9.3.0 V 9.3.0 IBM MQ 9.3.0'den IBM MQ classes for XMS .NET Framework ' yi çalıştırmak için Microsoft.NET Framework V4.7.2 ya da sonraki bir sürümünü kurmanız gerekir.

İlgili görevler

"XMS örnek uygulamalarının kullanılması" sayfa 605

XMS .NET örnek uygulamaları, her API ' nin ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluşunuzu ve ileti sistemi sunucusu ayarlarınızı doğrulamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

Linux Windows IBM MQ classes for XMS .NET dosyasını NuGet havuzundan yükleme

IBM MQ classes for XMS .NET , .NET Developers tarafından kolayca kullanılabilmesi için NuGet havuzundan karşıdan yüklenebilir.

Bu görev hakkında

NuGet , .NETda dahil olmak üzere Microsoft geliştirme platformları için paket yöneticisidir. NuGet istemci araçları, paket üretme ve kullanma yeteneği sağlar. NuGet paketi, derlenmiş kod (DLL), bu kodla ilgili diğer dosyaları ve paketin sürüm numarası gibi bilgileri içeren açıklayıcı bir bildirgeyi içeren `.nupkg` uzantılı tek bir sıkıştırılmış dosyadır.

Hem `amqmdnetstd.dll` kitaplığını hem de `amqmxmsstd.dll` kitaplığını içeren `IBMXMSDotnetClient` NuGet paketini, tüm paket yazarları ve tüketiciler tarafından kullanılan merkezi paket havuzu olan NuGet Gallery 'den yükleyebilirsiniz.

Not: V 9.3.1 IBM MQ 9.3.1' den NuGet paketi, hedef çerçeve olarak .NET Standard 2.0 ve .NET 6 kullanılarak oluşturulan kitaplıkları içerir. .NET Standard 2.0 kitaplıkları, `netstandard2.0` klasörü altında bulunur ve .NET 6 kitaplıkları `net6.0` klasörü altında bulunur.

`IBMXMSDotnetClient` paketini karşıdan yüklemenin üç yolu vardır:

- Microsoft Visual Studiokomutunu kullanarak. NuGet , Microsoft Visual Studio uzantısı olarak dağıtılır. Microsoft Visual Studio 2012' dan NuGet varsayılan olarak önceden kurulur.
- NuGet Package Manager ya da .NET CLI kullanılarak komut satırından.
- Bir web tarayıcısı kullanarak.

Yeniden dağıtılabilir pakete gelince, **XMS_TRACE_ON**ortam değişkenini kullanarak izlemeyi etkinleştirebilirsiniz.

Yordam

- Microsoft Visual Studioiçindeki Package Manager UI 'sini kullanarak `IBMXMSDotnetClient` paketini karşıdan yüklemek için aşağıdaki adımları tamamlayın:
  a) .NET projesini farenin sağ düğmesiyle tıklatın ve **Nuget Packages**öğesini seçin.
  b) **Göz At** etiketini tıklatın ve `"IBMXMSDotnetClient"`için arama yapın.
  c) Paketi seçin ve **Kur**düğmesini tıklatın.
  Kuruluş sırasında Package Manager, konsol deyimleri biçiminde aşama bilgileri sağlar.
- `IBMXMSDotnetClient` paketini komut satırından yüklemek için aşağıdaki seçeneklerden birini belirleyin:
  - NuGet Package Manager 'ı kullanarak aşağıdaki komutu girin:

    ```
    Install-Package IBMXMSDotnetClient -Version 9.1.4.0
    ```

    Kuruluş sırasında Package Manager, konsol deyimleri biçiminde aşama bilgileri sağlar. Çıkışı bir günlük dosyasına yeniden yönlendirebilirsiniz.
  - .NET CLI ' yı kullanarak şu komutu girin:

    ```
    dotnet add package IBMXMSDotnetClient --version 9.1.4
    ```
- Bir web tarayıcısı kullanarak, `IBMXMSDotnetClient` paketini https://www.nuget.org/packages/IBMXMSDotnetClientadresinden yükleyin.

İlgili kavramlar

“kurmaIBM MQ classes for .NET” sayfa 534
Örnekler de içinde olmak üzere IBM MQ classes for .NET, Windows ve Linux üzerinde IBM MQ ile birlikte kurulur.

IBM MQ Client for .NET lisans bilgileri

İlgili görevler

“NuGet havuzundan IBM MQ classes for .NET ürününü karşıdan yükleme” sayfa 539
IBM MQ classes for .NET , .NET Developers tarafından kolayca kullanılabilmesi için NuGet havuzundan karşıdan yüklenebilir.

Messaging Server ortamını ayarlama

Bu bölümdeki konular, XMS uygulamalarının bir sunucuya bağlanmasına izin vermek için ileti alışverişi sunucusu ortamının nasıl ayarlanacağını açıklar.

Bu görev hakkında

IBM MQ kuyruk yöneticisine bağlanan uygulamalar için IBM MQ istemcisi (ya da bağ tanımlama kipi için kuyruk yöneticisi) gereklidir.

Şu anda bir aracıya gerçek zamanlı bağlantı kullanan uygulamalar için önkoşul yok.

XMSile verilen örnek uygulamalar da içinde olmak üzere, herhangi bir XMS uygulamasını çalıştırmadan önce ileti sistemi sunucusu ortamını ayarlamanız gerekir.

Bu bölüm aşağıdaki konuları içerir:

- "IBM MQ kuyruk yöneticisine bağlanan bir uygulama için kuyruk yöneticisinin ve aracının yapılandırılması" sayfa 602
- "kurmaIBM MQ classes for XMS .NET" sayfa 595
- "Aracıya gerçek zamanlı bağlantı kullanan bir uygulama için aracı yapılandırılması" sayfa 603
- "WebSphere Application Server ile bağlantı kuran bir uygulama için hizmet bütünleştirme veriyolunu yapılandırma" sayfa 604

XMS .NET içindeki ileti dinleyicileri

İletileri zamanuyumsuz olarak almak için ileti dinleyici kullanılır. `MessageConsumer.receive()` çağrıdan farklı olarak, ileti dinleyici çağıran iş parçacığını engellemez; bunun yerine, iletileri genellikle `onMessage` yöntemi olan uygulama tarafından belirtilen bir geri çağırma yöntemine gönderir.

`Connection.Start()` yöntemi çağrıldığında ileti teslimi başlar. İleti teslimi, sırasıyla `Connection.Stop()` ve `Connection.Start()` yöntemleri kullanılarak her zaman durdurulabilir ve sürdürülebilir.

Bir oturumda ileti dinleyicisini en az bir tüketiciye ayarladıktan sonra `Connection.Start()` yöntemi çağrıldığında, o oturum zamanuyumsuz bir oturum olur. Bir oturum zamanuyumsuz olduğunda, herhangi bir XMS .NET zamanuyumlu yöntemi çağrılamaz. örneğin, `MessageProducer.Send()`. Bu, IBM MQ neden kodu MQRC_HCONN_ASYNC_ACTIVE (2500) ile bir kural dışı durumla sonuçlanır.

Zamanuyumsuz bir oturumda zamanuyumlu çağrılar

`Session.Close` , zamanuyumsuz bir oturumda izin verilen tek zamanuyumlu çağrıdır. Uygulamalar, ileti dinleyici geri çağırma yöntemini (yani `onMessage` yöntemini) kullanarak zamanuyumlu çağrılar da yapabilir (`Session.Close`dışında).

Bu iki seçeneğin dışında, bir uygulamanın zamanuyumlu çağrı yapması için `Connection.Stop()` yöntemini kullanarak bağlantıyı durdurmanız gerekir. Çağrılar yapıldıktan sonra, `Connection.Start()` yöntemini kullanarak bağlantıyı yeniden sürdürmeniz gerekir. ileti teslimini yeniden başlatır.

Bir oturumda kaç zamanuyumsuz ileti tüketicisi olabilir?

Bir oturumun birden çok zamanuyumsuz ileti tüketicisi olabilir. Ancak herhangi bir zamanda bir mesaj sadece bir tüketiciye gönderilir. Bunun pratik anlamı, XMS .NET ilk iletiyi teslim etmek için bir tüketicinin `onMessage()` yöntemini çağırırken ikinci bir ileti geldiğinde, ikinci ileti `onMessage()` yöntemi geri dönünceye kadar oturumda bir tüketiciye teslim edilmez.

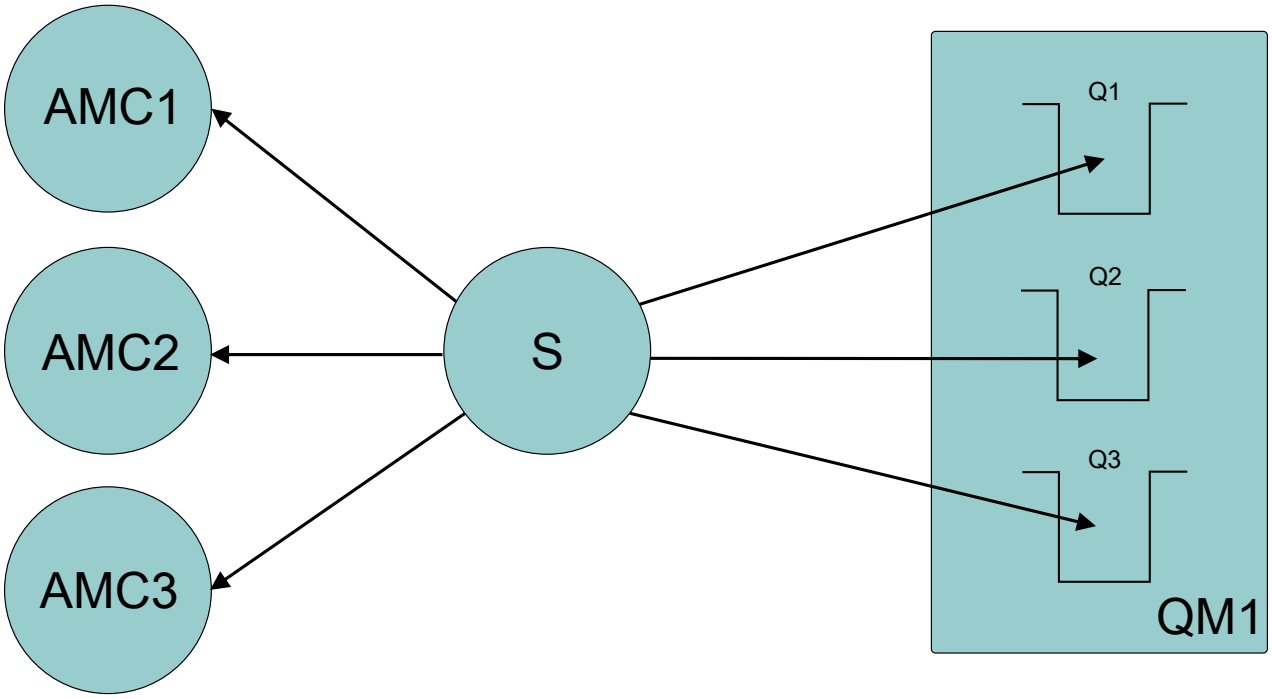
İkinci ileti, yalnızca `onMessage()` yöntemi geri döndükten sonra oturumdaki bir tüketiciye teslim edilir. Bunun nedeni, bir oturumun yalnızca tek bir iş parçacığı kullanarak tüketicilere ileti teslimini yönetmesi olabilir. Bu, aynı anda yalnızca bir iletinin teslim edilebildiği ve tüketicinin herhangi bir ileti olabileceği anlamına gelir.

Bir uygulama eşzamanlı ileti teslimi gerektiriyorsa, yani tüm tüketicilerin aynı anda ileti alması gerekiyorsa, uygulamanın birden çok oturum oluşturması ve her birinin bir zamanuyumsuz ileti tüketicisine sahip olması gerekir.

Aşağıdaki örnekler bu özelliği daha net bir şekilde göstermektedir.

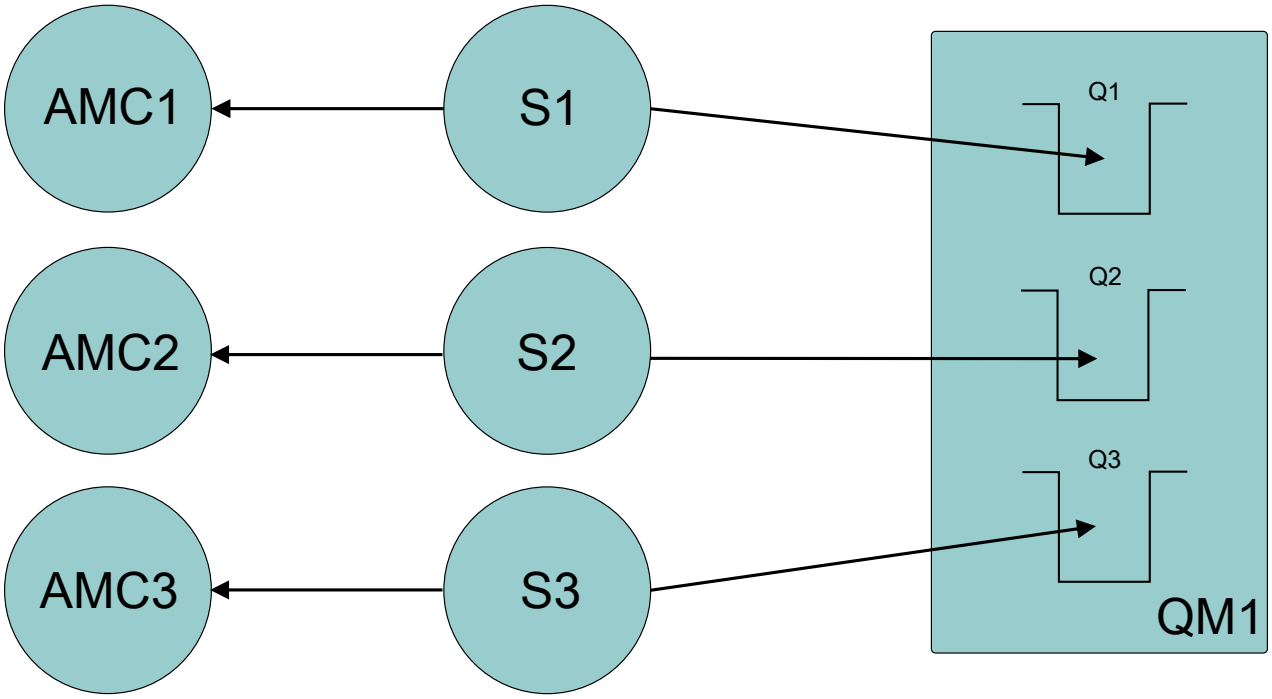
İlk örnekte, bir oturumda birden çok zamanuyumsuz ileti tüketicisi vardır. Oturum S ' nin üç zamanuyumsuz ileti tüketicisi vardır: AMC1, AMC2 ve AMC3 . Bu kullanıcılar üç farklı hedef Q1, Q2 ve Q3.

Yalnızca bir oturum Solduğundan, iletileri tüketicilere AMC1, AMC2ve AMC3teslim etmek için yalnızca ileti teslim iş parçacığı vardır. Oturum AMC1'e ileti gönderirken, Q2 ve Q3 ' de teslim edilmeye hazır iletiler olsa bile, diğer iki tüketici AMC2 ve AMC3 bekleyin.



Şekil 54. Üç zamanuyumsuz ileti tüketicisiyle bir oturum

İkinci durumda, sırasıyla bir zamanuyumsuz ileti tüketicisine AMC1, AMC2ve AMC3 sahip birden çok oturum S1, S2ve S3vardır. Her oturum için bir tüketici olduğundan, iletiler eşzamanlı olarak tüketicilere teslim edilir.



Şekil 55. Her biri bir zamanuyumsuz ileti tüketicisine sahip birden çok oturum

Bu, eşzamanlı ileti tesliminize gereksinim duyarsanız, birden çok oturuma ihtiyacınız olduğunu gösterir.

IBM MQ kuyruk yöneticisine bağlanan bir uygulama için kuyruk yöneticisinin ve aracının yapılandırılması

Bu bölümde, IBM WebSphere MQ 7.0.1ya da daha sonraki bir sürümü kullandığınız varsayılır. IBM MQ kuyruk yöneticisine bağlanan bir uygulamayı çalıştırmadan önce kuyruk yöneticisini yapılandırmanız gerekir. Yayınlama/abone olma uygulaması için, Kuyruğa yollanmış yayınlama/abone olma arabirimini kullanıyorsanız, ek yapılandırma gereklidir.

Başlamadan önce

XMS , IBM Integration Bus ya da WebSphere Message Broker 6.1 ya da daha sonraki bir yayın düzeyiyle çalışır

Bu görevi başlatmadan önce aşağıdaki adımları gerçekleştirin:

- Uygulamanızın çalışmakta olan bir kuyruk yöneticisine erişimi olduğunu doğrulayın.
- Uygulamanız bir yayınlama/abone olma uygulamasıysa ve Kuyruğa yollanmış yayınlama/abone olma arabirimini kullanıyorsa, kuyruk yöneticisinde **PSMODE** özniteliğinin ENABLED olarak ayarlandığından emin olun.
- Uygulamanızın, kuyruk yöneticisine bağlanmak için özellikleri uygun şekilde ayarlanmış bir bağlantı üreticisi kullandığından emin olun. Uygulamanız bir yayınlama/abone olma uygulamasıysa, aracıyı kullanmak için uygun bağlantı üreticisi özelliklerinin ayarlandığından emin olun. Bir bağlantı üreticisinin özellikleri hakkında daha fazla bilgi için bkz. ConnectionFactoryözellikleri.

Bu görev hakkında

Kuyruk yöneticisini ve aracıyı, XMS uygulamalarını, IBM MQ JMS uygulamalarını çalıştırmak üzere kuyruk yöneticisini ve kuyruğa alınan yayınlama/abone olma arabirimini yapılandırdığınız şekilde çalıştıracak şekilde yapılandırırsınız. Aşağıdaki adımlar, yapmanız gerekenleri özetler.

Yordam

1. Kuyruk yöneticisinde, uygulamanızın gereksinim duyduğu kuyrukları yaratın.

   Kuyrukları nasıl yaratmaya ilişkin genel bakış için Kuyrukları tanımlamabaşlıklı konuya bakın.

   Uygulamanız bir yayınlama/abone olma uygulamasıysa ve IBM MQ classes for JMS sistem kuyruklarına erişmesi gereken Kuyruklanmış yayınlama/abone olma arabirimini kullanıyorsa, kuyrukları oluşturmadan önce Adım 4a ' yı bekleyin.
2. Uygulamanızla ilişkili kullanıcı kimliğine, kuyruk yöneticisine bağlanma yetkisi ve kuyruklara erişmek için gereken yetkiyi verin.

   Yetkilendirmeye ilişkin genel bilgiler için bkz. Güvenlik. Uygulamanız istemci kipinde kuyruk yöneticisine bağlanıyorsa, İstemciler ve sunucularkonusuna bakın.
3. Uygulamanız istemci kipinde kuyruk yöneticisine bağlanıyorsa, kuyruk yöneticisinde bir sunucu bağlantısı kanalının tanımlandığından ve bir dinleyicinin başlatıldığından emin olun.

   Kuyruk yöneticisine bağlanan her uygulama için bu adımı gerçekleştirmeniz gerekmez. Bir sunucu bağlantısı kanal tanımlaması ve bir dinleyici, istemci kipinde bağlanan tüm uygulamaları destekleyebilir.
4. Uygulamanız bir yayınlama/abone olma uygulamasıysa ve Queued yayınlama/abone olma arabirimini kullanıyorsa, aşağıdaki adımları gerçekleştirin.
   a) Kuyruk yöneticisinde, IBM MQile verilen MQSC komutlarının komut dosyasını çalıştırarak IBM MQ classes for JMS sistem kuyruklarını yaratın. IBM Integration Bus ya da WebSphere Message Broker ile ilişkili kullanıcı kimliğinin kuyruklara erişme yetkisine sahip olduğundan emin olun.

      Komut dosyasının nerede bulunacağı ve nasıl çalıştırılacağı hakkında bilgi için bkz. IBM MQ classes for Java ' yi kullanma.

      Bu adımı kuyruk yöneticisi için yalnızca bir kez gerçekleştirin. Aynı IBM MQ classes for JMS sistem kuyrukları kümesi, kuyruk yöneticisine bağlanan tüm XMS ve IBM MQ classes for JMS uygulamalarını destekleyebilir.

   b) Uygulamanızla ilişkili kullanıcı kimliğine IBM MQ classes for JMS sistem kuyruklarına erişim yetkisi verin.

      Kullanıcı kimliğinin hangi yetkilere ihtiyacı olduğu hakkında bilgi için bkz. IBM MQ classes for JMS ' yi kullanma.

   c) Bir IBM Integration Bus ya da WebSphere Message Brokeraracısı için, uygulamaların yayınladıkları iletileri gönderdikleri kuyruğa hizmet etmek üzere bir ileti akışı yaratın ve konuşlandırın.

      Temel ileti akışı, yayınlanan iletileri okumak için bir MQInput ileti işleme düğümünden ve iletileri yayınlamak için bir Yayın ileti işleme düğümünden oluşur.

      Bir ileti akışının nasıl yaratılacağına ve konuşlandırılacağına ilişkin bilgi için IBM Integration Bus ürün belgeleri kitaplığı web sayfasıiçinde bulunan IBM Integration Bus ya da WebSphere Message Broker ürün belgelerine bakın.

      Aracıda konuşlandırılmış uygun bir ileti akışı varsa, bu adımı gerçekleştirmeniz gerekmez.

Sonuçlar

Artık uygulamanızı başlatabilirsiniz.

Aracıya gerçek zamanlı bağlantı kullanan bir uygulama için aracı yapılandırılması

Bir aracıya gerçek zamanlı bağlantı kullanan bir uygulamayı çalıştırmadan önce, o aracıyı yapılandırmanız gerekir.

Başlamadan önce

Bu görevi başlatmadan önce aşağıdaki adımları izleyin:

- Uygulamanızın çalışmakta olan bir aracıya erişimi olduğunu doğrulayın.
- Uygulamanızın, bir aracıya gerçek zamanlı bağlantı için özellikleri uygun şekilde ayarlanmış bir bağlantı üreticisi kullandığından emin olun. Bir bağlantı üreticisinin özellikleri hakkında daha fazla bilgi için bkz. ConnectionFactoryözellikleri.

Bu görev hakkında

Bir aracıyı, XMS uygulamalarını IBM MQ classes for JMS uygulamalarını çalıştıracak şekilde yapılandırdığınız şekilde çalıştıracak şekilde yapılandırırsınız. Aşağıdaki adımlar, yapmanız gerekenleri özetler:

Yordam

1. Bir aracının dinlediği TCP/IP kapısından iletileri okumak ve yayınlamak için bir ileti akışı yaratın ve konuşlandırın.

   Bunu aşağıdaki yollardan biriyle yapabilirsiniz:

   - **Real-timeOptimizedFlow** ileti işleme düğümü içeren bir ileti akışı yaratın.
   - **Real-timeInput** ileti işleme düğümü ve bir Yayın ileti işleme düğümü içeren bir ileti akışı yaratın.

   Real-timeOptimizedFlow ya da **Real-timeInput** düğümünü, gerçek zamanlı bağlantılar için kullanılan kapıyı dinleyecek şekilde yapılandırmanız gerekir. XMSiçinde, gerçek zamanlı bağlantılar için varsayılan kapı numarası 1506 'dır.

   Aracıda konuşlandırılmış uygun bir ileti akışı varsa, bu adımı gerçekleştirmeniz gerekmez.
2. İletilerin IBM MQ classes for JMSkullanarak uygulamanıza teslim edilmesini istiyorsanız, aracıyı çoklu yayını etkinleştirecek şekilde yapılandırın. Güvenilir çoklu yayın gerektiren konular için güvenilir bir hizmet kalitesi belirterek, çok hedefli olması gereken konuları yapılandırın.
3. Uygulamanız aracıya bağlandığında bir kullanıcı kimliği ve parola sağladıysa ve aracının bu bilgileri kullanarak uygulamanızı doğrulamasını istiyorsanız, kullanıcı adı sunucusunu ve aracıyı basit Telnet benzeri parola kimlik doğrulaması için yapılandırın.

Sonuçlar

Artık uygulamanızı başlatabilirsiniz.

WebSphere Application Server ile bağlantı kuran bir uygulama için hizmet bütünleştirme veriyolunu yapılandırma

WebSphere Application Server service integration technologies hizmet bütünleştirmesi veriyoluna bağlanan bir uygulamayı çalıştırabilmek için, hizmet bütünleştirmesini varsayılan ileti alışverişi sağlayıcısını kullanan JMS uygulamalarını çalıştıracak şekilde yapılandırmanız gerekir.

Başlamadan önce

Bu görevi başlatmadan önce aşağıdaki adımları gerçekleştirmeniz gerekir:

- Bir ileti sistemi veriyolu yaratıldığını ve sunucunuzun veriyoluna veriyolu üyesi olarak eklendiğini doğrulayın.
- Uygulamanızın, çalışmakta olan en az bir ileti alışverişi altyapısı içeren bir hizmet tümleştirme veriyoluna erişimi olduğunu doğrulayın.
- HTTP işlemi gerekiyorsa, bir HTTP ileti alışverişi altyapısı gelen aktarım kanalı tanımlanmalıdır. Varsayılan olarak, sunucu kuruluşu sırasında SSL ve TCP kanalları tanımlanır.
- Uygulamanızın, özellikleri bir önyükleme sunucusu kullanarak hizmet bütünleştirme veriyoluna bağlanmak için uygun şekilde ayarlanmış bir bağlantı üreticisi kullandığından emin olun. Gereken en düşük bilgi:

– İleti sistemi sunucusuyla bağlantı kurarken (önyükleme sunucusu aracılığıyla) kullanılacak konumu ve protokolü tanımlayan sağlayıcı uç noktası. En basit biçimiyle, varsayılan ayarlarla kurulan bir sunucu için, sağlama uç noktası, sunucunun anasistem adına ayarlanabilir.
– İletilerin gönderildiği veriyolunun adı.

Bir bağlantı üreticisinin özellikleri hakkında daha fazla bilgi için bkz. ConnectionFactoryözellikleri.

Bu görev hakkında

Gerek duyduğunuz kuyruk ya da konu alanları tanımlanmalıdır. Varsayılan olarak, sunucu kuruluşu sırasında Default.Topic.Space adlı bir konu alanı tanımlanır, ancak ek konu alanlarına gereksinim duyarsanız, bu konu alanlarını kendiniz yaratmanız gerekir. Sunucu bu konuları gerektiğinde dinamik olarak somutlaştırdığı için, konu alanında tek tek konuları önceden tanımlamanıza gerek yoktur.

Aşağıdaki adımlar, yapmanız gerekenleri özetler.

Yordam

1. Uygulamanızın noktadan noktaya ileti alışverişi için gereksinim duyduğu kuyrukları yaratın.
2. Uygulamanızın yayınlama/abone olma ileti sistemi için gereksinim duyduğu ek konu alanlarını oluşturun.

Sonuçlar

Artık uygulamanızı başlatabilirsiniz.

XMS örnek uygulamalarının kullanılması

XMS .NET örnek uygulamaları, her API ' nin ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluşunuzu ve ileti sistemi sunucusu ayarlarınızı doğrulamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

Bu görev hakkında

Kendi uygulamalarınızı yaratmak için yardıma gereksinim duyarsanız, örnek uygulamaları başlangıç noktası olarak kullanabilirsiniz. Her uygulama için hem kaynak hem de derlenmiş bir sürüm sağlanır. Örnek kaynak kodunu gözden geçirin ve uygulamanız için gerekli her bir nesneyi (ConnectionFactory, Connection, Session, Destination, and a Producer, and a Consumer ya da her ikisi) yaratmak ve uygulamanızın nasıl çalışmasını istediğinizi belirtmek için gereken belirli özellikleri ayarlamak için gereken anahtar adımlarını tanımlayın. Daha fazla bilgi için bkz “XMS uygulamaları yazılıyor” sayfa 608. Örnekler, XMS' in gelecekteki yayınlarında değiştirilebilir.

Aşağıdaki tabloda, XMSile birlikte sağlanan örnek uygulama kümeleri (her API için bir tane) gösterilmektedir.

Çizelge 81. XMS .NET için örnek uygulamalar

Örneğin adı	**Açıklama**
SampleConsumerCS	Bir kuyruktan ileti alan ya da bir konuya abone olan ileti tüketicisi uygulaması.
SampleProducerCS	Bir kuyruğa ya da konuya ileti üreten bir ileti üreticisi uygulaması.
SampleConfigCS	Dosya tabanlı bir yönetilen nesne havuzu yaratmak için kullanabileceğiniz bir yapılandırma uygulaması. Uygulama, belirli bağlantı ayarlarınız için bir bağlantı üreticisi ve hedef içerir. Bu yönetilen nesne havuzu, örnek tüketici ve üretici uygulamalarının her biriyle birlikte kullanılabilir.

Çeşitli API ' lerde aynı işlevleri destekleyen örneklerin sözdizimi farklılıkları vardır.

- Örnek ileti tüketici ve üretici uygulamalarının her ikisi de aşağıdaki işlevleri destekler:
  - IBM MQ, IBM Integration Bus (bir aracıya gerçek zamanlı bağlantı kullanarak) ve WebSphere Application Server service integration bus bağlantıları
  - İlk bağlam arabirimini kullanarak yönetilen nesne havuzu aramaları
  - Kuyruklara (IBM MQ ve WebSphere Application Server service integration bus) ve konulara bağlantılar (IBM MQ, bir aracıya gerçek zamanlı bağlantı ve WebSphere Application Server service integration bus)
  - Temel, bayt, eşlem, nesne, akış ve metin iletileri
- Örnek ileti tüketici uygulaması, zamanuyumlu ve zamanuyumsuz alma kiplerini ve SQL Seçici deyimlerini destekler.
- Örnek ileti üreticisi uygulaması, kalıcı ve kalıcı olmayan teslim kiplerini destekler.

Örnekler iki kipten birinde çalışabilir:

Basit kip
Örnekleri en düşük kullanıcı girişiyle çalıştırabilirsiniz.

Gelişmiş kip
Örneklerin çalışma şeklini daha iyi özelleştirebilirsiniz.

Tüm örnekler uyumludur ve bu nedenle diller arasında çalışabilir.

Windows IBM MQ 9.1.1içinden IBM MQ , Windows ortamlarındaki .NET Core for XMS .NET uygulamalarını destekler. IBM MQ classes for .NET Standart, örnekler de içinde olmak üzere, standart IBM MQ kuruluşunun bir parçası olarak varsayılan olarak kurulur.

Linux IBM MQ 9.1.2' den IBM MQ , Linux ortamlarındaki uygulamalar için .NET Core 'u da destekler.

XMS .NET için örnek uygulamalar `&MQINSTALL_PATH&/samp/dotnet/samples/cs/core/xms`içine kurulur.

Daha fazla bilgi için bkz “kurmaIBM MQ classes for XMS .NET” sayfa 595.

.NET örnek uygulamalarının çalıştırılması

.NET örnek uygulamalarını, otomatik oluşturulan ya da özelleştirilmiş yanıt dosyalarını kullanarak etkileşimli olarak ya da etkileşimli olmayan bir biçimde çalıştırabilirsiniz.

Başlamadan önce

Sağlanan örnek uygulamalardan herhangi birini çalıştırmadan önce, uygulamaların bir sunucuya bağlanabilmesi için önce ileti alışverişi sunucusu ortamını ayarlamanız gerekir. Bkz. “Messaging Server ortamını ayarlama” sayfa 600.

Yordam

.NET örnek uygulamasını çalıştırmak için aşağıdaki adımları izleyin:

İpucu: Örnek bir uygulama çalıştırırken, yazsın mı? Daha sonra ne yapacaktır konusunda yardım almak için herhangi bir zamanda.

1. Örnek uygulamayı çalıştırmak istediğiniz kipi seçin.
   `Advanced` ya da `Simple`yazın.
2. Sorulara cevap ver.
   Sorunun sonundaki köşeli parantez içinde gösterilen varsayılan değeri seçmek için Enter tuşuna basın. Farklı bir değer seçmek için uygun değeri yazın ve Enter tuşuna basın.

Örnek bir soru:

```
Enter connection type [wpm]:
```

Bu durumda, varsayılan değer `wpm` (bir WebSphere Application Server service integration busbağlantısı) değeridir.

Sonuçlar

Örnek uygulamaları çalıştırdığınızda, yanıt dosyaları yürürlükteki çalışma dizininde otomatik olarak oluşturulur. Yanıt dosyası adları *`connection_type-sample_type`*`.rsp`biçimindedir; örneğin, `wpm-producer.rsp`. Gerekirse, örnek uygulamayı aynı seçeneklerle yeniden çalıştırmak için oluşturulan yanıt dosyasını kullanabilirsiniz; böylece, seçenekleri yeniden girmenize gerek yoktur.

İlgili görevler

.NET örnek uygulamalarının oluşturulması
Örnek bir .NET uygulaması oluşturduğunuzda, seçtiğiniz örneğin yürütülebilir bir sürümü yaratılır.

Kendi uygulamalarınızı oluşturma
Örnek uygulamaları oluşturmanız gibi kendi uygulamalarınızı oluşturun.

.NET örnek uygulamalarının oluşturulması

Örnek bir .NET uygulaması oluşturduğunuzda, seçtiğiniz örneğin yürütülebilir bir sürümü yaratılır.

Başlamadan önce

Uygun derleyicinin kurulması. Bu görev, Microsoft Visual Studio 2012 ' in kurulu olduğunu ve bunu kullanmaya alışık olduğunuzu varsayar.

Yordam

Bir .NET örnek uygulaması oluşturmak için aşağıdaki adımları tamamlayın:

1. .NET örnekleriyle birlikte sağlanan `Samples.sln` çözüm dosyasını tıklatın.
2. Çözüm Gezgini penceresinde `Örnekler` çözümünü sağ tıklatın ve **Çözüm Oluştur**seçeneğini belirleyin.

Sonuçlar

Seçtiğiniz yapılandırmaya bağlı olarak, örneğin uygun alt klasöründe (`bin/Debug` ya da `bin/Release`) yürütülebilir bir program yaratılır. Bu program, CSsonekine sahip klasörle aynı ada sahip. Örneğin, ileti üreticisi örnek uygulamasının C# sürümünü oluşturyorsanız, `SampleProducerCS.exe` `SampleProducer` klasöründe oluşturulur.

İlgili görevler

.NET örnek uygulamalarının çalıştırılması
.NET örnek uygulamalarını, otomatik oluşturulan ya da özelleştirilmiş yanıt dosyalarını kullanarak etkileşimli olarak ya da etkileşimli olmayan bir biçimde çalıştırabilirsiniz.

Kendi uygulamalarınızı oluşturma
Örnek uygulamaları oluşturmanız gibi kendi uygulamalarınızı oluşturun.

“Kendi uygulamalarınızı oluşturma” sayfa 607
Örnek uygulamaları oluşturmanız gibi kendi uygulamalarınızı oluşturun.

Kendi uygulamalarınızı oluşturma

Örnek uygulamaları oluşturmanız gibi kendi uygulamalarınızı oluşturun.

Başlamadan önce

Uygun derleyicinin kurulması. Bu görev, Microsoft Visual Studio 2012 ' in kurulu olduğunu ve bunu kullanmaya alışık olduğunuzu varsayar.

Yordam

- .NET uygulamanızı “.NET örnek uygulamalarının oluşturulması” sayfa 607başlıklı konuda açıklandığı gibi oluşturun.

  Kendi uygulamalarınızı nasıl oluşturacağınıza ilişkin ek kılavuzluk için, her örnek uygulama için sağlanan makefile dosyalarını kullanın.

  İpucu: Bir hata durumunda sorun tanılamasında yardımcı olmak için, uygulamaları içinde yer alan simgelerle derlemek yararlı olabilir.

İlgili görevler

.NET örnek uygulamalarının çalıştırılması
.NET örnek uygulamalarını, otomatik oluşturulan ya da özelleştirilmiş yanıt dosyalarını kullanarak etkileşimli olarak ya da etkileşimli olmayan bir biçimde çalıştırabilirsiniz.

.NET örnek uygulamalarının oluşturulması
Örnek bir .NET uygulaması oluşturduğunuzda, seçtiğiniz örneğin yürütülebilir bir sürümü yaratılır.

XMS uygulamaları yazılıyor

Bu bölümdeki konular, genel olarak XMS uygulamaları yazarken size yardımcı olacak bilgileri sağlar.

Bu görev hakkında

Bu bölüm, XMS uygulamalarının yazılmasına ilişkin genel kavramları içerir. XMS .NET uygulamaları oluşturmaya özgü bilgiler için ayrıca bkz. “XMS .NET uygulamaları yazılıyor” sayfa 625 .

IBM MQ 9.2.0'den XMS.NET dinamik bağlantı kitaplıkları önemli ölçüde azaldı, toplam 5 'e kadar. Beş dinamik bağlantı kitaplığı şunlardır:

- `IBM.XMS.dll` -tüm ulusal dil iletilerini içerir
- `IBM.XMS.Comms.RMM.dll`
- Üç ilke dinamik bağlantı kitaplığı:
  - `policy.8.0.IBM.XMS.dll`
  - `policy.9.0.IBM.XMS.dll`
  - `policy.9.1.IBM.XMS.dll`

Removed V 9.3.0 V 9.3.0 XMS .NET Çok hedefli ileti sistemi (RMMkullanılarak) IBM MQ 9.2 ' den kullanımdan kaldırılmıştır ve IBM MQ 9.3adresinden kaldırılmıştır.

Bu bölüm aşağıdaki konuları içerir:

- “İş threading modeli” sayfa 610
- “ConnectionFactories ve Connection nesneleri” sayfa 610
- “Oturumlar” sayfa 611
- “Hedefler” sayfa 614
- “İleti üreticileri” sayfa 617
- “İleti tüketicileri” sayfa 617
- “Kuyruk tarayıcıları” sayfa 620
- “İstekte Bulunanlar” sayfa 621
- “Nesne silme” sayfa 621
- “XMS ilkel tipleri” sayfa 622
- “Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi” sayfa 622
- “Yineleyiciler” sayfa 624
- “Kodlanmış karakter takımı tanıtıcıları” sayfa 625

- “XMS hata ve kural dışı durum kodları” sayfa 625
- “Kendi uygulamalarınızı oluşturma” sayfa 607

Windows IBM MQ XMS .NET proje şablonunun kullanılması

IBM MQ XMS .NET istemcisi, XMS .NET Core uygulamalarınızı geliştirmenize yardımcı olacak bir proje şablonu kullanma yeteneği sunar.

Başlamadan önce

Sisteminizde Microsoft Visual Studio 2017ya da üstü ve .NET Core 2.1 olmalıdır.

XMS .NET şablonunu

&MQ_INSTALL_ROOT&\tools\dotnet\samples\cs\core\xms\ProjectTemplates\IBMXMS.NETClientApp.zip

dizin

&USER_HOME_DIRECTORY&\Documents\&Visual_Studio_Version&\Templates\ProjectTemplates

dizin, burada:

- *& MQ_INSTALL_ROOT* kuruluşunuzun kök dizinidir
- *& USER_HOME_DIRECTORY* sizin ana dizininizdir.

Şablonu almak için Microsoft Visual Studio ' i durdurup yeniden başlatmanız gerekir.

Bu görev hakkında

XMS .NET proje şablonu, uygulamalarınızı geliştirmenize yardımcı olmak için kullanabileceğiniz bazı ortak kodları içerir. Yerleşik kodla IBM MQ kuyruk yöneticisine bağlanabilir ve yerleşik koddaki özellikleri değiştirerek bir koyma ya da alma işlemi gerçekleştirebilirsiniz.

Yordam

1. Microsoft Visual Studio uygulamasını açın.
2. **Dosya**'yı, ardından **Yeni** ' yi ve ardından **Proje**' yi tıklatın.
3. *Yeni proje yarat penceresinde* `IBM XMS .NET Client App (.NET Core)` seçeneğini belirleyin ve **İleri**düğmesini tıklatın.
4. *Yeni projenizi yapılandır* penceresinde, isterseniz projenizin *Proje adını* değiştirin ve XMS .NET projesini yaratmak için **Yarat** düğmesini tıklatın.

   `XMSDotnetApp.cs` , proje dosyasıyla birlikte yaratılan dosyadır. Bu dosya, kuyruk yöneticisine bağlanan ve bir gönderme ve alma işlemi gerçekleştiren kodu içerir.

   Bağlantı özellikleri varsayılan değerlere ayarlanır:

   - WMQ_CONNECTION_NAME_LIST *localhost (1414)* olarak ayarlandı
   - XMSC.WMQ_CHANNEL *DOTNET.SVRCONN* olarak ayarlandı

   Kuyruk *Q1*olarak ayarlanır ve bu özellikleri buna göre değiştirebilirsiniz.
5. Uygulamayı derleyin ve çalıştırın.

İlgili kavramlar

IBM MQ bileşenleri ve özellikleri

.NET uygulama yürütme ortamı-yalnızca Windows

İş threading modeli

Genel kurallar, çok iş parçacıklı bir uygulamanın XMS nesnelerini nasıl kullanabileceğini yönetir.

- Farklı iş parçacıklarında eşzamanlı olarak yalnızca aşağıdaki tipteki nesneler kullanılabilir:
  - ConnectionFactory
  - Bağlantı
  - ConnectionMetaVerileri
  - Hedef
- Bir Oturum nesnesi, aynı anda yalnızca tek bir iş parçacığında kullanılabilir.

Bu kurallara ilişkin kural dışı durumlar, IBM Message Service Client for .NET başvuruiçindeki yöntemlerin arabirim tanımlarında "İş Parçacığı bağlamı" etiketli girişlerle gösterilir.

ConnectionFactories ve Connection nesneleri

ConnectionFactory nesnesi, bir uygulamanın Connection nesnesi yaratmak için kullandığı bir şablon sağlar. Uygulama, Oturum nesnesi yaratmak için Connection nesnesini kullanır.

.NETiçin, XMS uygulaması öncelikle gerekli protokol tipine uygun bir ConnectionFactory nesnesine başvuru almak için bir XMSFactoryFactory nesnesi kullanır. Bu ConnectionFactory nesnesi daha sonra yalnızca o protokol tipi için bağlantı üretebilir.

Bir XMS uygulaması birden çok bağlantı yaratabilir ve çok iş parçacıklı bir uygulama, birden çok iş parçacığına eşzamanlı olarak tek bir Connection nesnesini kullanabilir. Bir Connection nesnesi, bir uygulama ile bir ileti sistemi sunucusu arasındaki iletişim bağlantısını içerir.

Bir bağlantı birkaç amaca hizmet eder:

- Bir uygulama bağlantı yarattığında, uygulamanın kimliği doğrulanabilir.
- Bir uygulama, benzersiz bir istemci tanıtıcısını bir bağlantıyla ilişkilendirebilir. İstemci tanıtıcısı, yayınlama/abone olma etki alanında sürekli abonelikleri desteklemek için kullanılır. İstemci tanıtıcısı iki şekilde ayarlanabilir:

  Bir bağlantı istemcisi tanıtıcısı atamanın tercih edilen yolu, özellikleri kullanarak istemciye özgü bir ConnectionFactory nesnesi yapılandırmaktır ve bunu yarattığı bağlantıya saydam bir şekilde atamaktır.

  İstemci tanıtıcısı atamanın diğer bir yolu, Connection nesnesinde ayarlanan sağlayıcıya özgü bir değer kullanmaktır. Bu değer, yönetici tarafından yapılandırılan tanıtıcıyı geçersiz kılmaz. Bu, yönetimsel olarak belirtilmiş bir tanıtıcının olmadığı bir durum için sağlanır. Yönetimsel olarak belirtilen bir tanıtıcı varsa, sağlayıcıya özgü bir değerle geçersiz kılma girişimi kural dışı durum yayınlanmasına neden olur. Bir uygulama belirtik olarak bir tanıtıcı ayarlarsa, bağlantı yaratıldıktan hemen sonra ve bağlantıda başka bir işlem yapılmadan önce bunu yapmalıdır; tersi durumda bir kural dışı durum yayınlanır.

Bir XMS uygulaması genellikle bir bağlantı, bir ya da daha fazla oturum ve bir dizi ileti üreticisi ve ileti tüketicisi yaratır.

Bir bağlantı oluşturmak, bir iletişim bağlantısı kurmayı içerdiğinden ve uygulamanın doğrulanmasını da içerdiğinden, sistem kaynakları açısından nispeten pahalıdır.

Bağlantı başlatıldı ve durduruldu kipi

Bir bağlantı başlatıldı ya da durduruldu kipinde çalışabilir.

Bir uygulama bağlantı yarattığında, bağlantı durduruldu kipinde olur. Bağlantı durdurulduğunda, uygulama oturumları başlatabilir ve ileti gönderebilir, ancak bunları zamanuyumlu ya da zamanuyumsuz olarak alamaz.

Bir uygulama, `Start Connection` yöntemini çağırarak bağlantı başlatabilir. Bağlantı başlatma kipindeyken, uygulama ileti gönderebilir ve alabilir. Daha sonra uygulama, Bağlantıyı Durdur ve `Start Connection` yöntemlerini çağırarak bağlantıyı durdurup yeniden başlatabilir.

Bağlantı kapatma

Bir uygulama, Bağlantıyı Kapat yöntemini çağırarak bağlantıyı kapatır. Bir uygulama bağlantıyı kapattığında XMS aşağıdaki eylemleri gerçekleştirir:

- Bağlantıyla ilişkili tüm oturumları kapatır ve bu oturumlarla ilişkili belirli nesneleri siler. Hangi nesnelerin silindiği hakkında daha fazla bilgi için bkz. “Nesne silme” sayfa 621. Aynı zamanda XMS , oturumlar içinde devam etmekte olan tüm hareketleri geriye işler.
- İleti sistemi sunucusuyla iletişim bağlantısını sona erdirir.
- Bağlantı tarafından kullanılan belleği ve diğer iç kaynakları serbest bırakır.

XMS , bağlantıyı kapatmadan önce, oturum sırasında onaylamadığı iletilerin alındığını kabul etmez. İletilerin alındığının onaylanmasıyla ilgili daha fazla bilgi için bkz. “İleti alındı bildirimi” sayfa 612.

Kural dışı durumları işleme

XMS .NET kural dışı durumların tümü System.Exception' dan türetilir. Daha fazla bilgi için bkz “.NET içinde hata işleme” sayfa 629.

Hizmet bütünleştirme veriyolu bağlantısı

XMS uygulaması, bir WebSphere Application Server hizmet tümleştirme veriyoluna doğrudan TCP/IP bağlantısı kullanarak ya da TCP/IP üzerinden HTTP kullanarak bağlanabilir.

HTTP protokolü, doğrudan TCP/IP bağlantısının mümkün olmadığı durumlarda kullanılabilir. Sık rastlanan bir durum, iki kuruluşun ileti alışverişi gibi bir güvenlik duvarı üzerinden iletişim kurmaktır. Bir güvenlik duvarı üzerinden iletişim kurmak için HTTP ' yi kullanmak genellikle *HTTP tüneli*olarak adlandırılır. HTTP tüneli, ancak HTTP üstbilgileri aktarılan veri miktarına önemli ölçüde eklendiğinden ve HTTP iletişim kuralı TCP/IP ' den daha fazla iletişim akışı gerektirdiğinden, doğrudan TCP/IP bağlantısı kullanılmasından daha yavaş olur.

TCP/IP bağlantısı yaratmak için, bir uygulama XMSC_WPM_TARGET_TRANSPORT_CHAIN özelliği XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC olarak ayarlanmış bir bağlantı üreticisini kullanabilir. Bu, özelliğin varsayılan değeridir. Bağlantı başarıyla yaratılırsa, bağlantının XMSC_WPM_CONNECTION_PROTOCOL özelliği XMSC_WPM_CP_TCP olarak ayarlanır.

HTTPkullanan bir bağlantı yaratmak için, bir uygulamanın XMSC_WPM_TARGET_TRANSPORT_CHAIN özelliği, HTTP iletim kanalı kullanacak şekilde yapılandırılmış bir gelen iletim zincirinin adına ayarlanmış bir bağlantı üreticisini kullanması gerekir. Bağlantı başarıyla yaratılırsa, bağlantının XMSC_WPM_CONNECTION_PROTOCOL özelliği XMSC_WPM_CP_HTTP olarak ayarlanır. İletim zincirlerinin nasıl yapılandırılacağı hakkında bilgi için WebSphere Application Server ürün belgelerinde İletim zincirlerinin yapılandırılması başlıklı konuya bakın.

Bir uygulama, bir önyükleme sunucusuna bağlanırken benzer iletişim protokolü seçeneğine sahiptir. Bir bağlantı üreticisinin XMSC_WPM_PROVIDER_ENDPOINTS özelliği, önyükleme sunucularının bir ya da daha çok uç noktası adresi dizisidir. Her uç nokta adresinin önyükleme iletim zinciri bileşeni, bir önyükleme sunucusuna TCP/IP bağlantısı için XMSC_WPM_BOOTSTRAP_TCP ya da HTTPkullanan bir bağlantı için XMSC_WPM_BOOTSTRAP_HTTP olabilir.

Oturumlar

Oturum, ileti göndermek ve almak için tek iş parçacıklı bir bağlamdır.

Bir uygulama, ileti, ileti üreticileri, ileti tüketicileri, kuyruk tarayıcıları ve geçici hedefler yaratmak için oturum kullanabilir. Bir uygulama, yerel hareketleri çalıştırmak için bir oturumu da kullanabilir.

Bir uygulama, her oturumun iletileri diğer oturumlardan bağımsız olarak ürettiği ve kullandığı birden çok oturum yaratabilir. Ayrı oturumlarda (ya da aynı oturumda) iki ileti tüketicisi aynı konuya abone olursa, her biri o konuda yayınlanan iletinin bir kopyasını alır.

Bir Connection nesnesinden farklı olarak, bir Session nesnesi farklı iş parçacıklarında eşzamanlı olarak kullanılamaz. Oturum nesnesinin o sırada kullandığı iş parçacığından başka bir iş parçacığından yalnızca

Oturum kapatma yöntemi çağrılabilir. Oturumu Kapat yöntemi bir oturumu sonlandırır ve oturuma ayrılmış sistem kaynaklarını serbest bırakır.

Bir uygulamanın iletileri aynı anda birden çok iş parçacığında işlemesi gerekiyorsa, uygulama her iş parçacığında bir oturum yaratmalı ve o iş parçacığı içindeki herhangi bir gönderme ya da alma işlemi için bu oturumu kullanmalıdır.

İşlem uygulanan oturumlar

XMS uygulamaları yerel hareketleri çalıştırabilir. *Yerel hareket* , yalnızca uygulamanın bağlı olduğu kuyruk yöneticisinin ya da hizmet tümleştirme veriyolunun kaynaklarında yapılan değişiklikleri içeren bir harekettir.

Bu konudaki bilgiler yalnızca bir uygulama bir IBM MQ kuyruk yöneticisine ya da WebSphere Application Server hizmet tümleştirme veriyoluna bağlandığında anlamlıdır. Bilgiler, bir aracıya gerçek zamanlı bağlantı için uygun değildir.

Yerel hareketleri çalıştırmak için, bir uygulamanın öncelikle, oturumun hareketinin gerçekleştirildiğini belirten bir değiştirge olarak bir Bağlantı nesnesinin Oturum Yarat yöntemini çağırarak hareket uygulanan bir oturum yaratması gerekir. Daha sonra, oturum içinde gönderilen ve alınan tüm iletiler bir işlem dizisi içinde gruplanır. Hareket, uygulama, hareket başladığından bu yana gönderdiği ve aldığı iletileri kesinleştirdiğinde ya da geri aldığında sona erer.

Bir hareketi kesinleştirmek için, uygulama Oturum nesnesinin Kesinleştirme yöntemini çağırır. Bir işlem kesinleştirildiğinde, hareket içinde gönderilen tüm iletiler diğer uygulamalara teslim edilmek üzere kullanılabilir olur ve hareket içinde alınan tüm iletiler, ileti sistemi sunucusunun bunları uygulamaya yeniden teslim etmeye çalışmaması için onaylanır. Noktadan noktaya iletişim etki alanında, ileti sistemi sunucusu alınan iletileri kuyruklarından da kaldırır.

Bir hareketi geriye işlemek için uygulama, Oturum nesnesinin Geri Alma yöntemini çağırır. Bir hareket geriye işlendiğinde, hareket içinde gönderilen tüm iletiler ileti sistemi sunucusu tarafından atılır ve hareket içinde alınan tüm iletiler yeniden teslim için kullanılabilir duruma gelir. Noktadan noktaya iletişim etki alanında, alınan iletiler kuyruklarına geri konur ve diğer uygulamalar tarafından yeniden görünür hale gelir.

Bir uygulama işlemli bir oturum yarattığında ya da Kesinleştirme ya da Geri Alma yöntemini çağırdığında yeni bir hareket otomatik olarak başlar. Bu nedenle, işlem yapılan bir oturumun her zaman etkin bir hareketi vardır.

Bir uygulama işlemli bir oturumu kapattığında, örtük bir geri alma gerçekleşir. Bir uygulama bir bağlantıyı kapattığında, bağlantının işlem yapılan tüm oturumları için örtük bir geri alma oluşur.

Bir işlemin tamamı işlemli bir oturumda yer alır. Bir işlem oturumlara yayılamaz. Başka bir deyişle, bir uygulama iki ya da daha çok hareket içeren oturumda ileti gönderip alamaz ve tüm bu işlemleri tek bir hareket olarak kesinleştiremez ya da geriye işleyemez.

İlgili kavramlar

İleti alındı bildirimi

İşlem yapılmayan her oturumda, uygulama tarafından alınan iletilerin nasıl alındığını belirleyen bir alındı bildirimi kipi vardır. Üç alındı bildirimi kipi vardır ve alındı bildirimi kipinin seçimi, uygulamanın tasarımını etkiler.

İleti teslimi

XMS , ileti tesliminin kalıcı ve kalıcı olmayan kiplerini ve iletilerin zamanuyumsuz ve zamanuyumlu teslimini destekler.

İleti alındı bildirimi

İşlem yapılmayan her oturumda, uygulama tarafından alınan iletilerin nasıl alındığını belirleyen bir alındı bildirimi kipi vardır. Üç alındı bildirimi kipi vardır ve alındı bildirimi kipinin seçimi, uygulamanın tasarımını etkiler.

Bu konudaki bilgiler yalnızca bir uygulama bir IBM MQ kuyruk yöneticisine ya da WebSphere Application Server hizmet tümleştirme veriyoluna bağlandığında anlamlıdır. Bilgiler, bir aracıya gerçek zamanlı bağlantı için uygun değildir.

XMS , JMS ' nin kullandığı iletilerin alındığını kabul etmek için aynı mekanizmayı kullanır.

Bir oturum işlemden geçirilmezse, uygulama tarafından alınan iletilerin alınma şekli, oturumun alındı bildirimi kipine göre belirlenir. Üç alındı bildirimi kipi aşağıdaki paragraflarda açıklanmıştır:

XMSC_AUTO_ONAY
Oturum, uygulama tarafından alınan her iletiyi otomatik olarak onaylar.

İletiler uygulamaya zamanuyumlu olarak teslim edilirse, oturum, bir Alma çağrısının başarıyla tamamlandığı her zaman bir iletinin alındığını onaylar.

Uygulama başarılı bir şekilde bir ileti alırsa, ancak bir hata alındı bildiriminin oluşmasını engellerse, ileti yeniden teslim edilebilir duruma gelir. Bu nedenle uygulama, yeniden teslim edilen bir iletiyi işleyebilmelidir.

XMSC_DUPS_OK_ONAY
Oturum, seçtiği zaman uygulama tarafından alınan iletileri onaylar.

Bu alındı bildirimi kipinin kullanılması, oturumun yapması gereken iş miktarını azaltır, ancak ileti onayını önleyen bir hata, yeniden teslim için birden çok iletinin kullanılabilir olmasına neden olabilir. Bu nedenle uygulama, yeniden teslim edilen iletileri işleyebilmelidir.

XMSC_CLIENT_ONAY
Uygulama, İleti sınıfının Onay yöntemini çağırarak aldığı iletileri onaylar.

Uygulama, her bir iletinin ayrı ayrı alındığını onaylayabilir ya da bir ileti kümesi alabilir ve yalnızca aldığı son ileti için Onay yöntemini çağırabilir. Alındı bildirimi yöntemi çağrıldığında, yöntemin son çağrılmasından bu yana alınan tüm iletiler alınır.

Bu alındı bildirimi kiplerinin herhangi biriyle birlikte, bir uygulama Oturum sınıfının Kurtar yöntemini çağırarak bir oturumda ileti teslimini durdurabilir ve yeniden başlatabilir. Daha önce alındı bildirimi alınmamış olan iletiler yeniden teslim edilir. Ancak, bunlar daha önce teslim edildikleri sırayla teslim edilmeyebilir. Bu arada, daha yüksek öncelikli iletiler gelmiş olabilir ve özgün iletilerin bazılarının süresi dolmuş olabilir. Noktadan noktaya iletişim etki alanında, özgün iletilerin bazıları başka bir uygulama tarafından tüketilmiş olabilir.

Bir uygulama, iletinin JMSRegönderilen üstbilgi alanının içeriğini inceleyerek iletinin yeniden teslim edilip edilmediğini belirleyebilir. Uygulama bunu, İleti sınıfının JMSRegönderilen Alma yöntemini çağırarak yapar.

İlgili kavramlar
İşlem uygulanan oturumlar
XMS uygulamaları yerel hareketleri çalıştırabilir. *Yerel hareket* , yalnızca uygulamanın bağlı olduğu kuyruk yöneticisinin ya da hizmet tümleştirme veriyolunun kaynaklarında yapılan değişiklikleri içeren bir harekettir.

İleti teslimi
XMS , ileti tesliminin kalıcı ve kalıcı olmayan kiplerini ve iletilerin zamanuyumsuz ve zamanuyumlu teslimini destekler.

İleti teslimi
XMS , ileti tesliminin kalıcı ve kalıcı olmayan kiplerini ve iletilerin zamanuyumsuz ve zamanuyumlu teslimini destekler.

İleti teslim kipi

XMS iki ileti teslim kipini destekler:

Kalıcı
Kalıcı iletiler bir kez teslim edilir. Bir ileti sistemi sunucusu, iletilerin günlüğe kaydedilmesi gibi, bir hata durumunda bile iletilerin iletide kaybolmamasını sağlamak için özel önlemler alır.

Kalıcı olmayan
Kalıcı olmayan iletiler bir kereden fazla teslim edilmez. Kalıcı olmayan iletiler, bir hata durumunda geçiş sırasında kaybolabildiğinden kalıcı olmayan iletilere göre daha az güvenilirdir.

Teslim modunun seçimi, güvenilirlik ve performans arasında bir değiş tokuştur. Kalıcı olmayan iletiler genellikle kalıcı iletilerden daha hızlı aktarılır.

Zamanuyumsuz ileti teslimi

XMS , bir oturuma ilişkin tüm zamanuyumsuz ileti teslimatlarını işlemek için bir iş parçacığı kullanır. Bu, bir kerede tek bir ileti dinleyici işlevi ya da tek bir `onMessage()` yönteminin çalışabileceği anlamına gelir.

Bir oturumda birden çok ileti tüketicisi zamanuyumsuz olarak ileti alıyorsa ve bir ileti dinleyici işlevi ya da `onMessage()` yöntemi bir ileti tüketicisine ileti teslim alıyorsa, aynı iletiyi bekleyen diğer ileti tüketicilerinin de beklemeye devam etmesi gerekir. Oturuma teslim edilmeyi bekleyen diğer iletiler de beklemeye devam etmelidir.

Bir uygulama, iletilerin eşzamanlı olarak teslim edilmesini gerektiriyorsa, XMS ' un zamanuyumsuz ileti teslimini işlemek için birden çok iş parçacığı kullanması için birden çok oturum oluşturun. Bu şekilde, birden çok ileti dinleyici işlevi ya da `onMessage()` yöntemi eşzamanlı olarak çalışabilir.

Bir tüketiciye ileti dinleyicisi atanarak oturum zamanuyumsuz yapılmaz. Bir oturum yalnızca `Connection.Start` yöntemi çağrıldığında zamanuyumsuz olur. `Connection.Start` yöntemi çağrılıncaya kadar tüm zamanuyumlu çağrılara izin verilir. `Connection.Start` çağrıldığında tüketicilere ileti teslimi başlar.

Zamanuyumsuz bir oturumda tüketici ya da üretici yaratma gibi zamanuyumlu çağrıların çağrılması gerekiyorsa, `Connection.Stop` çağrılmalıdır. İletilerin teslimini başlatmak için `Connection.Start` yöntemi çağrılarak bir oturum sürdürülebilir. Bunun tek kural dışı durumu, iletileri geri çağırma işlevine teslim eden iş parçacığı olan Oturum iletisi teslim iş parçacığıdır. Bu iş parçacığı, ileti geri çağırma işlevinde (Kapatma çağrısı dışında) oturum üzerinde herhangi bir çağrı yapabilir.

Not: Yönetilmeyen kipte, bir geri çağırma işlevindeki MQDISC çağrısı IBM MQ .NET istemcisi tarafından desteklenmez. Bu nedenle, istemci uygulaması Zamanuyumsuz alma kipinde MessageListener geri çağrısı içinde oturum yaratamadı ya da kapatamadı. Oturumu `MessageListener` yönteminin dışında yaratın ve imha edin.

Zamanuyumlu ileti teslimi

Uygulama MessageConsumer nesnelerinin Alma yöntemlerini kullanıyorsa, iletiler bir uygulamaya zamanuyumlu olarak teslim edilir.

Alma (Receive) yöntemlerini kullanarak, bir uygulama bir ileti için belirli bir süre bekleyebilir ya da süresiz olarak bekleyebilir. Diğer bir seçenek olarak, bir uygulama ileti beklemek istemiyorsa, Beklemeyen Alma (Receive with No Wait) yöntemini kullanabilir.

İlgili kavramlar

İşlem uygulanan oturumlar
XMS uygulamaları yerel hareketleri çalıştırabilir. *Yerel hareket* , yalnızca uygulamanın bağlı olduğu kuyruk yöneticisinin ya da hizmet tümleştirme veriyolunun kaynaklarında yapılan değişiklikleri içeren bir harekettir.

İleti alındı bildirimi
İşlem yapılmayan her oturumda, uygulama tarafından alınan iletilerin nasıl alındığını belirleyen bir alındı bildirimi kipi vardır. Üç alındı bildirimi kipi vardır ve alındı bildirimi kipinin seçimi, uygulamanın tasarımını etkiler.

Hedefler

XMS uygulaması, gönderilmekte olan iletilerin hedefini ve alınmakta olan iletilerin kaynağını belirtmek için bir Hedef nesnesi kullanır.

XMS uygulaması, yürütme sırasında bir Hedef nesne yaratabilir ya da yönetilen nesneler havuzundan önceden tanımlanmış bir hedef elde edebilir.

ConnectionFactory' de olduğu gibi, bir XMS uygulamasının hedef belirtmesinin en esnek yolu, hedefi yönetilen nesne olarak tanımlamaktır. Bu yaklaşımı kullanarak, C, C + + ve .NET dillerinde yazılan

uygulamalar ve Java, hedefin tanımlarını paylaşabilir. Denetlenen Hedef nesnelerinin özellikleri, herhangi bir kod değiştirilmeden değiştirilebilir.

.NET uygulamaları için, CreateTopic ya da CreateQueue yöntemini kullanarak hedef yaratırsınız. Bu iki yöntem, .NET API 'sindeki hem ISession hem de XMSFactoryFactory nesnelerinde kullanılabilir. Daha fazla bilgi için bkz. “.NET içindeki hedefler” sayfa 627 ve ../refdev/sapidest.dita#sapidest.

Konu birörnek kaynak tanıtıcıları

Konu birörnek kaynak tanıtıcısı (URI), konunun adını belirtir; konu için bir ya da daha fazla özellik de belirtebilir.

Bir konuya ilişkin URI şu sıra konusuyla başlar: //, ardından konunun adı ve (isteğe bağlı) geri kalan konu özelliklerini ayarlanan ad-değer çiftlerinin bir listesi. Konu adı boş olamaz.

Aşağıda, .NET kodunun bir parçasındaki bir örnek verilmiştir:

```
topic = session.CreateTopic("topic://Sport/Football/Results?multicast=7");
```

Bir URI ' de kullanabileceğiniz ad ve geçerli değerler de içinde olmak üzere bir konunun özellikleri hakkında daha fazla bilgi için Hedef Özellikleribaşlıklı konuya bakın.

Abonelikte kullanılacak bir konu URI 'si belirtilirken genel arama karakterleri kullanılabilir. Bu joker karakterlerin sözdizimi bağlantı tipine ve aracı sürümüne bağlıdır; aşağıdaki seçenek kullanılabilir:

- WebSphere Application Server hizmet bütünleştirme veriyolu

WebSphere Application Server hizmet bütünleştirme veriyolu

WebSphere Application Server hizmet bütünleştirme veriyolu aşağıdaki genel arama karakterlerini kullanır:

* sıradüzeninde bir düzeydeki herhangi bir karakteri eşleştirmek için
// 0 ya da daha fazla düzey eşleştirmek için
//. 0 ya da daha fazla düzeyi eşleştirmek için (Konu ifadesinin sonunda)

Çizelge 82 sayfa 615 , bu genel arama karakteri şemasının nasıl kullanılacağına ilişkin bazı örnekler verir.

Çizelge 82. WebSphere Application Server hizmet bütünleştirme veriyolu için genel arama karakteri şeması kullanan örnek URI ' ler

Birörnek Kaynak Tanıtıcısı	**Eşleşir**	**Örnekler**
"topic://Sport/ * ball/ Results"	Spor ve Sonuçlar arasında "top" ile biten tek bir hiyerarşik düzey adı olan tüm konular	"topic://Sport/Football/Results" ve "topic://Sport/Netball/Results"
"topic://Sport// Results"	"Sport/" ile başlayan ve "/Results" ile biten tüm konular	"topic://Sport/Football/Results" ve "topic://Sport/Hockey/National/Div3/Results"
"topic://Sport/ Football//."	"Spor/Futbol/" ile başlayan tüm konular	"topic://Sport/Football/Results" ve "topic://Sport/Football/TeamNews/Signings/Managerial"
"topic://Sport/ * ball// Results//."	Konular	"topic://Sport/Football/Results" ve "topic://Sport/Netball/National/Div3/Results/2002/November"

İlgili kavramlar

Kuyruk birörnek kaynak tanıtıcıları
Bir kuyruğa ilişkin URI, kuyruğun adını belirtir; kuyruğun bir ya da daha çok özelliğini de belirtebilir.

Geçici hedefler

XMS uygulamaları geçici hedefler oluşturabilir ve kullanabilir.

Kuyruk birörnek kaynak tanıtıcıları

Bir kuyruğa ilişkin URI, kuyruğun adını belirtir; kuyruğun bir ya da daha çok özelliğini de belirtebilir.

Bir kuyruğa ilişkin URI, `queue://` sırasıyla başlar ve bunu kuyruğun adı izler; ayrıca, geri kalan kuyruk özelliklerini ayarlanan ad-değer çiftlerinin bir listesini de içerebilir.

IBM MQ kuyrukları için (ancak, WebSphere Application Server varsayılan ileti alışverişi sağlayıcısı kuyrukları için değil), kuyruğun bulunduğu kuyruk yöneticisi kuyruktan önce belirtilebilir ve kuyruk yöneticisi adını kuyruk adından ayıran/ayıran bir kuyruk yöneticisi olabilir.

Bir kuyruk yöneticisi belirtilirse, bu kuyruk kullanılarak bağlantı için XMS ' in doğrudan bağlandığı ya da bu kuyruktan erişilebilen bir kuyruk yöneticisi olmalıdır. Uzak kuyruk yöneticileri, kuyruklara ileti koymak için değil, yalnızca kuyruklardan ileti almak için desteklenir. Tüm ayrıntılar için IBM MQ kuyruk yöneticisi belgelerine bakın.

Kuyruk yöneticisi belirtilmezse, fazladan/ayırıcı isteğe bağlıdır ve kuyruğun varlığı ya da yokluğu, kuyruk tanımlamasında bir fark yaratmaz.

Aşağıdaki kuyruk tanımlamalarının tümü, XMS ' un doğrudan bağlı olduğu QM_A adlı bir kuyruk yöneticisinde QB adı verilen bir IBM MQ kuyruğunun eşdeğeridir:

```
queue://QB
queue:///QB
queue://QM_A/QB
```

İlgili kavramlar

Konu birörnek kaynak tanıtıcıları
Konu birörnek kaynak tanıtıcısı (URI), konunun adını belirtir; konu için bir ya da daha fazla özellik de belirtebilir.

Geçici hedefler
XMS uygulamaları geçici hedefler oluşturabilir ve kullanabilir.

Geçici hedefler

XMS uygulamaları geçici hedefler oluşturabilir ve kullanabilir.

Bir uygulama genellikle istek iletilerine yanıt almak için geçici bir hedef kullanır. Bir istek iletisine yanıt gönderileceği hedefi belirtmek için, bir uygulama istek iletisini gösteren İleti nesnesinin JMSReplyTo yöntemini ayarlar. Çağrıda belirtilen hedef geçici bir hedef olabilir.

Geçici bir hedef yaratmak için oturum kullanılsa da, geçici bir hedefin kapsamı, oturumu yaratmak için kullanılan bağlantıdır. Bağlantının oturumlarından herhangi biri, geçici hedef için ileti üreticileri ve ileti tüketicileri yaratabilir. Geçici hedef, belirtik olarak silininceye ya da bağlantı sona erinceye kadar (hangisi önce gerçekleşirse) kalır.

Bir uygulama geçici bir kuyruk yarattığında, uygulamanın bağlı olduğu ileti sistemi sunucusunda bir kuyruk yaratılır. Uygulama bir kuyruk yöneticisine bağlıysa, adı XMSC_WMQ_TEMPORARY_MODEL özelliği tarafından belirtilen model kuyruğundan dinamik bir kuyruk yaratılır ve dinamik kuyruğun adını oluşturmak için kullanılan önek XMSC_WMQ_TEMP_Q_PREFIX özelliğiyle belirtilir. Uygulama bir hizmet tümleştirme veriyoluna bağlıysa, veriyolunda geçici bir kuyruk yaratılır ve geçici kuyruk adını oluşturmak için kullanılan önek XMSC_WPM_TEMP_Q_PREFIX özelliği tarafından belirtilir.

Bir hizmet tümleştirme veriyoluna bağlı bir uygulama geçici bir konu yarattığında, geçici konunun adını oluşturmak için kullanılan önek XMSC_WPM_TEMP_TOPIC_PREFIX özelliğiyle belirtilir.

İlgili kavramlar

Konu birörnek kaynak tanıtıcıları
Konu birörnek kaynak tanıtıcısı (URI), konunun adını belirtir; konu için bir ya da daha fazla özellik de belirtebilir.

Kuyruk birörnek kaynak tanıtıcıları
Bir kuyruğa ilişkin URI, kuyruğun adını belirtir; kuyruğun bir ya da daha çok özelliğini de belirtebilir.

İleti üreticileri

XMSiçinde, bir ileti üreticisi geçerli bir hedefle ya da ilişkili bir hedef olmadan yaratılabilir. Boş değerli hedefle ileti üreticisi yaratılırken, ileti gönderilirken geçerli bir hedef belirtilmelidir.

İlişkili hedefi olan ileti üreticileri

Bu senaryoda, ileti üreticisi geçerli bir hedef kullanılarak yaratılır. Gönderme işlemi sırasında, hedefin belirtilmesi gerekmez.

İlişkili hedefi olmayan ileti üreticileri

XMS .NETolanağında, boş değerli bir hedefle bir ileti üreticisi yaratılabilir.

.NET API kullanılırken ilişkili hedefi olmayan bir ileti üreticisi yaratmak için, ISession nesnesinin `CreateProducer()` yöntemine parametre olarak NULL geçirilmelidir (örneğin, `session.CreateProducer(null)`). Ancak, ileti gönderildiğinde geçerli bir hedef belirtilmelidir.

İleti tüketicileri

İleti tüketicileri, sürekli ve dayanıklı olmayan aboneler ve zamanuyumlu ve zamanuyumsuz ileti tüketicileri olarak sınıflandırılabilir.

Sürekli aboneler

Sürekli abone, abone etkin değilken yayınlanan iletiler de içinde olmak üzere, bir konuda yayınlanan tüm iletileri alan bir ileti tüketicisidir.

Bu konudaki bilgiler yalnızca bir uygulama bir IBM MQ kuyruk yöneticisine ya da WebSphere Application Server hizmet tümleştirme veriyoluna bağlandığında anlamlıdır. Bilgiler, bir aracıya gerçek zamanlı bağlantı için uygun değildir.

Bir konuya ilişkin sürekli bir abone yaratmak için, bir uygulama, sürekli aboneliği tanıtan bir ad ve konuyu gösteren bir Hedef nesne olarak belirterek, Oturum nesnesinin Durabilir Abone Yarat yöntemini çağırır. Uygulama, ileti seçici ile ya da ileti seçici olmadan sürekli bir abone yaratabilir ve sürekli abonenin kendi bağlantısıyla yayınlanan iletileri alıp almayacağını belirtebilir.

Kalıcı bir abone yaratmak için kullanılan oturumun ilişkili bir istemci tanıtıcısı olmalıdır. İstemci tanıtıcısı, oturumu yaratmak için kullanılan bağlantıyla ilişkilendirilmiş tanıtıcıyla aynıdır; “ConnectionFactories ve Connection nesneleri” sayfa 610içinde açıklandığı gibi belirtilir.

Sürekli aboneliği tanımlayan ad, istemci tanıtıcısı içinde benzersiz olmalıdır ve bu nedenle istemci tanıtıcısı, sürekli aboneliğin tam, benzersiz tanıtıcısının bir parçasını oluşturur. İleti sistemi sunucusu sürekli aboneliğin bir kaydını tutar ve konu üzerinde yayınlanan tüm iletilerin sürekli abone tarafından alınıncaya ya da süresi doluncaya kadar korunmasını sağlar.

İleti alışverişi sunucusu, sürekli abone kapandıktan sonra bile sürekli abonelik kaydını tutmaya devam eder. Daha önce yaratılmış sürekli bir aboneliği yeniden kullanmak için, bir uygulamanın aynı abonelik adını belirten sürekli bir abone yaratması ve kalıcı abonelikle ilişkilendirilmiş olanlarla aynı istemci tanıtıcısına sahip bir oturum kullanması gerekir. Aynı anda yalnızca bir oturum, belirli bir sürekli abonelik için sürekli bir aboneye sahip olabilir.

Sürekli aboneliğin kapsamı, aboneliğin kaydını koruyan ileti sistemi sunucusudur. Farklı ileti sistemi sunucularına bağlı iki uygulama aynı abonelik adını ve istemci tanıtıcısını kullanarak sürekli bir abone oluşturursa, tamamen bağımsız iki sürekli abonelik yaratılır.

Sürekli aboneliği silmek için bir uygulama, sürekli aboneliği tanıtan bir değiştirge olarak adı belirterek Oturum nesnesinin Aboneliğini Kaldırma yöntemini çağırır. Oturumla ilişkili istemci tanıtıcısı, sürekli abonelikle ilişkili istemci tanıtıcısıyla aynı olmalıdır. İleti alışverişi sunucusu, bakımını yaptığı sürekli aboneliğin kaydını siler ve kalıcı aboneye başka ileti göndermez.

Var olan bir aboneliği değiştirmek için, bir uygulama aynı abonelik adını ve istemci tanıtıcısını kullanarak sürekli bir abone yaratabilir, ancak farklı bir konu ya da ileti seçici (ya da her ikisi) belirtebilir. Sürekli bir aboneliğin değiştirilmesi, aboneliğin silinmesine ve yeni bir abonelik oluşturulmasına eşdeğerdir.

Bir IBM MQ kuyruk yöneticisine bağlanan bir uygulama için, XMS abone kuyruklarını yönetir. Bu nedenle, uygulamanın bir abone kuyruğu belirtmesi gerekmez. Belirtildiyse, XMS abone kuyruğunu yoksayar.

Sürekli abonelik için abone kuyruğunu değiştiremeyeceğinizi unutmayın. Abone kuyruğunu değiştirmenin tek yolu, aboneliği silip yeni bir tane yaratmaktır.

Bir hizmet bütünleştirme veriyoluna bağlanan bir uygulama için, her sürekli abonenin belirlenmiş bir sürekli abonelik ana sayfası olmalıdır. Aynı bağlantıyı kullanan tüm sürekli aboneler için sürekli abonelik ana sayfasını belirtmek üzere, bağlantıyı yaratmak için kullanılan ConnectionFactory nesnesinin XMSC_WPM_DUR_SUB_HOME özelliğini ayarlayın. Tek bir konuya ilişkin sürekli abonelik ana sayfasını belirtmek için, konuyu gösteren Hedef nesnenin XMSC_WPM_DUR_SUB_HOME özelliğini ayarlayın. Bir uygulamanın bağlantıyı kullanan sürekli bir abone yaratabilmesi için, bağlantı için kalıcı bir abonelik ana dizini belirtilmelidir. Hedef için belirtilen herhangi bir değer, bağlantı için belirtilen değeri geçersiz kılar.

Zamanuyumlu ve zamanuyumsuz ileti tüketicileri

Zamanuyumlu ileti tüketicisi iletileri zamanuyumlu olarak kuyruktan alır ve zamanuyumsuz ileti tüketicisi, bir kuyruktan zamanuyumsuz olarak ileti alır.

Zamanuyumlu ileti tüketicileri

Zamanuyumlu ileti tüketicisi bir kerede bir ileti alır. `Receive(wait interval)` yöntemi kullanıldığında çağrı, ileti için milisaniye cinsinden yalnızca belirli bir süre bekler ya da ileti tüketicisi kapatılıncaya kadar bekler.

ReceiveNoWait () yöntemi kullanılırsa, zamanuyumlu ileti tüketicisi iletileri gecikmeden alır; sonraki ileti kullanılabiliyorsa hemen alınır; tersi durumda, boş değerli bir ileti nesnesine ilişkin bir gösterge döndürülür.

Zamanuyumsuz ileti tüketicileri

Kuyrukta yeni bir ileti varsa, uygulama tarafından kaydedilen ileti dinleyici çağrılır.

XMS içindeki zehirli iletiler

Zehirli ileti, alan MDB uygulaması tarafından işlenemeyen bir iletidir. Zehirli bir iletiyle karşılaşılırsa, XMS MessageConsumer nesnesi bunu iki kuyruk özelliğine (BOQUEUE ve BOTHRESH) göre yeniden kuyruğa alabilir.

Bazı durumlarda, MDB ' ye teslim edilen bir ileti IBM MQ kuyruğuna geri işlenebilir. Bu durum, örneğin, daha sonra geriye işlenen bir iş birimi içinde bir ileti teslim edilirse oluşabilir. Geriye işlenen bir ileti genellikle yeniden teslim edilir, ancak yanlış biçimlendirilmiş bir ileti defalarca bir MDB ' nin başarısız olmasına neden olabilir ve bu nedenle teslim edilemez. Böyle bir mesaj zehirli mesaj olarak adlandırılır. IBM MQ ' i, zehirli iletinin daha ayrıntılı inceleme için otomatik olarak başka bir kuyruğa aktarılması ya da atılması için yapılandırabilirsiniz. IBM MQ 'in bu şekilde nasıl yapılandırılacağına ilişkin bilgi için bkz. ASF' de zehirli iletilerin işlenmesi.

Bazen, hatalı biçimlendirilmiş bir ileti kuyruğa gelir. Bu bağlamda, hatalı biçimlendirilmiş, alan uygulamanın iletiyi doğru işleyemeyeceği anlamına gelir. Böyle bir ileti, alan uygulamanın başarısız olmasına ve bu hatalı biçimlendirilmiş iletiyi geri çekmesine neden olabilir. Daha sonra, ileti giriş kuyruğuna sürekli olarak teslim edilebilir ve uygulama tarafından arka arkaya yedeklenebilir. Bu mesajlar zehirli mesajlar olarak bilinir. XMS MessageConsumer nesnesi, zehirli iletileri algılar ve bunları alternatif bir hedefe yeniden yönlendirir.

IBM MQ kuyruk yöneticisi, her iletinin kaç kez geriletildiğini kaydeder. Bu sayı yapılandırılabilir bir eşik değerine ulaştığında, ileti tüketicisi iletiyi adlandırılmış bir geriletme kuyruğuna yeniden gönderir. Bu istek herhangi bir nedenle başarısız olursa, ileti giriş kuyruğundan kaldırılır ve teslim edilmeyen iletiler kuyruğuna yeniden gönderilir ya da atılır.

XMS ConnectionConsumer nesneleri, zehirli iletileri aynı şekilde ve aynı kuyruk özelliklerini kullanarak işliyor. Birden çok bağlantı tüketicisi aynı kuyruğu izliyorsa, zehirli ileti bir uygulamaya, istek oluşmadan önce eşik değerinden daha fazla teslim edilebilir. Bu davranış, tek tek bağlantı tüketicilerinin kuyrukları izleme ve zehirli iletileri yeniden kuyruğa alma biçiminden kaynaklanmaktadır.

Eşik değeri ve geri alma kuyruğunun adı, bir IBM MQ kuyruğunun öznitelikleridir. Özniteliklerin adları BackoutThreshold ve BackoutRequeueQName 'tir. Uygulandıkları kuyruk aşağıdaki gibidir:

- Noktadan noktaya iletişim ileti sistemi için bu, temeldeki yerel kuyruktur. Bu, ileti tüketicileri ve bağlantı tüketicileri kuyruk diğer adlarını kullandığında önemlidir.
- IBM MQ ileti alışverişi sağlayıcısı normal kipinde yayınlama/abone olma ileti sistemi için, Konu 'nun yönetilen kuyruğunun yaratıldığı model kuyruğudur.
- IBM MQ ileti alışverişi sağlayıcısı geçiş kipinde yayınlama/abone olma ileti sistemi için, TopicConnectionFactory nesnesinde ya da Konu nesnesinde tanımlanan CCDSUB kuyruğunda tanımlanan CCSUB kuyruğudur.

BackoutThreshold ve BackoutRequeueQName özniteliklerini ayarlamak için aşağıdaki MQSC komutunu verin:

```
ALTER QLOCAL(your.queue.name) BOTHRESH(threshold value)
BOQUEUE(your.backout.queue.name)
```

Yayınlama/abone olma ileti sistemi için, sisteminiz her abonelik için dinamik bir kuyruk oluştururs a, bu öznitelik değerleri IBM MQ classes for JMS model kuyruğu SYSTEM.JMS.MODEL.QUEUE. Bu ayarları değiştirmek için aşağıdakileri kullanın:

```
ALTER QMODEL(SYSTEM.JMS.MODEL.QUEUE) BOTHRESH(threshold value)
BOQUEUE(your.backout.queue.name)
```

Geriletme eşiği değeri sıfırsa, zehirli ileti işleme devre dışı bırakılır ve zehirli iletiler giriş kuyruğunda kalır. Tersi durumda, geriletme sayısı eşik değerine ulaştığında, ileti adı belirtilen geriletme kuyruğuna gönderilir.

Geriletme sayısı eşik değerine ulaşırsa, ancak ileti geriletme kuyruğuna gidemezse, ileti gitmeyen iletiler kuyruğuna gönderilir ya da ileti kalıcı değilse atılır.

Geriletme kuyruğu tanımlanmamışsa ya da MessageConsumer nesnesi iletiyi geriletme kuyruğuna gönderemezse bu durum oluşur.

Sisteminizin zehirli ileti işlemeyi gerçekleştirecek şekilde yapılandırılması

XMS .NET ' in **BOTHRESH** ve **BOQNAME** özniteliklerini sorgularken kullandığı kuyruk, gerçekleştirilmekte olan ileti alışverişi stiline bağlıdır:

- Noktadan noktaya iletişim ileti sistemi için bu, temeldeki yerel kuyruktur. Bir XMS .NET uygulaması, diğer ad kuyruklarından ya da küme kuyruklarından gelen iletileri tüketiyorken bu önemlidir.
- Yayınlama/abone olma ileti sistemi için, bir uygulamaya ilişkin iletileri tutmak üzere yönetilen bir kuyruk yaratılır. XMS .NET , **BOTHRESH** ve **BOQNAME** özniteliklerine ilişkin değerleri belirlemek için yönetilen kuyruğu sorgular.

  Yönetilen kuyruk, uygulamanın abone olduğu Konu nesnesiyle ilişkili bir model kuyruğundan oluşturulur ve **BOTHRESH** ve **BOQNAME** özniteliklerinin değerlerini model kuyruğundan devralır. Kullanılan model kuyruğu, alan uygulamanın kalıcı ya da kalıcı olmayan bir aboneliği çıkarmasına bağlıdır:

  – Sürekli abonelikler için kullanılan model kuyruğu, Konunun **MDURMDL** özniteliği tarafından belirtilir. Bu özniteliğin varsayılan değeri SYSTEM.DURABLE.MODEL.QUEUE.

  – Kalıcı olmayan abonelikler için, kullanılan model kuyruğu **MNDURMDL** özniteliği tarafından belirtilir. **MNDURMDL** özniteliğinin varsayılan değeri SYSTEM.NDURABLE.MODEL.QUEUE.

BOTHRESH ve **BOQNAME** özniteliklerini sorgularken XMS .NET:

- Yerel kuyruğu ya da diğer ad kuyruğuna ilişkin hedef kuyruğu açar.
- **BOTHRESH** ve **BOQNAME** özniteliklerini sorar.
- Yerel kuyruğu ya da bir diğer ad kuyruğuna ilişkin hedef kuyruğu kapatır.

Yerel bir kuyruk açılırken kullanılan açma seçenekleri ya da bir diğer ad kuyruğuna ilişkin hedef kuyruk, kullanılmakta olan IBM MQ sürümüne bağlıdır:

- IBM MQ 9.1.0 Fix Pack 4 Long Term Support ve öncesi ve IBM MQ 9.1.4 Continuous Delivery ve öncesi için: Yerel kuyruk ya da bir diğer ad kuyruğuna ilişkin hedef kuyruk bir küme kuyruğuysa, XMS .NET kuyruğu `MQOO_INPUT_AS_Q_DEF`, `MQOO_INQUIRE` ve `MQOO_FAIL_IF_QUIESCING` seçenekleriyle açar. Bu, alan uygulamayı çalıştıran kullanıcının küme kuyruğunun yerel eşgörünümüne ilişkin sorma ve erişim yetkisine sahip olması gerektiği anlamına gelir.

  XMS .NET , `MQOO_INQUIRE` ve `MQOO_FAIL_IF_QUIESCING`Gaçma seçenekleriyle diğer tüm yerel kuyruk tiplerini açar. XMS .NET ' in özniteliklerin değerlerini sorgulaması için, alan uygulamayı çalıştıran kullanıcının yerel kuyrukta sorgu erişimi olmalıdır.
- XMS .NET from IBM MQ 9.1.5 ve IBM MQ 9.1.0 Fix Pack 5kullanıldığında, alan uygulamayı çalıştıran kullanıcının, kuyruğun tipine bakılmaksızın, yerel kuyrukta sorgu erişimi olmalıdır.

Zehirli iletileri bir geriletme yeniden kuyruğa alma kuyruğuna ya da kuyruk yöneticisinin gitmeyen iletiler kuyruğuna taşımak için, uygulamayı çalıştıran kullanıcıya `koyma` ve `geçiş` yetkilerini vermeniz gerekir.

ASF ' de zehirli iletilerin işlenmesi
Application Server Facilities (ASF) kullandığınızda, MessageConsumeryerine ConnectionConsumerzehirli iletileri işler. ConnectionConsumer , kuyruğun BackoutThreshold ve BackoutRequeueQName özelliklerine göre iletileri yeniden kuyruğa yollar.

Bir uygulama ConnectionConsumerskomutunu kullandığında, iletinin geriletilmesi, uygulama sunucusunun sağladığı oturuma bağlıdır:

- Oturum işlem yapılmadığında, AUTO_ONAY ya da DUPS_OK_ONAY ile, bir ileti yalnızca bir sistem hatasından sonra ya da uygulama beklenmedik bir şekilde sona ererse geriletir.
- Oturum CLIENT_CEVAP ile işlem yapılmadığında, alınmamış iletiler `Session.recover()` ' i çağıran uygulama sunucusu tarafından yedeklenebilir.

  Genellikle, MessageListener istemci uygulaması ya da uygulama sunucusu `Message.acknowledge()`öğesini çağırır. `Message.acknowledge()` , şu ana kadar oturumda teslim edilen tüm iletileri onaylar.
- Oturum işlemden geçtiğinde, alınmamış iletiler `Session.rollback()` ' i çağıran uygulama sunucusu tarafından yedeklenebilir.

Kuyruk tarayıcıları

Bir uygulama, iletileri kaldırmadan kuyruktaki iletilere göz atmak için bir kuyruk tarayıcısını kullanır.

Bir kuyruk tarayıcısı yaratmak için, bir uygulama, parametre olarak göz atılacak kuyruğu tanıtan bir Hedef nesnesi belirterek, bir ISession nesnesinin Kuyruk Tarayıcısı Yarat yöntemini çağırır. Uygulama, ileti seçici ile ya da ileti seçici olmadan bir kuyruk tarayıcısı yaratabilir.

Bir kuyruk tarayıcısı yarattıktan sonra uygulama, kuyruktaki iletilerin listesini almak için IQueueBrowser nesnesinin GetEnumerator yöntemini çağırabilir. Bu yöntem, bir İleti nesneleri listesini çevreleyen bir sıralı değer listeleyici döndürür. Listedeki İleti nesnelerinin sırası, iletilerin kuyruktan alınacağı sırayla aynıdır. Uygulama daha sonra her iletiye sırayla göz atmak için numaralandırıcıyı kullanabilir.

Kuyruğa ileti konup kuyruktan kaldırıldığında sıralı değer listeleyici dinamik olarak güncellenir. Uygulamanın kuyruktaki sonraki iletiye göz atmak için IEnumerator.MoveNext() komutunu her çağırdığında, ileti kuyruğun yürürlükteki içeriğini yansıtır.

Bir uygulama, belirli bir kuyruk tarayıcısı için GetEnumerator yöntemini bir kereden fazla çağırabilir. Her çağrı yeni bir numaralandırıcı döndürür. Bu nedenle uygulama, kuyruktaki iletilere göz atmak ve kuyruk içinde birden çok konumu korumak için birden çok numaralandırıcıyı kullanabilir.

Bir uygulama, kuyruktan kaldırmak için uygun bir iletiyi aramak üzere bir kuyruk tarayıcısını kullanabilir ve iletiyi kaldırmak için ileti seçicisi olan bir ileti tüketicisini kullanabilir. İleti seçici, iletiyi JMSMessageID üstbilgi alanına göre seçebilir. Bu ve diğer JMS iletisi üstbilgi alanları hakkında bilgi için bkz. “XMS iletisindeki üstbilgi alanları” sayfa 641.

İstekte Bulunanlar

Bir uygulama, istek iletisi göndermek ve sonra yanıtı beklemek ve almak için istekte bulunan bir kişi kullanır.

Birçok ileti alışverişi uygulaması, istek iletisi gönderen ve yanıt bekleyen algoritmalara dayalıdır. XMS , bu uygulama stilinin geliştirilmesine yardımcı olmak için İstekte Bulunan adı verilen bir sınıf sağlar.

Bir istekte bulunan yaratmak için, bir uygulama istekte bulunan sınıfın İstek Oluşturan oluşturucusunu çağırır; bu oluşturucu, bir Oturum nesnesini ve istek iletilerinin gönderileceği yeri tanımlayan bir Hedef nesnesini değiştirgeler olarak belirler. Oturumun işlem yapılmaması ya da XMSC_CLIENT_ONAYI alındı bildirimi kipine sahip olmaması gerekir. Oluşturucu otomatik olarak, yanıt iletilerinin gönderileceği geçici bir kuyruk ya da konu yaratır.

Bir istekçi yarattıktan sonra, uygulama istek iletisi göndermek için İstek nesnesinin İstek yöntemini çağırabilir ve istek iletisini alan uygulamadan bir yanıt bekleyebilir ve alabilir. Çağrı, yanıt alınıncaya kadar ya da oturum sona erinceye kadar (hangisi önce gerçekleşirse) bekler. Her istek iletisi için istekte bulunan kişi için tek bir yanıt gerekli.

Uygulama istekte bulunana kapandığında, geçici kuyruk ya da konu silinir. Ancak, ilişkili oturum kapanmaz.

Nesne silme

Bir uygulama yarattığı bir XMS nesnesini sildiğinde, XMS nesneye ayrılmış iç kaynakları serbest bırakır.

Bir uygulama bir XMS nesnesi oluşturduğunda, XMS nesne için bellek ve diğer iç kaynakları ayırır. XMS , uygulama nesnenin kapatma ya da silme yöntemini çağırarak nesneyi belirtik olarak silinceye kadar bu iç kaynakları korur; bu noktada XMS , iç kaynakları serbest bırakır. Bir uygulama önceden silinmiş bir nesneyi silmeye çalışırsa, çağrı yoksayılır.

Bir uygulama bir Bağlantı ya da Oturum nesnesini sildiğinde, XMS ilişkili bazı nesneleri otomatik olarak siler ve iç kaynaklarını serbest bırakır. Bunlar, Connection ya da Session nesnesi tarafından yaratılan ve nesneden bağımsız işlevi olmayan nesnelerdir. Bu nesneler Çizelge 83 sayfa 621içinde gösterilir.

Not: Bir uygulama bağımlı oturumları olan bir bağlantıyı kapatırsa, bu oturumlara bağımlı tüm nesneler de silinir. Yalnızca bir Bağlantı ya da Oturum nesnesinin bağımlı nesneleri olabilir.

Çizelge 83. Otomatik olarak silinen nesneler

Silinen nesne	**Yöntem**	**Otomatik olarak silinen bağımlı nesneler**
Bağlantı	Bağlantıyı Kapat	ConnectionMetaVeri ve Oturum nesneleri
Oturum	Oturumu Kapat	MessageConsumer, MessageProducer, QueueBrowserve Requestor nesneleri

XMS aracılığıyla yönetilen IBM MQ XA işlemleri

Yönetilen IBM MQ XA işlemleri XMSaracılığıyla kullanılabilir.

XMSaracılığıyla XA hareketlerini kullanmak için, işlemli bir oturum yaratılmalı. XA hareketi kullanımdayken, hareket denetimi Dağıtılmış Hareket Koordinatörü (DTC) genel hareketler aracılığıyla olur ve XMS oturumları aracılığıyla olmaz. XA hareketleri kullanılırken, XMS oturumunda `Session.commit` ya da `Session.rollback` yayınlanamaz. Bunun yerine, `Transscope.Commit` ya da `Transscope.Rollback` DTC yöntemlerini kullanarak hareketleri kesinleştirin ya da geri döndürün. XA hareketi için bir oturum kullanılıyorsa, oturum kullanılarak yaratılan üretici ya da tüketici XA hareketinin bir parçası olmalıdır. Bunlar, XA hareket kapsamı dışındaki herhangi bir işlem için kullanılamaz. Bunlar, XA hareketi dışında `Producer.send` ya da `Consumer.receive` gibi işlemler için kullanılamaz.

Aşağıdakiler durumunda bir `IllegalStateException` kural dışı durum nesnesi yayınlanır:

- `Session.commit` ya da `Session.rollback`için XA işlemli oturum kullanılır.

- XA işlemli oturumda bir kez kullanılan üretici ya da tüketici nesneleri, XA hareket kapsamı dışında kullanılır.

XA hareketleri zamanuyumsuz tüketiciler tarafından desteklenmez.

Not:

1. XA hareketi kesinleştirmesinden önce `Producer`, `Consumer`, `Session`ya da `Connection` nesnesinde bir kapatma komutu verilebilir. Bu durumlarda, hareketteki iletiler geriye işlenir. Benzer şekilde, bağlantı XA hareketi kesinleştirilmeden önce kesilirse, hareketteki tüm iletiler geriye işlenir. `Producer` nesnesi için geriye işleme, iletilerin kuyruğa konmadığı anlamına gelir. Bir `Consumer` nesnesi için geriye işleme, iletilerin kuyrukta kaldığı anlamına gelir.
2. `Producer` nesnesi `TransactionScope` içinde `TimeToLive` olan bir ileti yerleştirirse ve süre dolduktan sonra bir `commit` yayınlandıysa, iletinin süresi `commit` yayınlanmadan önce sona erebilir. Bu durumda, ileti `Consumer` nesnelerinin kullanımına sunulmaz.
3. `Session` nesneleri iş parçacıklarında desteklenmez. İş parçacıkları arasında paylaşılan `Session` nesneleriyle hareket kullanımı desteklenmez.

XMS ilkel tipleri

XMS , sekiz Java temel tipinin (byte, short, int, long, float, double, char ve boolean) eşdeğerlerini sağlar. Bu, verilerin kaybolmasına ya da bozulmasına gerek kalmadan XMS ile JMS arasında ileti değiş tokuna izin verir.

Çizelge 84 sayfa 622 içinde, her bir XMS temel tipi için Java eşdeğer veri tipi, boyutu ve değer alt sınırı ve üst sınırı listelenir.

Çizelge 84. XMS veri tipleri ve Java eşdeğerleri

XMS veri tipi	UyumluJava veri tipi	Büyüklük	Minimum değer	Maksimum değer
System.Boolean	boole	32 bit	yanlış	doğru
System.SBYTE	Byte	8 bit	-2^7 (-128)	2^7-1 (127)
System.BYTE	Byte	8 bit	-2^7 (-128)	2^7-1 (127)
System.CHAR	Byte	8 bit	-2^7 (-128)	2^7-1 (127)
System.Int16	kısa	16 bit	-2^{15} (-32768)	2^{15}-1 (32767)
System.Int32	int	32 bit	-2^{31} (-2147483648)	2^{31}-1 (2147483647)
System.Int64	uzun	64 bit	-2^{63} (-9223372036854775808)	2^{63}-1 (9223372036854775807)
System.Single	kayan nokta	32 bit	-3.402823E-38 (7 basamaklı duyarlık)	3.402823E+38 (7 basamaklı duyarlık)
System.Double	çift	64 bit	-1.79769313486231E-308 (15 basamaklı duyarlık)	1.79769313486231E+308 (15 basamaklı duyarlık)

Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi

Bir uygulama bir özelliğin değerini aldığında, değer XMS tarafından başka bir veri tipine dönüştürülebilir. Hangi dönüşümlerin desteklendiğini ve XMS ' in dönüşümleri nasıl gerçekleştirdiğini birçok kural yönetir.

Bir nesnenin özelliğinin bir adı ve değeri vardır; değerin, bir özelliğin değerine *özellik tipi*olarak da atıfta bulunulduğu ilişkili bir veri tipi vardır.

Bir uygulama, nesnelerin özelliklerini almak ve ayarlamak için PropertyContext sınıfının yöntemlerini kullanır. Bir özelliğin değerini almak için uygulama, özellik tipi için uygun olan yöntemi çağırır. Örneğin, bir tamsayı özelliğinin değerini almak için, bir uygulama genellikle GetIntÖzellik yöntemini çağırır.

Ancak, bir uygulama bir özelliğin değerini aldığında, değer XMS tarafından başka bir veri tipine dönüştürülebilir. Örneğin, bir tamsayı özelliğinin değerini almak için bir uygulama, özelliğin değerini dizgi olarak döndüren GetStringProperty yöntemini çağırabilir. XMS tarafından desteklenen dönüştürmeler Çizelge 85 sayfa 623içinde gösterilir.

Çizelge 85. Bir özellik tipinden diğer veri tiplerine desteklenen dönüştürmeler

Özellik tipi	**Desteklenen hedef veri tipleri**
System.String	System.Boolean, System.Double, System.Float, System.Int32, System.Int64, System.SByte, System.Int16
System.Boolean	System.String, System.SByte, System.Int32, System.Int64, System.Int16
System.Char	System.String
System.Double	System.String
System.Float	System.String, System.Double
System.Int32	System.String, System.Int64
System.Int64	System.String
System.SByte	System.String, System.Int32, System.Int64, System.Int16
System.SByte dizisi	System.String
System.Int16	System.String, System.Int32, System.Int64

Aşağıdaki genel kurallar, desteklenen dönüştürmeleri yönetir:

- Dönüştürme sırasında veri kaybı olmaması koşuluyla, sayısal özellik değerleri bir veri tipinden diğerine dönüştürülebilir. Örneğin, System.Int32 veri tipindeki bir özelliğin değeri System.Int64veri tipindeki bir değere dönüştürülebilir, ancak System.Int16veri tipindeki bir değere dönüştürülemez.
- Herhangi bir veri tipindeki bir özellik değeri dizgiye dönüştürülebilir.
- Dizginin dönüştürme için doğru biçimlenmesi koşuluyla, dizgi özelliği değeri başka bir veri tipine dönüştürülebilir. Bir uygulama doğru biçimlenmemiş bir dizgi özelliği değerini dönüştürmeyi denerse, XMS hatalar döndürebilir.
- Bir uygulama desteklenmeyen bir dönüştürme girişiminde bulunursa, XMS bir hata döndürebilir.

Bir özellik değeri bir veri tipinden diğerine dönüştürüldüğünde aşağıdaki kurallar geçerlidir:

- Bir Boole özellik değeri dizgiye dönüştürülürken, true değeri "true"dizgisine ve false değeri "false"dizgisine dönüştürülür.
- Bir Boole özellik değeri System.SBytede içinde olmak üzere sayısal bir veri tipine dönüştürülürken, true değeri 1 'e, false değeri 0 'a dönüştürülür.
- Bir dizgi özelliği değeri Boole değerine dönüştürülürken, "true" (büyük ve küçük harfe duyarlı değil) ya da "1" dizgisi true (doğru) değerine ve "false" (büyük ve küçük harfe duyarlı değil) ya da "0" dizgisi false (yanlış) değerine dönüştürülür. Diğer tüm dizgiler dönüştürülemiyor.
- Bir dizilim özelliği değerini System.Int32, System.Int64, System.SByteya da System.Int16veri tipli bir değere dönüştürürken, dizilim aşağıdaki biçimde olmalıdır:

  [*boşluklar*] [*işaret*]*rakamlar*

  Dizgi bileşenleri aşağıdaki gibi tanımlanır:

  boşluklar
  İsteğe bağlı baştaki boş karakterler.

  İşaret
  İsteğe bağlı bir artı işareti (+) ya da eksi işareti (-) karakteri.

  Rakamlar
  Bitişik sayı karakterleri sırası (0-9). En az bir basamak karakteri bulunmalıdır.

Sayı karakterleri dizgisinden sonra, dizilim rakam karakterleri olmayan diğer karakterleri içerebilir, ancak bu karakterlerin ilkine ulaşılır ulaşılmaz dönüştürme durur. Dizginin bir ondalık tamsayıyı temsil ettiği varsayılır.

Dizgi doğru biçimlendirilmediyse, XMS hata döndürebilir.

- Bir dizilim özelliği değerini System.Double ya da System.Floatveri tipindeki bir değere dönüştürürken, dizginin biçimi aşağıdaki olmalıdır:

  [*boşluklar*] [*işaret*] [*rakamlar*] [*nokta*[*d_rakamları*]] [*e_char*[*e_imi*]*e_rakamları*]

  Dizgi bileşenleri aşağıdaki gibi tanımlanır:

  boşluklar
  (İsteğe bağlı) Önden gelen boş karakterler.

  İşaret
  (İsteğe bağlı) Artı işareti (+) ya da eksi işareti (-) karakteri.

  Rakamlar
  Bitişik sayı karakterleri sırası (0-9). *basamaklar* ya da *d_basamaklar*içinde en az bir basamak karakteri bulunmalıdır.

  nokta
  (İsteğe bağlı) Ondalık nokta (.).

  d_basamakları
  Bitişik sayı karakterleri sırası (0-9). *basamaklar* ya da *d_basamaklar*içinde en az bir basamak karakteri bulunmalıdır.

  e_char
  E ya da *e*olan üstel bir karakter.

  e_işareti
  (İsteğe bağlı) Üstel için artı işareti (+) ya da eksi işareti (-) karakteri.

  e_basamaklar
  Üs için bitişik sayı karakterleri sırası (0-9). Dizgi bir üstel karakter içeriyorsa, en az bir basamak karakteri bulunmalıdır.

  Sayı karakterleri sırasından ya da bir üstel karakteri gösteren isteğe bağlı karakterlerden sonra, dizgi rakam karakteri olmayan diğer karakterleri içerebilir, ancak bu karakterlerin ilkine ulaşıldığında dönüştürme durur. Dizginin, 10 üslü bir ondalık kayan nokta sayısını temsil ettiği varsayılır.

  Dizgi doğru biçimlendirilmediyse, XMS hata döndürebilir.

- Sayısal bir özellik değeri, System.SByteveri tipindeki bir özellik değeri de içinde olmak üzere bir dizgiye dönüştürülürken, değer, o değere ilişkin ASCII karakterini içeren dizgiye değil, değerin ondalık sayı olarak gösterilmesine dönüştürülür. Örneğin, 65 tamsayısı "A"dizgisine değil, "65"dizgisine dönüştürülür.
- Bir bayt dizisi özellik değeri dizgiye dönüştürülürken, her bayt, baytı gösteren 2 onaltılı karaktere dönüştürülür. Örneğin, {0xF1, 0x12, 0x00, 0xFF} bayt dizisi "F11200FF"dizgisine dönüştürülür.

Bir özellik tipinden diğer veri tiplerine dönüştürmeler, hem Özellik hem de PropertyContext sınıflarının yöntemleri tarafından desteklenir.

Yineleyiciler

Yineleyici, listede yürürlükteki konumu koruyan bir nesne ve imleç listesini içerir. IBM MQ Message Service Client (XMS) for C/C++içinde olduğu gibi bir Yineleyici kavramı, IBM MQ Message Service Client (XMS) for .NETiçinde IEnumerator arabirimi kullanılarak uygulanır.

Bir yineleyici yaratıldığında, imlecin konumu ilk nesneden önce gelir. Bir uygulama, sırayla her bir nesneyi almak için bir yineleyici kullanır.

IBM MQ Message Service Client (XMS) for C/C++ iterator sınıfı, Javaiçindeki Enumerator sınıfına eşdeğerdir. IBM MQ Message Service Client (XMS) for .NET , Java ile benzerdir ve bir IEnumerator arabirimi kullanır.

Bir uygulama, aşağıdaki görevleri gerçekleştirmek için bir IEnumerator kullanabilir:

- Bir iletinin özelliklerini almak için
- Bir eşlem iletisinin gövdesindeki ad-değer çiftlerini almak için
- Kuyruktaki iletilere göz atmak için
- Bir bağlantı tarafından desteklenen JMS tanımlı ileti özelliklerinin adlarını almak için

Kodlanmış karakter takımı tanıtıcıları

XMS .NETiçinde, tüm dizgiler yerel .NET dizgisi kullanılarak geçirilir. Bu sabit bir kodlamaya sahip olduğundan, yorumlamak için daha fazla bilgi gerekmez. Bu nedenle, XMS .NET uygulamaları için XMSC_CLIENT_CCSID özelliği gerekli değildir.

XMS hata ve kural dışı durum kodları

XMS , hataları belirtmek için bir hata kodları aralığı kullanır. Bu hata kodları, yayından yayına değişebildiği için bu belgede açık bir şekilde listelenmez. Yalnızca XMS kural dışı durum kodları (XMS_X_ ... biçiminde), XMSyayınlarında aynı kaldıkları için belgelenir.

XMS aracılığıyla otomatik IBM MQ Client yeniden bağlantısı

XMS istemcinizi, ileti sağlayıcısı olarak IBM WebSphere MQ 7.1 istemcisini ve üstünü kullanırken bir ağ, kuyruk yöneticisi ya da sunucu hatasının ardından otomatik olarak yeniden bağlanacak şekilde yapılandırın.

Otomatik olarak yeniden bağlanmak üzere bir istemci bağlantısı yapılandırmak için `MQConnectionFactory` sınıfının `WMQ_CONNECTION_NAME_LIST` ve `WMQ_CLIENT_RECONNECT_OPTIONS` özelliklerini kullanın. Otomatik istemci yeniden bağlantısı, bir bağlantı başarısız olduktan sonra ya da kuyruk yöneticisini durdurduktan sonra bir seçenek olarak yeniden bağlanır. Bazı istemci uygulamalarının tasarımı, bunların otomatik yeniden bağlantı için uygun olmamasına neden olur.

Bağlantı kurulduktan sonra, otomatik olarak yeniden bağlanabilir istemci bağlantıları yeniden bağlanabilir hale gelir.

Not: `İstemci Yeniden Bağlanma Seçenekleri`, `İstemci Yeniden Bağlanma Zamanaşımıve Bağlantı Adlandırma` özellikleri, İstemci Kanal Tanımları Çizelgesi (CCDT) aracılığıyla ya da `mqclient.ini` dosyası aracılığıyla istemci yeniden bağlantısının etkinleştirilmesiyle de ayarlanabilir.

Not: `ConnectionFactory` nesnesinde ve CCDT ' de yeniden bağlantı özellikleri ayarlanırsa, öncelik kuralı aşağıdaki gibidir. `ConnectionFactory` nesnesinde bağlantı adı listesi özelliğinin varsayılan değeri ayarlanırsa, CCDT öncelikli olur. Bağlantı adı listesi varsayılan değerine ayarlanmazsa, `ConnectionFactory` nesnesinde ayarlanan özellik değerleri öncelikli olur. Bağlantı ad değerinin varsayılan değeri `localhost(1414)` değeridir.

XMS .NET uygulamaları yazılıyor

Bu bölümdeki konular, XMS .NET uygulamaları yazarken size yardımcı olacak bilgiler sağlar.

Bu görev hakkında

Bu bölümde, XMS .NET uygulamalarının yazılmasına özgü bilgiler sağlanır. XMS uygulamalarının yazılmasına ilişkin genel bilgi için bkz. “XMS uygulamaları yazılıyor” sayfa 608.

Bu bölüm aşağıdaki konuları içerir:

- ".NET içinde var olmayan özellik işleme" sayfa 628
- ".NET içinde hata işleme" sayfa 629
- ".NET içinde ileti ve kural dışı durum dinleyicilerinin kullanılması" sayfa 629

.NET için veri tipleri

XMS .NET , System.Boolean, System.Byte, System.SByte, System.Char, System.String, System.Single, System.Double, System.Decimal, System.Int16, System.Int32, System.Int64, System.UInt16, System.UInt32, System.UInt64, ve System.Object. XMS .NET veri tipleri, XMS C/C + + veri tiplerinden farklıdır. Bu konuyu, ilgili veri tiplerini tanımlamak için kullanabilirsiniz.

Aşağıdaki çizelge, ilgili XMS .NET ve XMS C/C++ veri tiplerini gösterir ve bunları kısaca açıklar.

Çizelge 86. XMS .NET ve XMS C/C++ için veri tipleri

XMS .NET tipi	**XMS C/C++ tipi**	**Açıklama**
System.SByte	xmsSBYTE xmsINT8	İmzalı 8 bitlik değer
System.Byte	xmsBYTE xmsUINT8	İmzasız 8 bitlik değer
System.Int16	xmsINT16 xmsSHORT	İmzalı 16 bitlik değer
System.UInt16	xmsUINT16 xmsUSHORT	İmzasız 16 bitlik değer
System.Int32	xmsINT32 xmsINT	İmzalı 32 bitlik değer
System.UInt32	xmsUINT32 xmsUINT	İmzasız 32 bitlik değer
System.Int64	xmsLONG xmsINT64	İmzalı 64 bitlik değer
System.UInt64	xmsULONG xmsUINT64	İmzasız 64 bitlik değer
System.Char	xmsCHAR16	İmzasız 16 bitlik karakter (.NET için Unicode)
System.Single	xmsFLOAT	IEEE 32 bit kayar noktalı sayı
System.Double	xmsDOUBLE	IEEE 64 bitlik kayar noktalı sayı
System.Boolean	xmsBOOL	Doğru/Yanlış değeri
Geçerli değildir	xmsCHAR	İmzalı ya da İmzasız 8 bitlik değer (imzalanmış ya da imzalanmamış, platforma bağlıdır)
System.Decimal	Geçerli değildir	96 bit işaretli tamsayı çarpı 10^{0} -10^{28}
System.Object	Geçerli değildir	Tüm tiplerin temeli

Çizelge 86. XMS .NET ve XMS C/C++ için veri tipleri (devamı var)		
XMS .NET tipi	**XMS C/C++ tipi**	**Açıklama**
System.String	Geçerli değildir	Dizgi Tipi

.NET içinde yönetilen ve yönetilmeyen işlemler

Yönetilen kod yalnızca .NET ortak dil çalıştırma zamanı ortamında yürütülür ve tamamen o çalıştırma zamanı tarafından sağlanan hizmetlere bağlıdır. Uygulamanın herhangi bir parçası hizmetleri .NET ortak dil yürütme ortamı dışında çalıştırırsa ya da çağırırsa, uygulama yönetilmeyen olarak sınıflandırılır.

Bazı gelişmiş işlevler şu anda yönetilen .NET ortamında desteklenmemektedir.

Uygulamanız, şu anda tam olarak yönetilen ortamda desteklenmeyen bazı işlevler gerektiriyorsa, uygulamanızda önemli bir değişiklik gerektirmeden, uygulamanızı yönetilmeyen ortamı kullanacak şekilde değiştirebilirsiniz. Ancak, bu seçim yapıldığında XMS yığınının yönetilmeyen koddan yararlandığını unutmayın.

IBM MQ kuyruk yöneticisine bağlantılar

WMQ_CM_CLIENT ile yapılan yönetilen bağlantılar, TCP dışı iletişimi ve kanal sıkıştırmayı desteklemez. Ancak, bu bağlantılar yönetilmeyen bir bağlantı (WMQ_CM_CLIENT_UNMANAGED) kullanılarak desteklenebilir. Daha fazla bilgi için bkz “.NET uygulamalarının geliştirilmesi” sayfa 533.

Yönetilmeyen bir ortamda denetlenen bir nesneden bağlantı üreticisi yaratırsanız, bağlantı kipinin değerini el ile XMSC_WMQ_CM_CLIENT_UNMANAGED olarak değiştirmeniz gerekir.

Bir WebSphere Application Server hizmet bütünleştirmesi veriyolu ileti alışverişi altyapısına yönelik bağlantılar

SSL iletişim kuralının (HTTPSdahil) kullanılmasını gerektiren bir WebSphere Application Server hizmet bütünleştirmesi veriyolu ileti alışverişi altyapısına yönelik bağlantılar şu anda yönetilen kod olarak desteklenmez.

.NET içindeki hedefler

.NETiçinde, hedefler protokol tipine göre oluşturulur ve yalnızca oluşturuldukları protokol tipinde kullanılabilir.

Hedef yaratmak için iki işlev sağlanır: Biri konular için, diğeri kuyruklar için:

- `IDestination CreateTopic(String topic);`
- `IDestination CreateQueue(String queue);`

Bu işlevler, API 'de aşağıdaki iki nesne üzerinde kullanılabilir:

- ISession (Oturum)
- XMSFactoryFactory

Her iki durumda da bu yöntemler, aşağıdaki biçimde parametreleri içerebilen bir URI stili dizgisini kabul edebilir:

```
"topic://some/topic/name?priority=5"
```

Diğer bir seçenek olarak, bu yöntemler yalnızca hedef adı kabul edebilir; yani, konu: // ya da kuyruk: // öneki olmayan ve parametre içermeyen bir ad.

Bu, aşağıdaki URI stili dizgisinin olduğu anlamına gelir:

```
CreateTopic("topic://some/topic/name");
```

şu hedef adla aynı sonucu verir:

```
CreateTopic("some/topic/name");
```

WebSphere Application Server Service Integration Bus JMSile ilgili olarak, konular hem *topicname* , hem de *topicspace* içeren ancak parametre içeremeyen kısa bir biçimde de belirtilebilir:

```
CreateTopic("topicspace:topicname");
```

.NET içindeki özellikler

.NET uygulaması, nesnelerin özelliklerini almak ve ayarlamak için PropertyContext arabirimindeki yöntemleri kullanır.

PropertyContext arabirimi, özellikleri alan ve ayarlanan yöntemleri içerir. Bu yöntemler, aşağıdaki sınıflar tarafından doğrudan ya da dolaylı olarak edinilir:

- BytesMessage
- Bağlantı
- ConnectionFactory
- ConnectionMetaVerileri
- Hedef
- MapMessage
- İleti
- MessageConsumer
- MessageProducer
- ObjectMessage
- QueueBrowser
- Oturum
- StreamMessage
- TextMessage

Bir uygulama bir özelliğin değerini ayarlarsa, yeni değer, özelliğin sahip olduğu önceki değerlerin yerine geçer.

XMS özellikleriyle ilgili ek bilgi için XMS nesnelerinin özellikleribaşlıklı konuya bakın.

Kullanım kolaylığı için, XMS içindeki XMS özellik adları ve değerleri, XMSC adlı bir yapıda genel sabitler olarak önceden tanımlanmıştır. Bu sabitlerin adları XMSC.*constant*biçimindedir; örneğin, XMSC.USERID (bir özellik adı değişmezi) ve XMSC.DELIVERY_AS_APP (bir değer sabiti).

Buna ek olarak, IBM MQ değişmezlerine IBM.XMS.MQC yapısını kullanarak da erişebilirsiniz. IBM.XMS ad alanı zaten içe aktarıldı, bu özelliklerin değerlerine MQC.*constant*biçiminde erişebilirsiniz. Örneğin, MQC.MQRO_COA_WITH_FULL_DATA.

Ayrıca, .NET için hem XMS .NET hem de IBM MQ sınıflarını kullanan ve hem IBM.XMS ve IBM.WMQ ad alanlarını, her oluşum için benzersiz olduğundan emin olmak için MQC yapı ad alanını tam olarak nitelemeniz gerekir.

Bazı gelişmiş işlevler şu anda yönetilen .NET ortamında desteklenmiyor. Daha fazla ayrıntı için bkz. “.NET içinde yönetilen ve yönetilmeyen işlemler” sayfa 627 .

.NET içinde var olmayan özellik işleme

XMS .NET içinde var olmayan özelliklerin işlenmesi, JMS belirtimiyle ve XMSC ve C++ uygulamalarıyla da genel olarak tutarlıdır.

JMS ' de var olmayan bir özelliğe erişim, bir yöntem var olmayan (null) değeri gerekli tipe dönüştürmeyi denediğinde Java sistem kural dışı durumuyla sonuçlanabilir. Bir özellik yoksa aşağıdaki kural dışı durumlar oluşur:

- getStringÖzellik ve getObjectÖzellik boş değer döndürdü
- getBooleanözelliği, Boolean.valueOf(null) false değerini döndürdüğünden false değerini döndürür
- Integer.valueOf(null) kural dışı durum verdiğinden getIntProperty etc.throw java.lang.NumberFormatException

XMS .NET içinde bir özellik yoksa aşağıdaki kural dışı durumlar oluşur:

- GetStringÖzelliği ve GetObjectÖzelliği (ve GetBytesÖzelliği) boş değer (Javaile aynıdır) döndürür
- GetBooleanözelliği System.NullReferenceException verir.
- GetIntÖzellik, System.NullReferenceException verir.

Bu uygulama Java' dan farklıdır, ancak JMS belirtimiyle ve XMS C ve C++ arabirimleriyle büyük ölçüde tutarlıdır. Java somutlaması gibi, XMS .NET da çağıran için System.Convert çağrısından gelen kural dışı durumları yayayacaktır. Java ' in tersine, XMS belirtik olarak NullReferencekural dışı durumları yayınlıyor; bunun yerine, .NET çerçevesinin yerel davranışını boş değerli sistem dönüştürme yordamlarını geçirerek kullanıyor. Uygulamanız bir özelliği "abc" gibi bir Dizgiye ayarlarsa ve GetIntözelliğini çağırırsa, Convert.ToInt32("abc") tarafından yayınlanan System.FormatException Javaile tutarlı olan çağırana yayılır. MessageFormatYalnızca setProperty ve getProperty için kullanılan tipler uyumsuzsa kural dışı durum yayınlanır. Bu davranış Javaile de tutarlıdır.

.NET içinde hata işleme

XMS .NET kural dışı durumların tümü System.Exception' dan türetilir. XMS yöntem çağrıları, MessageFormatException, genel XMSExceptions ya da NullReferenceException gibi sistem kural dışı durumları gibi belirli XMS kural dışı durumlarını yayınlayabilir.

Belirli yakalama öbeklerinde ya da genel olarak `System.Exception` yakalama öbeklerinde, uygulama gereksinimlerine uygun şekilde, bu hatalardan herhangi birini yakalamak için yazma uygulamaları.

.NET içinde ileti ve kural dışı durum dinleyicilerinin kullanılması

.NET uygulaması, iletileri zamanuyumsuz olarak almak için bir ileti dinleyicisini kullanır ve bir bağlantıyla ilgili bir sorunla ilgili zamanuyumsuz olarak bildirim almak için bir kural dışı durum dinleyicisini kullanır.

Bu görev hakkında

İleti ve kural dışı durum dinleyicilerinin işlevleri .NET ve C + + için aynıdır. Ancak, bazı küçük uygulama farklılıkları vardır.

Yordam

- İletileri zamanuyumsuz olarak almak üzere bir ileti dinleyici ayarlamak için aşağıdaki adımları tamamlayın:
  a) İleti dinleyici temsilcisinin imzasıyla eşleşen bir yöntem tanımlayın.

  Tanımladığınız yöntem statik ya da somut örnek yöntemi olabilir ve herhangi bir erişilebilir sınıfta tanımlanabilir. Temsilci imzası aşağıdaki gibidir:

  ```
  public delegate void MessageListener(IMessage msg);
  ```

  ve yöntemi şöyle tanımlayabilmeniz için:

  ```
  void SomeMethodName(IMessage msg);
  ```

  b) Aşağıdaki örneğe benzer bir şey kullanarak bu yöntemi temsilci olarak somutlaştırın:

```
MessageListener OnMsgMethod = new MessageListener(SomeMethodName)
```

c) Temsilciyi, tüketicinin MessageListener özelliğine ayarlayarak bir ya da daha fazla tüketiciye kaydedin:

```
consumer.MessageListener = OnMsgMethod;
```

MessageListener öğesini boş değere ayarlayarak temsilciyi kaldırabilirsiniz:

```
consumer.MessageListener = null;
```

- Bir kural dışı durum dinleyicisi oluşturmak için aşağıdaki adımları tamamlayın.

  Kural dışı durum dinleyicisi, ileti dinleyiciyle aynı şekilde çalışır, ancak farklı bir temsilci tanımlaması vardır ve ileti tüketicisi yerine bağlantıya atanır. Bu, C + + ile aynıdır.

  a) Yöntemi tanımlayın.

  Temsilci imzası aşağıdaki gibidir:

```
public delegate void ExceptionListener(Exception ex);
```

ve bu nedenle, tanımlanan yöntem şöyle olabilir:

```
void SomeMethodName(Exception ex);
```

b) Aşağıdaki örneğe benzer bir şey kullanarak bu yöntemi temsilci olarak somutlaştırın:

```
ExceptionListener OnExMethod = new ExceptionListener(SomeMethodName)
```

c) ExceptionListener özelliğini ayarlayarak temsilciyi bağlantıya kaydedin:

```
connection.ExceptionListener = OnExMethod ;
```

ExceptionListener ayarını aşağıdaki şekilde ayarlayarak temsilciyi kaldırabilirsiniz:

```
null:  connection.ExceptionListener = null;
```

XMS .NET tarafından denetlenen nesnelerle çalışma

Bu bölümdeki konular, yönetilen nesnelerle ilgili bilgi sağlar. XMS uygulamaları, merkezi yönetilen bir nesne havuzundan nesne tanımlamalarını alabilir ve bunları bağlantı üreticileri ve hedefler yaratmak için kullanabilir.

Bu görev hakkında

Bu bölümde, XMS ' in desteklediği yönetilen nesne havuzu tiplerini açıklayarak, yönetilen nesnelerin oluşturulmasına ve yönetilmesine yardımcı olacak bilgiler sağlanır. Bu bölümde, XMS uygulamasının gerekli yönetilen nesneleri almak için yönetilen nesneler havuzuna nasıl bağlantı yaptığı da açıklanmaktadır.

Bu bölüm aşağıdaki konuları içerir:

- "XMS .NET desteklenen yönetilen nesne havuzu tipleri" sayfa 631
- "Yönetilen nesneler için XMS .NET özellik eşlemesi" sayfa 631
- "Yönetilen ConnectionFactory nesneleri için XMS .NET gerekli özellikler" sayfa 633
- "XMS .NET denetlenen Hedef nesneler için gerekli özellikler" sayfa 634

- “XMS .NET yönetilen nesneler oluşturma” sayfa 635
- “XMS .NET InitialContext nesneleri oluşturma” sayfa 635
- “XMS .NET InitialContext özellikleri” sayfa 636
- “XMS başlangıç bağlamları için URI biçimi” sayfa 636
- “XMS .NET için JNDI Arama Web hizmeti” sayfa 637
- “XMS .NET yönetilen nesnelerin alınması” sayfa 637

XMS .NET desteklenen yönetilen nesne havuzu tipleri

Dosya Sistemi ve LDAP tarafından yönetilen nesneler IBM MQ ve WebSphere Application Server'e bağlanmak için kullanılabilir, COS Adlandırma ise yalnızca WebSphere Application Server' e bağlanmak için kullanılabilir.

Dosya Sistemi nesne dizinleri, diziselleştirilmiş Java Naming Directory Interface (JNDI) nesnelerinin biçimini alır. LDAP nesne dizinleri, JNDI nesnelerini içeren dizinlerdir. Dosya Sistemi ve LDAP nesne dizinleri, IBM MQ ve sonraki sürümleriyle birlikte sağlanan IBM MQ Explorertarafından yönetilebilir. IBM MQ bağlantı üreticilerini ve hedeflerini merkezileştirerek istemci bağlantılarını yönetmek için hem Dosya sistemi hem de LDAP nesne dizinleri kullanılabilir. Ağ yöneticisi, aynı merkezi havuza başvuran ve merkezi havuzda yapılan bağlantı ayarlarında yapılan değişiklikleri yansıtacak şekilde otomatik olarak güncellenen birden çok uygulamayı dağıtabilir.

Bir COS adlandırma dizini, WebSphere Application Server service integration bus bağlantı üreticilerini ve hedeflerini içerir ve WebSphere Application Server denetim konsolu kullanılarak yönetilebilir. Bir XMS uygulamasının COS adlandırma dizininden nesneleri alabilmesi için bir JNDI arama web hizmeti konuşlandırılmalıdır. Bu web hizmeti tüm WebSphere Application Server service integration technologiesüzerinde kullanılamaz. Ayrıntılar için ürün belgelerine bakın.

Not: Nesne dizininde yapılan değişikliklerin yürürlüğe girmesi için uygulama bağlantılarını yeniden başlatın.

Yönetilen nesneler için XMS .NET özellik eşlemesi

XMS .NET uygulamalarının IBM MQ JMS ve WebSphere Application Server bağlantı üreticisi ve hedef nesne tanımlamalarını kullanmasını sağlamak için, bu tanımlamalardan alınan özelliklerin, XMS bağlantı üreticileri ve hedeflerinde ayarlanabilen ilgili XMS özellikleriyle eşlenmesi gerekir.

Örneğin, bir IBM MQ JMS bağlantı üreticisinden alınan özelliklere sahip bir XMS bağlantı üreticisi yaratmak için, özelliklerin ikisi arasında eşlenmesi gerekir.

Tüm özellik eşlemeleri otomatik olarak gerçekleştirilir.

Çizelge 87 sayfa 631 , bağlantı üreticilerinin ve hedeflerin en yaygın özelliklerinden bazıları arasındaki eşlemeleri gösterir. Bu çizelgede gösterilen özellikler yalnızca küçük bir örnek kümeleridir ve gösterilen özelliklerin tümü tüm bağlantı tipleri ve sunucularla ilgili değildir.

Çizelge 87. Bağlantı üreticisi ve hedef özellikleri için ad eşleme örnekleri

IBM MQ JMS özellik adı	**XMS özellik adı**	**WebSphere Application Server service integration bus özellik adı**
KALıCıLıK (BAşıNA)	XMSC_DELIVERY_MODE	
EXPIRY (EXP)	XMSC_TIME_TO_LIVE	
ÖNCELIK (PRI)	XMSC_PRIORITY	
	XMSC_WPM_ANASISTEM_ADI	serverName
	XMSC_WPM_BUS_NAME	busName
	XMSC_WPM_TOPIC_SPACE	topicName

Not: Çizelge 88 sayfa 632 içinde gösterilen özellikler, JMS ve XMS .NETiçin geçerlidir.

Çizelge 88. XMS .NET Özellikler

Özellik	**Nesne tipi**				
	CF	**QCF**	**TCF.**	**Kuyruk**	**Konu**
UYGULAMAADI	Y	Y	Y	Yok	Yok
ZAMANUYUMSUZ KURAL Dışı DURUM	Y	Y	Y	Yok	Yok
CCDTURL	Y	Y	Y	Yok	Yok
Kanal	Y	Y	Y	Yok	Yok
BAğLANTı KURAMCıSı	Y	Y	Y	Yok	Yok
CLIENTRECONNECTOPTIONS	Y	Y	Y	Yok	Yok
CLIENTRECONNECTTIMEOUT	Y	Y	Y	Yok	Yok
İSTEMCIID	Yok	Y	Yok	Yok	Yok
COMPHDR "1" sayfa 633	Y	Yok	Y	Yok	Yok
ŞIRKET "1" sayfa 633	Y	Y	Y	Yok	Yok
CONNOPT "1" sayfa 633	Y	Y	Y	Yok	Yok
CONNTAG "1" sayfa 633	Y	Y	Y	Yok	Yok
Açıklama"1" sayfa 633	Yok	Y	Yok	Y	Y
EXPIRY "1" sayfa 633	Yok	Yok	Yok	Y	Y
FAILIFQUIESCE	Y	Y	Y	Y	Y
ANASİSTEM ADI	Yok	Y	Yok	Yok	Yok
YERELADRES	Yok	Y	Yok	Yok	Yok
Kalıcılık	Yok	Yok	Yok	Y	Y
PORT	Yok	Y	Yok	Yok	Yok
ÖNCELIK "1" sayfa 633	Yok	Yok	Yok	Y	Y
PROVIDERVERSION "1" sayfa 633	Yok	Y	Yok	Yok	Yok
YöNETICI	Y	Y	Y	Y	Yok

Çizelge 88. XMS .NET Özellikler (devamı var)					
Özellik	**Nesne tipi**				
	CF	**QCF**	**TCF.**	**Kuyruk**	**Konu**
KUYRUK "1" sayfa 633	Yok	Yok	Yok	Y	Yok
FARKLıLıKLı	Y	Y	Y	Yok	Yok
Konu"1" sayfa 633	Yok	Yok	Yok	Yok	Y
İLETIM "1" sayfa 633	Yok	Y	Yok	Yok	Yok

Not:

1. Bu özelliklerin uygulama düzeyi özellikleri yoktur, ancak isteğe bağlı olarak denetlenen özellikler kullanılarak ayarlanabilirler.

OutboundSNI özellik

V 9.3.0

IBM MQ 9.3.0içinden, bir uygulamada **OutboundSNI** özelliğini ayarlayan XMSC_WMQ_OUTBOUND_SNI özelliğini ayarlayabilirsiniz.

XMSC_WMQ_OUTBOUND_SNI_PROPERTY aşağıdaki değerleri alır:

- "CHANNEL" ile eşlenen XMSC_WMQ_OUTBOUND_SNI_CHANNEL
- XMSC_WMQ_OUTBOUND_SNI_HOSTNAME, "HOSTNAME" ile eşlenir
- XMSC_WMQ_OUTBOUND_SNI_ASTERISK, "*" ile eşlenir

Buna ek olarak, **OutboundSNI** özelliğini aşağıdaki değerleri alan MQOUTBOUND_SNI ortam değişkenini kullanarak ayarlayabilirsiniz:

- Kanal
- ANASİSTEM ADI
- *

Not: Belirli bir değer ayarlanmazsa, özellik varsayılan olarak XMSC_WMQ_OUTBOUND_SNI_CHANNEL değerine ayarlanır.

Yönetilen düğümde **OutboundSNI** özelliğini ayarlamak için öncelik sırası:

1. Uygulama düzeyi özelliği
2. Ortam değişkeni

Yönetilmeyen düğümdeki **OutboundSNI** özelliği için yalnızca `mqclient.ini` desteklenir.

Yönetilen ConnectionFactory nesneleri için XMS .NET gerekli özellikler

Bir uygulama bir bağlantı üreticisi yarattığında, bir ileti sistemi sunucusuna bağlantı yaratmak için bazı özellikler tanımlanmalıdır.

Aşağıdaki çizelgelerde listelenen özellikler, bir uygulamanın ileti sistemi sunucusuna bağlantı yaratmak üzere ayarlanması için gereken alt sınırdır. Bir bağlantının yaratılma şeklini uyarlamak istiyorsanız, uygulamanız ConnectionFactory nesnesinin ek özelliklerini gereken şekilde ayarlayabilir. Daha fazla bilgi için bkz. ConnectionFactory. Kullanılabilir özelliklerin tam bir listesi eklenir.

IBM MQ kuyruk yöneticisine bağlantı

Çizelge 89. Bir IBM MQ kuyruk yöneticisine yönelik bağlantılar için ConnectionFactory nesnelerini yönetmeye ilişkin özellik ayarları

Gerekli XMS özelliği	Eşdeğer IBM MQ JMS özelliği gerekli
XMSC_CONNECTION_TYPE	XMS , bağlantı üreticisi sınıfı adı ve TRANSPORT (TRAN) özelliğinden bu şekilde çalışır.
XMSC_WMQ_HOST_NAME	ANASISTEM ADı (ANASISTEM)
XMSC_WMQ_PORT	PORT
XMSC_WMQ_QUEUE_MANAGER	Kuyruk yöneticisinin adı

Bir aracıya gerçek zamanlı bağlantı

Çizelge 90. Bir aracıya gerçek zamanlı bağlantılar için denetlenen ConnectionFactory nesnelerine ilişkin özellik ayarları

ZorunluXMS	Eşdeğer IBM MQ JMS özelliği gerekli
XMSC_CONNECTION_TYPE	XMS , bağlantı üreticisi sınıfı adı ve TRANSPORT (TRAN) özelliğinden bu şekilde çalışır.
XMSC_RTT_HOST_NAME	ANASISTEM ADı (ANASISTEM)
XMSC_RTT_PORT	PORT

WebSphere Application Server service integration bus bağlantısı

Çizelge 91. Bir WebSphere Application Server service integration bus bağlantısı için yönetilen ConnectionFactory nesnelerine ilişkin özellik ayarları

XMS özellik	Açıklama
XMSC_CONNECTION_TYPE	Bir uygulamanın bağlandığı ileti sistemi sunucusunun tipi.. Bu, bağlantı üreticisi sınıfı adından saptanır.
XMSC_WPM_BUS_NAME	Bir bağlantı üreticisi için, uygulamanın bağlandığı hizmet tümleştirme veriyolunun adı ya da hedef için, hedefin bulunduğu hizmet tümleştirme veriyolunun adı.

XMS .NET denetlenen Hedef nesneler için gerekli özellikler

Hedef yaratan bir uygulama, yönetilen bir Hedef nesnesinde uygulamanın belirlediği bazı özellikleri ayarlamalıdır.

Çizelge 92. Yönetilen Hedef nesnelere ilişkin özellik ayarları

Bağlantı tipi	Özellik	Açıklama
IBM MQ Kuyruk Yöneticisi	KUYRUK (QU) KONU (üST)	Bağlanmak istediğiniz kuyruk Uygulamanın hedef olarak kullandığı konu
Bir aracıya gerçek zamanlı bağlantı	KONU (üST)	Uygulamanın hedef olarak kullandığı konu
WebSphere Application Server service integration bus	topicName queueName	Uygulamanız bir konuya bağlanıyorsa Uygulamanız bir kuyruğa bağlanıyorsa

XMS .NET yönetilen nesneler oluşturma

XMS uygulamalarının bir ileti sistemi sunucusuyla bağlantı kurmak için gereken ConnectionFactory ve Hedef nesne tanımlamaları, uygun denetim araçları kullanılarak yaratılmalıdır.

Başlamadan önce

XMS tarafından desteklenen farklı yönetilen nesne havuzu tipleriyle ilgili daha fazla ayrıntı için bkz. “XMS .NET desteklenen yönetilen nesne havuzu tipleri” sayfa 631.

Bu görev hakkında

IBM MQ için yönetilen nesneleri oluşturmak üzere IBM MQ Explorer ya da IBM MQ JMS yönetimi (JMSAdmin) aracını kullanın.

IBM MQ ya da IBM Integration Busiçin yönetilen nesneler oluşturmak üzere IBM MQ JMS yönetimi (JMSAdmin) aracını kullanın.

WebSphere Application Server service integration busiçin yönetilen nesneler oluşturmak üzere WebSphere Application Server yönetim konsolunu kullanın.

Yönetim araçlarında özellik, kısaca **APPLICATIONNAME** veya **APPNAME** olarak bilinir.

Not: TRANSPORT (UNMANAGED) ayarını tanımlamak için JMSAdmin 'i kullanamazsınız. Bu nedenle, denetimci olarak seçilen bir uygulama adını kullanarak yönetilmeyen bir XMS istemcisi almak için aşağıdaki komutu girmeniz gerekir:

```
cf.SetIntProperty(XMSC.WMQ_CONNECTION_MODE, XMSC.WMQ_CM_CLIENT_UNMANAGED);
```

Aşağıdaki adımlarda, yönetilen nesneler oluşturmak için neler yapacağınız özetlenmektedir.

Yordam

1. Bir bağlantı üreticisi yaratın ve uygulamanızdan seçtiğiniz sunucuya bağlantı yaratmak için gereken özellikleri tanımlayın.

   XMS ' in bağlantı kurmak için gerektirdiği en düşük özellikler “Yönetilen ConnectionFactory nesneleri için XMS .NET gerekli özellikler” sayfa 633içinde tanımlanır.
2. Uygulamanızın bağlandığı ileti sistemi sunucusunda gereken hedefi yaratın:
   - Bir IBM MQ kuyruk yöneticisine yönelik bağlantı için bir kuyruk ya da konu yaratın.
   - Bir aracıya gerçek zamanlı bağlantı için bir konu yaratın.
   - WebSphere Application Server service integration busbağlantısı için bir kuyruk ya da konu oluşturun.

   XMS ' in bağlantı kurmak için gerektirdiği en düşük özellikler “XMS .NET denetlenen Hedef nesneler için gerekli özellikler” sayfa 634içinde tanımlanır.

XMS .NET InitialContext nesneleri oluşturma

Bir uygulama, gerekli yönetilen nesneleri almak üzere yönetilen nesneler havuzuna bağlantı kurmak için kullanılacak bir başlangıç bağlamı yaratmalıdır.

Bu görev hakkında

InitialContext nesnesi, havuza yönelik bir bağlantıyı içerir. XMS API, aşağıdaki görevleri gerçekleştirmek için yöntemler sağlar:

- InitialContext nesnesi yarat
- Yönetilen nesne havuzunda denetlenen bir nesneyi arayın.

Yordam

- InitialContext nesnesi yaratılmasıyla ilgili daha fazla ayrıntı için .NET ve InitialContextiçin InitialContext başlıklı konuya bakın.

XMS .NET InitialContext özellikleri

InitialContext oluşturucusunun değiştirgeleri, birörnek kaynak göstergesi (URI) olarak verilen, denetlenen nesneler havuzunun yerini içerir. Bir uygulamanın havuzla bağlantı kurması için, URI ' de bulunan bilgilerden daha fazla bilgi sağlanması gerekebilir.

JNDI 'da ve XMS' in .NET somutlamasında, ek bilgiler oluşturucuya Hashtable ortamında sağlanır.

Yönetilen nesne havuzunun yeri XMSC_IC_URL özelliğinde tanımlanır. Bu özellik genellikle Create çağrısında iletilir, ancak aramadan önce farklı bir adlandırma dizinine bağlanmak için değiştirilebilir. FileSystem ya da LDAP bağlamları için bu özellik, dizinin adresini tanımlar. COS adlandırması için, JNDI dizinine bağlanmak üzere bu özellikleri kullanan web hizmetinin adresidir.

Aşağıdaki özellikler, JNDI dizinine bağlanmak için kullanılacak web hizmetine değiştirilmeden geçirilir.

- XMSC_IC_PROVIDER_URL
- XMSC_IC_SECURITY_CREDENTIALS
- XMSC_IC_SECURITY_AUTHENTICATION
- XMSC_IC_SECURITY_PRINCIPAL
- XMSC_IC_SECURITY_PROTOCOL

XMS başlangıç bağlamları için URI biçimi

Yönetilen nesnelerin havuzunun konumu, birörnek kaynak göstergesi (URI) olarak sağlanır. URI ' nin biçimi bağlam tipine bağlıdır.

FileSystem bağlamı

FileSystem bağlamı için URL , dosya sistemi tabanlı dizinin konumunu gösterir. URL 'nin yapısı RFC 1738, *Uniform Resource Locators (URL)*tarafından tanımlandığı gibidir: URL önekine sahiptir `file://`ve bu öneki izleyen sözdizimi, XMS ' un çalıştığı sistemde açılabilen bir dosyanın geçerli bir tanımıdır.

Bu sözdizimi altyapıya özgü olabilir ve '/ayırıcıları ya da' \ ' ayırıcılarını kullanabilir. '\' kullanırsanız, her ayırıcıya ek bir '\' kullanılarak çıkış karakteri eklenmesi gerekir. Bu, .NET çerçevesinin ayırıcıyı, aşağıdaki karakterler için bir çıkış karakteri olarak yorumlamaya çalışmasını önler.

Bu örnekler bu sözdizimini göstermektedir:

```
file://myBindings
file:///admin/.bindings
file://\\admin\\.bindings
file://c:/admin/.bindings
file://c:\\admin\\.bindings
file://\\\\madison\\shared\\admin\\.bindings
file:///usr/admin/.bindings
```

LDAP bağlamı

LDAP bağlamı için, URL ' nin temel yapısı RFC 2255, *LDAP URL Format*, büyük ve küçük harfe duyarsız önekle `ldap://` tanımlanmıştır.

Kesin sözdizimi aşağıdaki örnekte gösterilmiştir:

```
LDAP://[Hostname][:Port]["/"[DistinguishedName]]
```

Bu sözdizimi RFC ' de tanımlandığı gibidir, ancak herhangi bir öznitelik, kapsam, süzgeç ya da uzantı desteği yoktur.

Bu sözdizimine ilişkin örnekler şunlardır:

```
ldap://madison:389/cn=JMSData,dc=IBM,dc=UK
ldap://madison/cn=JMSData,dc=IBM,dc=UK
LDAP:///cn=JMSData,dc=IBM,dc=UK
```

WSS bağlamı

WSS bağlamı için URL , `http://`önekine sahip bir web hizmetleri uç noktası biçimindedir.

Diğer bir seçenek olarak, `cosnaming://` ya da `wsvc://`önekini kullanabilirsiniz.

Bu iki önek, http üzerinden erişilen URL ile bir WSS bağlamı kullandığınız anlamına gelir; bu, ilk bağlam tipinin doğrudan URL' den türetilmesini sağlar.

Bu sözdizimine ilişkin örnekler şunlardır:

```
http://madison.ibm.com:9080/xmsjndi/services/JndiLookup
cosnaming://madison/jndilookup
```

XMS .NET için JNDI Arama Web hizmeti

Bir COS adlandırma dizinine XMSiçinden erişmek için, bir JNDI Arama web hizmeti WebSphere Application Server service integration bus sunucusunda konuşlandırılmalıdır. Bu web hizmeti, COS adlandırma hizmetindeki Java bilgilerini XMS uygulamalarının okuyabileceği bir forma çevirir.

Web hizmeti, kuruluş dizininde bulunan SIBXJndiLookupEAR.earkurumsal arşiv dosyasında sağlanır. Geçerli IBM MQ Message Service Client (XMS) for .NETyayını için SIBXJndiLookupEAR.ear , *install_dir*`\java\lib` dizininde bulunabilir. Bu, yönetim konsolu ya da wsaadmin komut dosyası oluşturma aracı kullanılarak bir WebSphere Application Server service integration bus sunucusuna kurulabilir. Web hizmeti uygulamalarının konuşlandırılmasına ilişkin ek bilgi için ürün belgelerine bakın.

XMS uygulamaları içinde web hizmetini tanımlamak için, InitialContext nesnesinin XMSC_IC_URL özelliğini URLweb hizmeti uç noktasına ayarlamanız yeterlidir. Örneğin, web hizmeti MyServeradlı bir sunucu anasisteminde konuşlandırıldıysa, bir web hizmeti uç noktası örneği URL:

```
wsvc://MyHost:9080/SIBXJndiLookup/services/JndiLookup
```

XMSC_IC_URL özelliğinin ayarlanması, InitialContext Arama çağrılarına tanımlı uç noktada web hizmetini çağırma izni verir; bu da COS adlandırma hizmetinden denetlenen gerekli nesneyi arar.

.NET uygulamaları web hizmetini kullanabilir. Sunucu tarafı konuşlandırması XMS C, /C++ ve XMS .NETiçin aynıdır. XMS .NET , web hizmetini doğrudan Microsoft .NET Frameworkaracılığıyla çağırır.

XMS .NET yönetilen nesnelerin alınması

XMS , InitialContext nesnesi yaratıldığında ya da InitialContext özelliklerinde sağlanan adresi kullanarak havuzdan denetlenen bir nesneyi alır.

Alınacak nesneler aşağıdaki ad tiplerine sahip olabilir:

- Hedef nesnesini açıklayan basit bir ad; örneğin, SalesOrders adlı bir kuyruk hedefi
- '/' ile ayrılmış olarak SubContexts' dan oluşabilen ve nesne adıyla bitmesi gereken bileşik ad. Bileşik ad örneği: "Warehouse/PickLists/DispatchQueue2"; burada Warehouse ve Picklist 'ler adlandırma dizininde SubContexts ve DispatchQueue2 hedef nesnenin adıdır.

Uygulamaların daha yeni bir XMS sürümünü kullanmasını önleme

Varsayılan olarak, daha yeni bir XMS sürümü kurulduğunda, önceki sürümü kullanan uygulamalar recompile.However, uygulama yapılandırma dosyasında bir öznitelik ayarlayarak uygulamaların daha yeni sürümü kullanmasını önleyebilirsiniz.

Bu görev hakkında

Birden çok sürüm birlikte var olma özelliği, daha yeni bir XMS sürümünün kuruluşunun önceki XMS sürümünün üzerine yazmamasını sağlar. Bunun yerine, benzer XMS .NET düzeneklerinin birden çok örneği Global Assembly Cache (GAC) içinde birlikte bulunur, ancak farklı sürüm numaralarına sahiptir. Dahili olarak GAC, uygulama çağrılarını XMS' un en son sürümüne yönlendirmek için bir ilke dosyası kullanır. Uygulamalar yeniden derleme gereksinimi olmadan çalışır ve daha yeni XMS .NET sürümünde bulunan yeni özellikleri kullanabilir.

Yordam

- Daha eski XMS .NET sürümünü kullanmak için bir uygulama gerekiyorsa, uygulama yapılandırma dosyasında `publisherpolicy` özniteliğini no olarak ayarlayın.

  Not: Uygulama yapılandırma dosyası, dosyanın ilişkili olduğu yürütülür programın adını içeren .config sonekini içeren bir dosyadır. Örneğin, text.exe için uygulama yapılandırma dosyası text.exe.configadını içerir.

  Ancak, herhangi bir zamanda, bir sistemin tüm uygulamaları aynı XMS .NETsürümünü kullanır.

XMS uygulamaları için iletişimi güvenceye alma

Bu bölümde, XMS uygulamalarının bir WebSphere Application Server service integration bus ileti alışverişi altyapısına ya da IBM MQ kuyruk yöneticisine SSL (Secure Sockets Layer; Güvenli Yuva Katmanı) aracılığıyla bağlanmasını sağlamak için güvenli iletişim ayarlamaya ilişkin bilgiler sağlanır.

Bu görev hakkında

Bölüm aşağıdaki konuları içerir:

- “IBM MQ kuyruk yöneticisine güvenli bağlantılar” sayfa 638
- “Bir IBM MQ kuyruk yöneticisine XMS bağlantıları için CipherSuite ve CipherSpec ad eşlemeleri” sayfa 639
- “WebSphere Application Server service integration bus ileti alışverişi altyapısına güvenli bağlantılar” sayfa 640
- “WebSphere Application Server service integration bus bağlantılarına ilişkin CipherSuite ve CipherSpec ad eşlemeleri” sayfa 640

IBM MQ kuyruk yöneticisine güvenli bağlantılar

Bir XMS .NET uygulamasının bir IBM MQ kuyruk yöneticisiyle güvenli bağlantı kurmasını sağlamak için ilgili özellikler ConnectionFactory nesnesinde tanımlanmalıdır.

Şifreleme anlaşmasında kullanılan protokol, ConnectionFactory nesnesinde belirlediğiniz CipherSuite ' e bağlı olarak SSL (Secure Sockets Layer; Güvenli Yuva Katmanı) ya da TLS (Transport Layer Security; İletim Katmanı Güvenliği) olabilir.

ConnectionFactory özellikleri, SSL kullanan bir IBM MQ kuyruk yöneticisine ilişkin kısa bir açıklamayla birlikte aşağıdaki çizelgede gösterilir:

Çizelge 93. SSL aracılığıyla IBM MQ kuyruk yöneticisine bağlantılar için ConnectionFactory özellikleri	
Özelliğin adı	**Açıklama**
XMSC_WMQ_SSL_CERT_STORES	Kuyruk yöneticisine SSL bağlantısında kullanılacak sertifika iptal listelerini (CRL) bulunduran sunucuların yerleri.
XMSC_WMQ_SSL_CIPHER_SPEC	Bir kuyruk yöneticisine güvenli bir bağlantıda kullanılacak CipherSpec ' in adı.
XMSC_WMQ_SSL_CIPHER_SUITE	Bir kuyruk yöneticisine TLS bağlantısında kullanılacak CipherSuite ' in adı. Güvenli bağlantının kararlaştırılmasında kullanılan protokol, belirtilen CipherSuite' e bağlıdır.
XMSC_WMQ_SSL_CRYPTO_HW	İstemci sistemine bağlı şifreleme donanımına ilişkin yapılandırma ayrıntıları.
XMSC_WMQ_SSL_FIPS_REQUIRED	Bu özelliğin değeri, bir uygulamanın FIPS uyumlu olmayan şifreleme takımlarını kullanıp kullanamayacağını belirler. Bu özellik true olarak ayarlanırsa, istemci-sunucu bağlantısı için yalnızca FIPS algoritmaları kullanılır.
XMSC_WMQ_SSL_KEY_REPOSITORY	Anahtarların ve sertifikaların saklandığı anahtar veritabanı dosyasının konumu.
XMSC_WMQ_SSL_KEY_RESETCOUNT	KeyResetSayısı, gizli anahtar yeniden anlaşılmadan önce SSL etkileşimi içinde gönderilen ve alınan toplam şifrelenmemiş bayt sayısını temsil eder.
XMSC_WMQ_SSL_PEER_NAME	Kuyruk yöneticisine SSL bağlantısında kullanılacak eşdüzey ad.

Bir IBM MQ kuyruk yöneticisine XMS bağlantıları için CipherSuite ve CipherSpec ad eşlemeleri

InitialContext JMSAdmin Connection Factory özelliği SSLCIPHERSUITE ile XMS neredeyse eşdeğer XMSC_WMQ_SSL_CIPHER_SPEC arasında çevrilir. XMSC_WMQ_SSL_CIPHER_SUITE için bir değer belirtirseniz, ancak XMSC_WMQ_SSL_CIPHER_SPEC için değer atlarsanız benzer bir çeviri gereklidir.

Çizelge 94 sayfa 639 içinde kullanılabilir CipherSpecs ve JSSE CipherSuite eşdeğerleri listelenir.

Çizelge 94. Kullanılabilir CipherSpecs ve JSSE CipherSuite eşdeğerleri	
CipherSpec	**Eşdeğer JSSE CipherSuite**
TLS_RSA_WITH_3DES_EDE_CBC_SHA	SSL_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_DES_CBC_SHA	SSL_RSA_WITH_DES_CBC_SHA

Not: Deprecated TLS_RSA_WITH_3DES_EDE_CBC_SHA kullanımdan kaldırıldı. Ancak, bağlantı AMQ9288hatasıyla sonlandırılmadan önce 32 GB ' ye kadar veri aktarmak için kullanılabilir. Bu hatayı önlemek için üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckullanırken gizli anahtar sıfırlamasını etkinleştirmeniz gerekir.

WebSphere Application Server service integration bus ileti alışverişi altyapısına güvenli bağlantılar

Bir XMS .NET uygulamasının bir WebSphere Application Server service integration bus ileti alışverişi altyapısıyla güvenli bağlantılar kurmasını sağlamak için, ilgili özelliklerin ConnectionFactory nesnesinde tanımlanması gerekir.

XMS , WebSphere Application Server service integration busbağlantıları için SSL ve HTTPS desteği sağlar. SSL ve HTTPS , kimlik doğrulama ve gizlilik için güvenli bağlantılar sağlar.

WebSphere Güvenlik gibi XMS güvenliği, güvenli bir bağlantı kararlaştırılırken kullanılan algoritmaları belirtmek için CipherSuites kullanımını içeren JSSE güvenlik standartları ve adlandırma kurallarına göre yapılandırılır. Şifreleme anlaşmasında kullanılan protokol, ConnectionFactory nesnesinde belirlediğiniz CipherSuite ' e bağlı olarak SSL ya da TLS olabilir.

Çizelge 95 sayfa 640 içinde, ConnectionFactory nesnesinde tanımlanması gereken özellikler listelenir.

Çizelge 95. WebSphere Application Server service integration bus ileti alışverişi altyapısına güvenli bağlantılar için ConnectionFactory özellikleri

Özelliğin adı	**Açıklama**
XMSC_WPM_SSL_CIPHER_SUITE	Bir WebSphere Application Server service integration bus ileti alışverişi altyapısına TLS bağlantısında kullanılacak CipherSuite ' in adı. Güvenli bağlantının kararlaştırılmasında kullanılan protokol, belirtilen CipherSuite' e bağlıdır.
XMSC_WPM_SSL_KEYRING_LABEL	Sunucuyla kimlik doğrulaması yaparken kullanılacak sertifika.

Aşağıda, bir WebSphere Application Server service integration bus ileti alışverişi altyapısına güvenli bağlantılar için ConnectionFactory özelliklerinin bir örneği verilmiştir:

```
cf.setStringProperty(XMSC_WPM_PROVIDER_ENDPOINTS, host_name:port_number:chain_name);
cf.setStringProperty(XMSC_WPM_SSL_KEY_REPOSITORY, key_repository_pathname);
cf.setStringProperty(XMSC_WPM_TARGET_TRANSPORT_CHAIN, transport_chain);
cf.setStringProperty(XMSC_WPM_SSL_CIPHER_SUITE, cipher_suite);
cf.setStringProperty(XMSC_WPM_SSL_KEYRING_STASH_FILE, stash_file_pathname);
```

Burada zincir_adı BootstrapTunneledSecureMessaging ya da BootstrapSecureMessaging ve port_number, önyükleme sunucusunun gelen istekleri dinlediği kapının numarasıdır.

Aşağıda, örnek değerler eklenmiş bir WebSphere Application Server service integration bus ileti alışverişi altyapısına güvenli bağlantılar için ConnectionFactory özellikleri örneği verilmiştir:

```
  /* CF properties needed for an SSL connection */
    cf.setStringProperty(XMSC_WPM_PROVIDER_ENDPOINTS,"localhost:7286:BootstrapSecureMessaging");
    cf.setStringProperty(XMSC_WPM_TARGET_TRANSPORT_CHAIN,"InboundSecureMessaging");
    cf.setStringProperty(XMSC_WPM_SSL_KEY_REPOSITORY,"C:\\Program Files\\IBM\\gsk7\\bin\
\XMSkey.kdb");
    cf.setStringProperty(XMSC_WPM_SSL_KEYRING_STASH_FILE,"C:\\Program Files\\IBM\\gsk7\\bin\
\XMSkey.sth");
    cf.setStringProperty(XMSC_WPM_SSL_CIPHER_SUITE,"SSL_RSA_EXPORT_WITH_RC4_40_MD5");
```

WebSphere Application Server service integration bus bağlantılarına ilişkin CipherSuite ve CipherSpec ad eşlemeleri

IBM Global Security Kit (GSKit) CipherSuitesyerine CipherSpecs kullandığından, XMSC_WPM_SSL_CIPHER_SUITE özelliğinde belirtilen JSSE stili CipherSuite adları GSKit-style CipherSpec adlarıyla eşlenmelidir.

Çizelge 96 sayfa 641 içinde, tanınan her CipherSuiteiçin eşdeğer CipherSpec listelenir.

Çizelge 96. Kullanılabilir CipherSuites ve eşdeğer CipherSpecs	
CipherSuite	**CipherSpec eşdeğeri**
TLS_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA	TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA

Not: Deprecated TLS_RSA_WITH_3DES_EDE_CBC_SHA kullanımdan kaldırıldı. Ancak, bağlantı AMQ9288hatasıyla sonlandırılmadan önce 32 GB ' ye kadar veri aktarmak için yine de kullanılabilir. Bu hatayı önlemek için üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpec' i kullanırken gizli anahtar sıfırlamasını etkinleştirmeniz gerekir.

XMS ileti

Bu bölümde, XMS iletilerinin yapısı ve içeriği ve uygulamaların XMS iletilerini nasıl işlediği açıklanmaktadır.

Bu bölüm aşağıdaki konuları içerir:

- “XMS iletisinin kısımları” sayfa 641
- “XMS iletisindeki üstbilgi alanları” sayfa 641
- “XMS iletisinin özellikleri” sayfa 642
- “XMS iletisinin gövdesi” sayfa 645
- “İleti seçiciler” sayfa 648
- “XMS iletilerinin IBM MQ iletileriyle eşlenmesi” sayfa 649

XMS iletisinin kısımları

XMS iletisi, bir üstbilgi, bir özellikler kümesi ve bir gövdeden oluşur.

Üstbilgi
Bir iletinin üstbilgisi alanlar içeriyor ve tüm iletiler aynı üstbilgi alanları kümesini içeriyor. XMS ve uygulamalar, iletileri tanımlamak ve yönlendirmek için üstbilgi alanlarının değerlerini kullanır. Üstbilgi alanları hakkında daha fazla bilgi için bkz. “XMS iletisindeki üstbilgi alanları” sayfa 641.

Özellikler kümesi
Bir iletinin özellikleri, iletiye ilişkin ek bilgileri belirtir. Tüm iletiler aynı üstbilgi alanları kümesine sahip olsa da, her ileti farklı bir özellik kümesine sahip olabilir. Daha fazla bilgi için, bkz. “XMS iletisinin özellikleri” sayfa 642.

Gövde
Bir iletinin gövdesi uygulama verilerini içerir. Daha fazla bilgi için, bkz. “XMS iletisinin gövdesi” sayfa 645.

Bir uygulama, hangi iletileri almak istediğini seçebilir. Seçim ölçütlerini belirten ileti seçiciler kullanılarak. Ölçütler, belirli üstbilgi alanlarının değerlerini ve bir iletinin özelliklerinden herhangi birinin değerlerini temel alabilir. İleti seçiciler hakkında daha fazla bilgi için bkz. “İleti seçiciler” sayfa 648.

XMS iletisindeki üstbilgi alanları

XMS uygulamasının bir WebSphere JMS uygulamasıyla ileti alışverişi yapmasına izin vermek için XMS iletisinin üstbilgisi, JMS ileti üstbilgisi alanlarını içerir.

Bu üstbilgi alanlarının adları JMSönekiyle başlar. JMS ileti üstbilgisi alanlarının bir açıklaması için bkz. *Java Message Service Belirtimi*.

XMS , bir İleti nesnesinin öznitelikleri olarak JMS ileti üstbilgisi alanlarını uygular. Her üstbilgi alanının değerini ayarlamak ve almak için kendi yöntemleri vardır. Bu yöntemlere ilişkin açıklamalar için bkz. IMessage. Üstbilgi alanı her zaman okunabilir ve yazılabilir.

Çizelge 97 sayfa 642 , JMS ileti üstbilgisi alanlarını listeler ve iletilen bir ileti için her bir alanın değerinin nasıl ayarlandığını gösterir. Bazı alanlar, bir uygulama bir ileti gönderdiğinde ya da JMSReteslim edilirse, bir uygulama bir ileti aldığında XMS tarafından otomatik olarak ayarlanır.

Çizelge 97. JMS ileti üstbilgisi alanları.]

JMS ileti üstbilgisi alanının adı	**İletilen bir ileti için değerin nasıl ayarlandığı (*yöntem* [*sınıf*] biçiminde)**
JMSCorrelationID	JMSCorrelationID değerini ayarla [İleti]
JMSDeliveryMode	Gönder: [MessageProducer]
JMSDestination (JMS Hedefi)	Gönder: [MessageProducer]
JMSExpiration	Gönder: [MessageProducer]
JMSMessageID	Gönder: [MessageProducer]
JMS Önceliği	Gönder: [MessageProducer]
Yeniden Teslim edilen JMS	Al [MessageConsumer]
JMSReplyTo	JMSReplyTo Olarak Ayarla [İleti]
JMSTimestamp	Gönder: [MessageProducer]
JMSType	JMSType 'ı ayarla [İleti]

XMS iletisinin özellikleri

XMS üç tip ileti özelliğini destekler: JMS tanımlı özellikler, IBM tanımlı özellikler ve uygulama tanımlı özellikler.

Bir XMS uygulaması, XMS bir İleti nesnesinin aşağıdaki önceden tanımlanmış özelliklerini desteklediğinden WebSphere JMS uygulamasıyla ileti değiş tokuşu yapabilir:

- WebSphere JMS tarafından desteklenen aynı JMStanımlı özellikler. Bu özelliklerin adları JMSX önekiyle başlar.
- WebSphere JMS tarafından desteklenen aynı IBMtanımlı özellikler. Bu özelliklerin adları JMS_IBM_ önekiyle başlar.

Önceden tanımlanmış her özelliğin iki adı vardır:

- IBMtanımlı bir özellik için, JMStanımlı bir özellik ya da WebSphere JMS adı için bir JMS adı.

  Bu, özelliğin JMS ya da WebSphere JMSiçinde bilindiği addır ve aynı zamanda bu özelliği içeren bir iletiyle iletilen addır. XMS uygulaması, bir ileti seçici ifadesinde özelliği tanımlamak için bu adı kullanır.
- Bir ileti seçici ifadesi dışında tüm durumlarda özelliği tanıtan bir XMS adı. Her XMS adı, `IBM.XMS.XMSC` sınıfında adlandırılmış bir değişmez olarak tanımlanır. Adı belirtilen değişmezin değeri, ilgili JMS ya da WebSphere JMS adıdır.

Önceden tanımlanmış özelliklere ek olarak, bir XMS uygulaması kendi ileti özellikleri kümesini oluşturabilir ve kullanabilir. Bu özelliklere *uygulama tanımlı özellikler* denir.

Bir uygulama bir ileti oluşturduktan sonra, iletinin özellikleri okunabilir ve yazılabilir olur. Uygulama iletiyi gönderdikten sonra özellikler okunabilir ve yazılabilir kalır. Bir uygulama bir ileti aldığında, iletinin özellikleri salt okunur olur. Bir uygulama, bir iletinin özellikleri salt okunur olduğunda İleti sınıfının `Clear Properties` yöntemini çağırırsa, özellikler okunabilir ve yazılabilir olur. Yöntem, özellikleri de temizler.

Alınan ileti, ileti özellikleri temizlendikten sonra iletildiğinde, standart bir WMQ XMS for .NET BytesMessage iletisinin davranışıyla tutarlı bir şekilde hareket eder ve ileti özellikleri temizlenir.

Ancak, aşağıdaki özellikler kaybedileceği için bu önerilmez:

- JMS_IBM_Encoding özellik değeri; ileti verilerinin kodunun anlamlı olarak çözülemeyeceğini belirtir.
- JMS_IBM_Format özellik değeri, (MQMD ya da yeni MQRFH2) ileti üstbilgisi ile var olan üstbilgiler arasındaki zincirleme üstbilginin bozulacağını belirtir.

Bir iletinin tüm özelliklerinin değerlerini saptamak için, bir uygulama İleti sınıfının Özellikleri Al yöntemini çağırabilir. Yöntem, her Özellik nesnesinin iletinin bir özelliğini temsil ettiği bir Özellik nesneleri listesini saran bir yineleyici yaratır. Uygulama daha sonra her bir Özellik nesnesini sırayla almak için Yineleyici sınıfının yöntemlerini kullanabilir ve her özelliğin adını, veri tipini ve değerini almak için Özellik sınıfının yöntemlerini kullanabilir.

JMS-defined properties of a message (İletinin tanımlı özellikleri)

Bir iletinin JMS tarafından tanımlanan bazı özellikleri hem XMS hem de WebSphere JMStarafından desteklenir.

Çizelge 98 sayfa 643 içinde hem XMS hem de WebSphere JMStarafından desteklenen bir iletinin JMS tanımlı özellikleri listelenir. JMStanımlı özelliklerin açıklaması için bkz. *Java Message Service Belirtim*. JMStanımlı özellikler, bir aracıya gerçek zamanlı bağlantı için geçerli değildir.

Çizelge, her özelliğin veri tipini belirtir ve iletilen bir ileti için özelliğin değerinin nasıl ayarlandığını gösterir. Bir uygulama bir ileti gönderdiğinde ya da JMSXDeliveryCountdurumunda, bir uygulama bir ileti aldığında, bazı özellikler XMS tarafından otomatik olarak ayarlanır.

Çizelge 98. JMS-defined properties of a message (İletinin tanımlı özellikleri)

JMS tanımlı özelliğin XMS adı	**JMS ad**	**Veri tipi**	**İletilen bir ileti için değerin nasıl ayarlandığı (*yöntem* [*sınıf*] biçiminde)**
JMSX_APPID	JMSXAppID	System.String	Gönder: [MessageProducer]
JMSX_DELIVERY_COUNT	JMSXDeliveryCount	System.Int32	Al [MessageConsumer]
JMSX_GRUPID	JMSXGroupID	System.String	Dizgi Özelliğini Ayarla [PropertyContext]
JMSX_GRUPSEQ	JMSXGroupSeq	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMSX_USERID	JMSXUserID	System.String	Gönder: [MessageProducer]

IBM-defined properties of a message (İletinin tanımlı özellikleri)

Bir iletinin bazı IBMtanımlı özellikleri XMS ve WebSphere JMStarafından desteklenir.

Çizelge 99 sayfa 644 içinde, hem XMS hem de WebSphere JMStarafından desteklenen bir iletinin IBM tanımlı özellikleri listelenir. IBMtanımlı özelliklerle ilgili daha fazla bilgi için IBM MQ ya da WebSphere Application Server ürün belgelerine bakın.

Çizelge, her özelliğin veri tipini belirtir ve iletilen bir ileti için özelliğin değerinin nasıl ayarlandığını gösterir. Bazı özellikler, bir uygulama ileti gönderdiğinde XMS tarafından otomatik olarak ayarlanır.

Çizelge 99. IBMtanımlı ileti özellikleri			
IBM tanımlı özelliğin XMS adı	**WebSphere JMS adı**	**Veri tipi**	**İletilen bir ileti için değerin nasıl ayarlandığı (*yöntem* [*sınıf*] biçiminde)**
JMS_IBM_CHARACTER_SET	JMS_IBM_Character_Set	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_ENCODING	JMS_IBM_Kodlaması	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_EXCEPTIONMESSAGE	JMS_IBM_ExceptionMessage	System.String	Al [MessageConsumer]
JMS_IBM_EXCEPTIONREASON	JMS_IBM_ExceptionReason	System.Int32	Al [MessageConsumer]
JMS_IBM_EXCEPTIONTIMESTAMP	JMS_IBM_ExceptionTimestamp	System.Int64	Al [MessageConsumer]
JMS_IBM_EXCEPTIONPROBLEM Hedef	JMS_IBM_ExceptionProblemHedefi	System.String	Al [MessageConsumer]
JMS_IBM_FEEDBACK	JMS_IBM_Feedback	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_FORMAT	JMS_IBM_Biçimi	System.String	Dizgi Özelliğini Ayarla [PropertyContext]
JMS_IBM_LAST_MSG_IN_GROUP	JMS_IBM_Last_Msg_In_Group	System.Boolean	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_MSGTYPE	JMS_IBM_MsgType	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_PUTAPPLTYPE	JMS_IBM_PutApplTipi	System.Int32	Gönder: [MessageProducer]
JMS_IBM_PUTDATE	JMS_IBM_PutDate	System.String	Gönder: [MessageProducer]
JMS_IBM_PUTTIME	JMS_IBM_PutTime	System.String	Gönder: [MessageProducer]
JMS_IBM_REPORT_COA	JMS_IBM_Report_COA	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_COD	JMS_IBM_Report_COD	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_DISCARD_MSG	JMS_IBM_Report_Discard_Msg	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_EXCEPTION	JMS_IBM_Report_Exception	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_SÜRESONU	JMS_IBM_Report_Expiration	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_NAN	JMS_IBM_Report_NAN	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]

Çizelge 99. IBMtanımlı ileti özellikleri (devamı var)			
IBM tanımlı özelliğin XMS adı	**WebSphere JMS adı**	**Veri tipi**	**İletilen bir ileti için değerin nasıl ayarlandığı (*yöntem* [*sınıf*] biçiminde)**
JMS_IBM_REPORT_PAN	JMS_IBM_Report_PAN	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_PASS_CORREL_ Tanıtıcı	JMS_IBM_Report_Pass_Correl_ID	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_PASS_MSG_ID	JMS_IBM_Report_Pass_Msg_ID	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_SYSTEM_MESSAGEID	JMS_IBM_System_MessageID	System.String	Gönder: [MessageProducer]

Bir iletinin uygulama tanımlı özellikleri

Bir XMS uygulaması kendi ileti özellikleri kümesini oluşturabilir ve kullanabilir. Bir uygulama bir ileti gönderdiğinde, bu özellikler de iletiyle birlikte iletilir. İleti seçicileri kullanan bir alıcı uygulama, bu özelliklerin değerlerine dayalı olarak hangi iletileri almak istediğini seçebilir.

Bir WebSphere JMS uygulamasının bir XMS uygulaması tarafından gönderilen iletileri seçmesine ve işlemesine izin vermek için, uygulama tanımlı bir özelliğin adı, ileti seçici ifadelerinde tanıtıcı oluşturmaya ilişkin kurallara uygun olmalıdır. Daha fazla bilgi için bkz “JMS içindeki ileti seçiciler” sayfa 139. Uygulama tanımlı bir özelliğin değeri şu veri tiplerinden birine sahip olmalıdır: System.Boolean, System.SByte, System.Int16, System.Int32, System.Int64, System.Float, System.Doubleya da System.String.

XMS iletisinin gövdesi

Bir iletinin gövdesi uygulama verilerini içerir. Ancak, bir iletinin gövdesi olamaz ve yalnızca üstbilgi alanlarını ve özelliklerini oluşturur.

XMS beş tip ileti gövdesini destekler:

Bayt
Gövde bir bayt akışı içeriyor. Bu gövde tipine sahip bir iletiye *byte iletisi*adı verilir. IBytesMessage arabirimi, bir bayt iletisinin gövdesini işleme yöntemlerini içerir.

Harita
Gövde, her değerin ilişkili bir veri tipine sahip olduğu bir ad-değer çiftleri kümesi içerir. Bu tip gövdeye sahip bir iletiye *eşlem iletisi*adı verilir. IMapMessage arabirimi, bir eşlem iletisinin gövdesinin işlenmesine ilişkin yöntemleri içerir.

Nesne
Gövde diziselleştirilmiş bir Java ya da .NET nesnesi içeriyor. Bu gövde tipine sahip bir iletiye *nesne iletisi*denir. IObjectMessage arabirimi, bir nesne iletisinin gövdesini işlemeye ilişkin yöntemleri içerir.

Akış
Gövde, her değerin ilişkili bir veri tipine sahip olduğu bir değer akışı içerir. Bu gövde tipine sahip bir iletiye *akış iletisi*denir. IStreamMessage arabirimi, bir akış iletisinin gövdesini işleme yöntemlerini içerir.

Metin
Gövde bir dizgi içeriyor. Bu gövde tipine sahip bir iletiye *metin iletisi*denir. ITextMessage arabirimi, bir metin iletisinin gövdesini işlemeye ilişkin yöntemleri içerir.

IMessage arabirimi, tüm ileti nesnelerinin üst öğesidir ve XMS ileti tiplerinden herhangi birini göstermek için ileti alışverişi işlevlerinde kullanılabilir.

Bu veri tiplerinin her birinin büyüklük ve üst sınır ve alt sınır değerleri hakkında bilgi için bkz. Çizelge 84 sayfa 622.

Bayt iletileri

Bir bayt iletisinin gövdesi bir bayt akışı içerir. Gövde yalnızca gerçek verileri içerir ve bu verileri yorumlamak için gönderen ve alan uygulamaların sorumluluğundadır.

Bir XMS uygulamasının XMS ya da JMS uygulama programlama arabirimini kullanmayan uygulamalarla ileti değiş tokuşu yapması gerekiyorsa, bayt iletileri yararlı olur.

Bir uygulama bayt iletisi yarattıktan sonra, iletinin gövdesi salt yazılır olur. Uygulama, .NETiçin IBytesMessage arabiriminin uygun yazma yöntemlerini çağırarak uygulama verilerini gövdede toplar. Uygulamanın byte ileti akımına her değer yazdığında, değer, uygulama tarafından yazılan önceki değerden hemen sonra toplanır. XMS , birleştirilen son baytın konumunu hatırlamak için bir iç imleç tutar.

Uygulama iletiyi gönderdiğinde, iletinin gövdesi salt okunur olur. Bu kipte, uygulama iletiyi yinelemeli olarak gönderebilir.

Bir uygulama bayt iletisi aldığında, iletinin gövdesi salt okunur olur. Uygulama, bayt ileti akışının içeriğini okumak için IBytesMessage arabiriminin uygun okuma yöntemlerini kullanabilir. Uygulama baytları sırayla okur ve XMS , okunan son baytın konumunu hatırlamak için bir iç imleç tutar.

Bir uygulama, bayt iletisi gövdesi yazılabilir olduğunda IBytesMessage arabiriminin Sıfırlama yöntemini çağırırsa, gövde salt okunur olur. Yöntem, imleci bayt ileti akımının başına da konumlandırır.

Bir uygulama, bir bayt iletisinin gövdesi salt okunur olduğunda .NET için IMessage arabiriminin `Clear` Body yöntemini çağırırsa, gövde yazılabilir olur. Bu yöntem gövdeyi de temizler.

İletileri eşle

Bir eşlem iletisinin gövdesi, her değerin ilişkili bir veri tipine sahip olduğu bir ad-değer çiftleri kümesi içerir.

Her ad-değer çiftinde, ad değeri tanımlayan bir dizedir ve değer, Çizelge 100 sayfa 648içinde listelenen XMS veri tiplerinden birine sahip uygulama verilerinin bir öğesidir. Ad-değer çiftlerinin sırası tanımlanmadı. MapMessage sınıfı, ad-değer çiftlerini ayarlama ve alma yöntemlerini içerir.

Bir uygulama, adını belirterek ad-değer çiftine rasgele erişebilir.

Bir .NET uygulaması, eşlem iletisinin gövdesindeki adlara ilişkin bir sıralı değer listesi almak için MapNames özelliğini kullanabilir.

Bir uygulama ad-değer çiftinin değerini aldığında, değer XMS tarafından başka bir veri tipine dönüştürülebilir. Örneğin, bir eşlem iletisinin gövdesinden bir tamsayı almak için, bir uygulama MapMessage sınıfının GetString yöntemini çağırabilir ve bu da tamsayıyı dizgi olarak döndürür. Desteklenen dönüştürmeler, XMS bir özellik değerini bir veri tipinden diğerine dönüştürdüğünde desteklenenlerle aynıdır. Desteklenen dönüştürmeler hakkında daha fazla bilgi için bkz. “Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi” sayfa 622.

Uygulama bir eşlem iletisi yarattıktan sonra, iletinin gövdesi okunabilir ve yazılabilir olur. Uygulama iletiyi gönderdikten sonra gövde okunabilir ve yazılabilir kalır. Bir uygulama bir eşlem iletisi aldığında, iletinin gövdesi salt okunur olur. Bir uygulama, bir eşlem iletisinin gövdesi salt okunur olduğunda İleti sınıfının Clear Body yöntemini çağırırsa, gövde okunabilir ve yazılabilir olur. Bu yöntem gövdeyi de temizler.

Nesne iletileri

Bir nesne iletisinin gövdesi diziselleştirilmiş birJava ya da .NET nesnesi içeriyor.

XMS uygulaması bir nesne iletisi alabilir, üstbilgi alanlarını ve özelliklerini değiştirebilir ve daha sonra bunu başka bir hedefe gönderebilir. Bir uygulama, bir nesne iletisinin gövdesini kopyalayıp başka bir nesne iletisi oluşturmak için de kullanabilir. XMS , bir nesne iletisinin gövdesini bayt dizisi olarak işler.

Bir uygulama bir nesne iletisi oluşturduktan sonra, iletinin gövdesi okunabilir ve yazılabilir olur. Uygulama iletiyi gönderdikten sonra gövde okunabilir ve yazılabilir kalır. Bir uygulama bir nesne iletisi aldığında, iletinin gövdesi salt okunur olur. Bir uygulama, bir nesne iletisinin gövdesi salt okunur olduğunda .NET için IMessage arabiriminin `Clear Body` yöntemini çağırırsa, gövde okunabilir ve yazılabilir olur. Bu yöntem gövdeyi de temizler.

Akış iletileri

Bir akış iletisinin gövdesi, her değerin ilişkili bir veri tipine sahip olduğu bir değer akışı içerir.

Bir değerin veri tipi, Çizelge 100 sayfa 648içinde listelenen XMS veri tiplerinden biridir.

Bir uygulama akış iletisi yarattıktan sonra, iletinin gövdesi yazılabilir olur. Uygulama, .NETiçin IStreamMessage arabiriminin uygun yazma yöntemlerini çağırarak uygulama verilerini gövdede toplar. Uygulamanın ileti akımına her değer yazdığında, değer ve veri tipi, uygulama tarafından yazılan önceki değerden hemen sonra toplanır. XMS , birleştirilen son değerin konumunu hatırlamak için bir iç imleç tutar.

Uygulama iletiyi gönderdiğinde, iletinin gövdesi salt okunur olur. Bu kipte, uygulama iletiyi birden çok kez gönderebilir.

Bir uygulama bir akış iletisi aldığında, iletinin gövdesi salt okunur olur. Uygulama, ileti akışının içeriğini okumak için .NET için IStreamMessage arabiriminin uygun okuma yöntemlerini kullanabilir. Uygulama değerleri sırayla okur ve XMS okunan son değerin konumunu hatırlamak için bir iç imleç tutar.

Bir uygulama ileti akışından bir değer okuduğunda, değer XMS tarafından başka bir veri tipine dönüştürülebilir. Örneğin, ileti akımından bir tamsayıyı okumak için bir uygulama, tamsayıyı dizgi olarak döndüren ReadString yöntemini çağırabilir. Desteklenen dönüştürmeler, XMS bir özellik değerini bir veri tipinden diğerine dönüştürdüğünde desteklenenlerle aynıdır. Desteklenen dönüştürmeler hakkında daha fazla bilgi için bkz. “Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi” sayfa 622.

Bir uygulama ileti akımından bir değeri okumaya çalışırken bir hata oluşursa, imleç ilerlemez. Uygulama, değeri başka bir veri tipi olarak okumaya çalışarak hatadan kurtulabilir.

Bir uygulama, akış iletisinin gövdesi salt yazılır olduğunda XMS için IStreamMessage arabiriminin Sıfırlama yöntemini çağırırsa, gövde salt okunur olur. Yöntem, imleci ileti akımının başına da konumlandırır.

Bir uygulama, bir akış iletisinin gövdesi salt okunur olduğunda XMS için IMessage arabiriminin `Clear Body` yöntemini çağırırsa, gövde salt okunur olur. Bu yöntem gövdeyi de temizler.

Metin iletileri

Bir metin iletisinin gövdesi bir dizgi içeriyor.

Bir uygulama bir metin iletisi oluşturduktan sonra, iletinin gövdesi okunabilir ve yazılabilir olur. Uygulama iletiyi gönderdikten sonra gövde okunabilir ve yazılabilir kalır. Bir uygulama bir metin iletisi aldığında, iletinin gövdesi salt okunur olur. Bir uygulama, bir metin iletisinin gövdesi salt okunur olduğunda .NET için IMessage arabiriminin Clear Body yöntemini çağırırsa, gövde okunabilir ve yazılabilir olur. Bu yöntem gövdeyi de temizler.

Uygulama verileri öğeleri için veri tipleri

Bir XMS uygulamasının bir IBM MQ classes for JMS uygulamasıyla ileti alışverişi yapabilmesini sağlamak için, her iki uygulamanın da bir iletinin gövdesindeki uygulama verilerini aynı şekilde yorumlayabilmesi gerekir.

Bu nedenle, bir XMS uygulaması tarafından bir iletinin gövdesine yazılan her uygulama verileri öğesi, Çizelge 100 sayfa 648içinde listelenen veri tiplerinden birine sahip olmalıdır. Çizelge, her veri tipi için uyumlu Java veri tipini gösterir. XMS , uygulama verilerinin öğelerini yalnızca bu veri tipleriyle yazma yöntemlerini sağlar.

Çizelge 100. Java veri tipleriyle uyumlu XMS veri tipleri		
XMS Veri tipi	**Gösterir**	**Uyumlu Java veri tipi**
System.Boolean	Boole değeri true ya da false	boole
System.Char16	Çift baytlık karakter	DAMGA
System.SByte	İşaretli 8 bitlik tamsayı	Byte
System.Int16	İşaretli 16 bitlik tamsayı	kısa
System.Int32	İmzalı 32 bitlik tamsayı	int
System.Int64	İşaretli 64 bitlik tamsayı	uzun
System.Float	İşaretli kayar noktalı sayı	kayan nokta
System.Double	İşaretli çift duyarlıklı kayan noktalı sayı	çift
System.String	Karakter dizesi	Dizgi

Bu veri tiplerinin her birinin büyüklük, değer üst sınırı ve değer alt sınırı hakkında bilgi için bkz. "XMS ilkel tipleri" sayfa 622.

İleti seçiciler

XMS uygulaması, almak istediği iletileri seçmek için ileti seçicileri kullanır.

Bir uygulama bir ileti tüketicisi oluşturduğunda, bir ileti seçici ifadesini tüketiciyle ilişkilendirebilir. İleti seçici ifadesi seçim ölçütlerini belirtir.

Bir uygulama IBM WebSphere MQ 7.0 kuyruk yöneticisine bağlanırken, kuyruk yöneticisi tarafında ileti seçimi yapılır. XMS , herhangi bir seçim yapmaz ve kuyruk yöneticisinden aldığı iletiyi teslim ederek daha iyi performans sağlar.

Bir uygulama, her biri kendi ileti seçici ifadesiyle birden çok ileti tüketicisi yaratabilir. Gelen bir ileti birden çok ileti tüketicisinin seçim ölçütlerini karşılıyorsa, XMS iletiyi bu tüketicilerin her birine gönderir.

Bir ileti seçici ifadesi, bir iletinin aşağıdaki özelliklerine gönderme yapabilir:

- JMS tanımlı özellikler
- IBMtanımlı özellikler
- Uygulama tanımlı özellikler

Aşağıdaki ileti üstbilgisi alanlarına da gönderme yapabilir:

- JMSCorrelationID
- JMSDeliveryMode
- JMSMessageID
- JMS Önceliği
- JMSTimestamp
- JMSType

Ancak, ileti seçici ifadesi bir iletinin gövdesindeki verilere gönderme yapamaz.

Aşağıda bir ileti seçici ifadesi örneği verilmiştir:

```
JMSPriority > 3 AND manufacturer = 'Jaguar' AND model in ('xj6','xj12')
```

XMS , bu ileti seçici ifadesiyle bir ileti tüketicisine, iletinin önceliği 3değerinden büyükse bir ileti gönderir; `Jaguar;` değerine sahip uygulama tanımlı bir özellik, üretici ve `xj6` ya da `xj12.` değerine sahip başka bir uygulama tanımlı özellik, model.

XMS içinde bir ileti seçici ifadesi oluşturmaya ilişkin sözdizimi kuralları, IBM MQ classes for JMSiçinde bulunanlarla aynıdır. İleti seçici ifadesi oluşturulmasıyla ilgili bilgi için IBM MQ ürün belgelerine bakın. Bir ileti seçici ifadesinde, JMStanımlı özelliklerin adlarının JMS adları olması ve IBMtanımlı özelliklerin adlarının IBM MQ classes for JMS adları olması gerektiğini unutmayın. İleti seçici ifadesinde XMS adlarını kullanamazsınız.

XMS iletilerinin IBM MQ iletileriyle eşlenmesi

Bir XMS iletisinin JMS üstbilgi alanları ve özellikleri, IBM MQ iletisinin üstbilgi yapılarındaki alanlarla eşlenir.

Bir XMS uygulaması bir IBM MQ kuyruk yöneticisine bağlandığında, kuyruk yöneticisine gönderilen iletiler, benzer koşullarda IBM MQ classes for JMS iletilerinin IBM MQ iletileriyle eşlendiği gibi IBM MQ iletileriyle eşlenir.

Bir Hedef nesnenin XMSC_WMQ_TARGET_CLIENT özelliği XMSC_WMQ_TARGET_DEST_JMS olarak ayarlanırsa, hedefe gönderilen bir iletinin JMS üstbilgi alanları ve özellikleri, IBM MQ iletisinin MQMD ve MQRFH2 üstbilgi yapılarıyla eşlenir. XMSC_WMQ_TARGET_CLIENT özelliğinin bu şekilde ayarlanması, iletiyi alan uygulamanın bir MQRFH2 üstbilgisini işleyebileceğini varsayar. Bu nedenle, alan uygulama, MQRFH2 üstbilgisini işlemek üzere tasarlanmış başka bir XMS uygulaması, IBM MQ classes for JMS uygulaması ya da yerel IBM MQ uygulaması olabilir.

Hedef nesnenin XMSC_WMQ_TARGET_CLIENT özelliği XMSC_WMQ_TARGET_DEST_MQ olarak ayarlanırsa, hedefe gönderilen bir iletinin JMS üstbilgi alanları ve özellikleri, IBM MQ iletisinin MQMD üstbilgi yapısındaki alanlarla eşlenir. İleti bir MQRFH2 üstbilgisi içermiyor ve MQMD üstbilgi yapısındaki alanlarla eşlenemeyen JMS üstbilgi alanları ve özellikleri yoksayıldı. Bu nedenle, iletiyi alan uygulama, MQRFH2 üstbilgisini işlemek üzere tasarlanmamış yerel bir IBM MQ olabilir.

Bir kuyruk yöneticisinden alınan IBM MQ iletileri, benzer koşullarda IBM MQ iletilerin IBM MQ classes for JMS iletileriyle eşlendiği gibi XMS iletileriyle eşlenir.

Gelen bir IBM MQ iletisinin MQRFH2 üstbilgisi varsa, sonuçtaki XMS iletisinin gövdesi, MQRFH2 üstbilgisinin `mcd` klasöründe bulunan **Msd** özelliğinin değerine göre belirlenir. **Msd** özelliği MQRFH2 üstbilgisinde yoksa ya da IBM MQ iletisinde MQRFH2 üstbilgisi yoksa, sonuçtaki XMS iletisinin tipi MQMD üstbilgisindeki *`Format`* alanının değeriyle belirlenir. *`Format`* alanı MQFMT_STRING olarak ayarlanırsa, XMS iletisi bir metin iletisidir. Tersi durumda, XMS iletisi bayt iletisidir. IBM MQ iletisinde MQRFH2 üstbilgisi yoksa, yalnızca MQMD üstbilgisindeki alanlardan türetilebilir JMS üstbilgi alanları ve özellikleri ayarlanır.

IBM MQ classes for JMS iletilerini IBM MQ iletileriyle eşleme hakkında daha fazla bilgi için bkz. “JMS iletilerinin IBM MQ iletileriyle eşlenmesi” sayfa 142.

IBM MQ Message Service Client (XMS) for .NET uygulamasından ileti tanımlayıcısının okunması ve yazılması

Bir IBM MQ iletisinin StrucId ve Version dışındaki tüm ileti tanımlayıcı (MQMD) alanlarına erişebilirsiniz; BackoutCount değeri okunabilir, ancak yazılamaz.

IBM MQ Message Service Client (XMS) for .NET tarafından sağlanan ileti öznitelikleri, XMS uygulamalarının MQMD alanlarını ayarlamasını ve IBM WebSphere MQ uygulamalarını da kullanmasını kolaylaştırır.

Yayınlama/abone olma ileti sistemi kullanılırken bazı kısıtlamalar geçerlidir. Örneğin, MsgID ve CorrelIdgibi MQMD alanları ayarlanırsa yoksayılır.

PROVIDERVERSION özelliği 6olarak ayarlandığında işlev de kullanılamaz.

Bir IBM MQ Message Service Client (XMS) for .NET uygulamasından IBM MQ ileti verilerine erişilmesi

Bir IBM MQ Message Service Client (XMS) for .NET uygulaması içindeki MQRFH2 üstbilgisi (varsa) ve diğer IBM MQ üstbilgileri (varsa) de dahil olmak üzere tüm IBM MQ ileti verilerine JMSBytesMessagegövdesi olarak erişebilirsiniz.

Bu konuda açıklanan işlev yalnızca bir IBM WebSphere MQ 7.0 ya da daha sonraki bir kuyruk yöneticisine bağlanırken ve IBM MQ ileti alışverişi sağlayıcısı normal kipte olduğunda kullanılabilir.

Hedef nesne özellikleri, XMS uygulamasının bir JMSBytesMessagegövdesi olarak bir IBM MQ iletisinin tamamına (varsa MQRFH2 üstbilgisi dahil) nasıl eriştiğini belirler.

ALW AMQP istemci uygulamalarının geliştirilmesi

AMQP API ' leri için IBM MQ desteği, IBM MQ yöneticisinin bir AMQP kanalı oluşturmasına olanak sağlar. Bu kanal başlatıldığında, AMQP istemci uygulamalarından gelen bağlantıları kabul eden bir kapı numarası tanımlar.

Bir AMQP kanalını AIX, Linux, and Windows sistemlerine kurabilirsiniz; bu kanal IBM i ya da z/OSüzerinde kullanılamaz.

Bir AMQP 1.0 istemci uygulaması, bir AMQP kanalı ile kuyruk yöneticisine bağlanabilir.

Apache Qpid JMS kitaplığını kullanarak uygulama geliştirilmesi

Giriş

Apache Qpid JMS kitaplığı, JMS 2 belirtiminin bir somutlamasını sağlamak için AMQP 1.0 iletişim kuralını kullanır.

Apache Qpid JMS , AMQP 1.0 iletişim kuralının bazı yönlerini MQ Light ileti sistemi API ' larından farklı bir şekilde kullanır. IBM MQ 9.2 , IBM MQ AMQP kanallarına destek ekler; böylece Apache Qpid JMS uygulamaları IBM MQ ' a bağlanabilir ve paylaşılan aboneliklerin kullanımı da dahil olmak üzere yayınlama/abone olma ileti sistemi yapabilir.

V 9.3.0 IBM MQ 9.3 , IBM MQ AMQP kanallarına daha fazla destek ekler; böylece Apache Qpid JMS uygulamaları IBM MQ ' a bağlanabilir ve noktadan noktaya ileti alışverişi gerçekleştirebilir. Ek bilgi için bkz. “AMQP kanallarında noktadan noktaya destek” sayfa 655 .

V 9.3.0 IBM MQ 9.3.0 , IBM MQ AMQP kanalları için daha fazla kuyruk göz atma desteği ekler; böylece Apache Qpid JMS uygulamaları IBM MQ olanağına bağlanabilir ve bir kuyruktan iletilere göz atma gerçekleştirebilir. Ek bilgi için bkz. “AMQP kanallarında noktadan noktaya destek” sayfa 655 .

V 9.3.0 IBM MQ 9.3.0 , AMQP kanalları için TMPMODEL ve TMPQPRFXolmak üzere iki ek kanal özniteliği ekler. Bu öznitelikler, geçici kuyruk yaratılırken kullanılacak model kuyruğu ve geçici kuyruk öneki içindir.

Diğer IBM MQ uygulamalarıyla iletişim

Apache Qpid JMS uygulamaları ile diğer IBM MQ uygulamaları arasında ileti gönderilebilir. Örneğin, bir Apache Qpid uygulaması bir konudaki iletileri yayınlayabilir ve MQ Light uygulamaları bunları bir abonelik oluşturarak alabilir.

Apache Qpid JMS uygulaması, geleneksel IBM MQ uygulamaları tarafından kullanılan iletileri de yayınlayabilir; örneğin, aynı konuya abone olmak için MQSUB API çağrısını kullanabilir.

Benzer şekilde, Apache Qpid JMS uygulamaları, geleneksel IBM MQ uygulamalarının ileti yayınladığı IBM MQ konularına abone olabilir.

Her iki istemci de aynı paylaşım adını ve konu kalıbını belirttiği sürece, Apache Qpid JMS uygulamasının bir MQ Light uygulamasıyla aboneliği paylaşması da mümkündür.

Bunu yapmak için Apache Qpid JMS uygulamasının bir istemci tanıtıcısıyla bağlanmaması gerektiğini unutmayın. Bu, her iki uygulama tarafından kullanılan IBM MQ abonelik adının aynı olmasını sağlar.

Uyarı: Apache Qpid JMS uygulamasının bir aboneliği IBM MQ JMS uygulamasıyla paylaşması mümkün değildir.

Apache Qpid JMS kısıtlamaları

Aşağıdaki JMS yetenekleri desteklenir:

- Müşteri onaylar, otomatik onaylar ve "dups" onay kipi (DUPS_OK_ALINDI bildirimi)
  - Kimlik bilgileriyle ya da kimlik bilgileri olmadan bağlanma
  - Konu hedefinde tüketici yaratılması
  - Konu hedefinde sürekli tüketici yaratılması
  - Konu hedefinde paylaşılan tüketici yaratılması
  - Konu hedefinde paylaşılan sürekli tüketici yaratılması
  - Müşteri onaylama ve otomatik onaylama kipleri
  - İleti alındı bildirimi ve oturum alındı bildirimi
  - Sürekli abonelikten aboneliği kaldırma
  - V 9.3.0 Geçici kuyruk yaratılması
  - V 9.3.0 Kuyrukta ya da geçici kuyruk hedefinde tüketici yaratılması
  - V 9.3.0 JMS MessageListeners
  - V 9.3.0 Gövdeyi alacak JMS Tüketicisi; `Consumer.receiveBody()` adlı JMS 2.0 yöntemi
  - V 9.3.0 Aşağıdaki JMS ileti tipleri desteklenir:
    - BytesMessage
    - MapMessage
    - ObjectMessage
    - StreamMessage
    - TextMessage
  - V 9.3.0 Kuyruktaki iletilere göz atma

Aşağıdaki JMS yetenekleri AMQP istemcileri tarafından desteklenmez:

- İşlemli oturumların ve işlemli JMSConmetin kullanımı
  - İleti seçicilerin kullanımı
  - **`nolocal`** özniteliğinin kullanımı
  - İşlemli oturumların kullanımı
  - Teslimat gecikmesinin kullanımı
  - IBM MQ 9.3.0adresinde, bir kuyruktaki iletilere göz atılıyor.
  - Aynı istemci tanıtıcısı ve konusuyla birden çok sürekli abonelik ya da tüketici yaratılması
  - V 9.3.0 JMS Geçici Konuları
  - AMQP süzgeçleri desteklenmiyor.

V 9.3.3 IBM MQ 9.3.3 ' den itibaren aşağıdaki not artık Continuous Delivery kullanıcıları için geçerli değildir.

Uyarı: V 9.3.0 Müşteri, kararsız AMQP ileti aktarımlarının kullanılması durumunda, yani bir müşterinin ileti onayının gerekli olduğu durumlarda, müşterilerin, ileti onaylarını, istemcinin onay

kipini kullanırken zamanında göndererek zamanında teslim etmesi ya da kuyruk yöneticisi özelliğini MARKINT (**MsgMarkBrowseInterval**) daha yüksek bir değere ayarlamayı düşünmesi gerektiğini kabul eder.

MsgMarkBrowseInterval için varsayılan değer beş saniyedir. Bir uygulama bu varsayılan değer içinde yerleşmezse, yinelenen iletiler görülebilir. İletilerin yinelenmesini önlemek için, **MsgMarkBrowseInterval** değerini, sınırsız bir zaman aralığını göstermek üzere ideal olarak NOLIMIT olarak ayarlamanız gerekir. Bir ileti çözülmeden önce bir uygulama çökerse ya da bağlantısı kesilirse, iletiler başka bir uygulama tarafından kullanılabilir duruma getirilir.

Daha fazla bilgi için bkz. **MsgMarkBrowseInterval** . Bu bir kuyruk yöneticisi özelliği olduğundan, ayarladığınız değer, o kuyruk yöneticisine bağlı tüm uygulamalara uygulanır.

AMQP ' de **MsgMarkBrowseInterval** , abonelikler için değil, yalnızca kuyruklar için geçerlidir.

Örnek AMQP istemcileri karşıdan yükleniyor

IBM MQ , AMQP istemcilerini göndermez, ancak MQ Light istemcilerini karşıdan yükleyebilir ya da Apache Qpid kitaplıklarına dayalı olarak açık kaynaklı AMQP istemcilerini karşıdan yükleyebilirsiniz. Daha fazla bilgi için bkz. IBM MQ Light ve Apache Qpid.

Apache Qpid kitaplıklarına dayalı olarak diğer açık kaynaklı AMQP istemcilerini de yükleyebilirsiniz. Daha fazla bilgi için bkz https://qpid.apache.org/index.html.

Uyarı: IBM Destek, bu müşteri paketleri için yapılandırma veya hata desteği sağlayamıyor ve herhangi bir kullanım sorusu veya kod hatası raporu ilgili projelere yönlendirilmelidir.

AMQP istemcilerinin IBM MQ ' a konuşlandırılması

Bir uygulama devreye alınmaya hazır olduğunda, diğer kurumsal uygulamaların tüm izleme, güvenilirlik ve güvenlik yeteneklerini gerektirir. Ayrıca diğer kurumsal uygulamalarla da veri alışverişi yapabilir.

Bir AMQP istemcisi konuşlandırdığınızda, IBM MQ uygulamalarıyla ileti alışverişi yapabilirsiniz. Örneğin, bir JavaScript dizgi iletisi göndermek için AMQP istemcisini kullanırsanız, IBM MQ uygulaması bir MQ iletisi alır; burada MQMD ' nin biçim alanı MQSTR olarak ayarlanır.

AMQP kanalının yönetilmesi

AMQP kanalı, diğer MQ kanallarıyla aynı şekilde yönetilebilir. Kanalları tanımlamak, başlatmak, durdurmak ve yönetmek için MQSC komutlarını, PCF komut iletilerini ya da IBM MQ Explorer komutunu kullanabilirsiniz. AMQP kanallarının oluşturulması ve kullanılmasıalanında, istemcilerin bir kuyruk yöneticisine bağlanmasını tanımlamak ve başlatmak için örnek komutlar sağlanır.

Bir AMQP kanalı başlatıldığında, AMQP 1.0 istemcisini bağlayarak kanalı sınayabilirsiniz. Örneğin, MQ Light, Apache Qpid Proton ya da Apache Qpid JMS.

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW MQ Light, Apache Qpid JMSve AMQP (Gelişmiş İleti Kuyruğa Alma İletişim Kuralı)

MQ Light istemcisi, Apache Qpid istemcileri (Apache Proton ve Apache Qpid JMS API ' leri) OASIS Standard AMQP 1.0 aktarım kanalı iletişim kuralını temel alır. AMQP, gönderenler ve alıcılar arasında iletilerin nasıl gönderildiğini belirtir. Uygulama, Message Broker 'a IBM MQgibi bir ileti gönderdiğinde uygulama gönderen olarak hareket eder. IBM MQ , bir AMQP uygulamasına ileti gönderdiğinde gönderen olarak hareket eder.

AMQP ' nin bazı yararları şunlardır:

- Açık standartlaştırılmış bir iletişim kuralı
- Diğer açık kaynak AMQP 1.0 istemcileriyle uyumluluk
- Birçok açık kaynaklı istemci uygulaması var

Herhangi bir AMQP 1.0 istemcisi bir AMQP kanalına bağlanabilse de, bazı AMQP özellikleri (örneğin, işlemler ya da birden çok oturum) desteklenmez.

Daha fazla bilgi için bkz. AMQP.org web sitesi ve OASIS Standard AMQP 1.0 PDF.

MQ Light ve Apache Qpid JMS API ' leri aşağıdaki ileti sistemi özelliklerine sahiptir:

- En çok bir kez ileti teslimi
- En az bir kez ileti teslimi
- Konu dizesi hedef adreslemesi
- İleti ve hedef dayanıklılığı
- Birden çok abonenin iş yükünü paylaşmasına izin vermek için paylaşılan hedefler
- Askıdaki istemcilerin kolayca çözülmesi için müşteri devralma
- İletilerden önce yapılandırılabilir okuma
- İletilerin yapılandırılabilir onayı

Apache Qpid JMS API 'sinin eksiksiz belgeleri için bkz. Qpid JMS.

İlgili görevler
AMQP kanallarının oluşturulması ve kullanılması
AMQP istemcilerinin güvenliğinin sağlanması

ALW AMQP 1.0 desteği

AMQP kanalları, AMQP 1.0-compliant uygulamaları için bir destek düzeyi sağlar.

AMQP kanalları, AMQP 1.0 protokolünün bir altkümesini destekler. AMQP 1.0 uyumlu istemcilerini bir IBM MQ AMQP kanalına bağlayabilirsiniz. AMQP kanalları tarafından desteklenen tüm ileti sistemi özelliklerini kullanmak için, belirli AMQP 1.0 alanlarının değerini doğru şekilde ayarlamanız gerekir.

Bu bilgiler, AMQP alanlarının nasıl biçimlendirilmesi gerektiğini özetler ve AMQP kanalları tarafından desteklenmeyen AMQP 1.0 belirtiminin özelliklerini listeler.

AMQP 1.0 belirtiminin aşağıdaki özellikleri desteklenmez ya da kullanımları sınırlıdır:

ÇERÇEVE EKLE

V 9.3.0

AMQP kanalları, ATTACH çerçevesindeki yeteneklerin aşağıdakilerden birini içermesini bekler:

```
topic
temporary queue
queue
shared
```

Yetenekler, nesne tipini belirtir ve çoklu yetenekler söz konusu olduğunda, yeteneğin seçilmesinde öncelik sırası konu, geçici kuyruk, kuyruk şeklindedir.

Bir yetenek beklenen bir değer içermiyorsa, varsayılan yetenek konudur. Diğer yetenekler yoksayılır.

Not: Bazı AMQP istemcileri bu yetenekleri ayarlamaz ve yayınlama/abone olma IBM MQ varsayılan davranışını alır. Örneğin, Quarkus Reactive Messaging AMQP 1.0 Connector yalnızca 2.8.0CR1 sürümünden başlayarak yetenekleri ayarlar.

V 9.3.0 AMQP kanalları, ATTACH çerçevesindeki `distribution-Mode` öğesinin bir kaynak ya da hedef için aşağıdakilerden birini içermesini bekler:

- `taşıma`
- `kopyala`

Burada `taşıma` yıkıcı bir get (alma) ve `kopyalama` (kopyalama) tarayıcıyı belirtir.

Not: `distribution-Mode` ayarlanmazsa ya da `copy`dışında bir değere ayarlanırsa, `taşıma` varsayılır.

Bağlantı adları

AMQP kanalları, bir AMQP bağlantısının adının beş biçiminden birini izlemesini bekler:

- Düz bir konu (yayınlama ve abone olma için)
  - İletilerin yayınlanması: Düz konu dizgisi (örneğin, `"/sports/football"`bağlantı adı), `/sports/football` konusunda bir iletinin yayınlanmasına neden olur.
  - İletileri almak için bir konuya abone olma: düz konu dizgisi (örneğin, `"/sports/football"` bağlantı adı, `/sports/football` konusunda bir aboneliğin tanımlanmasına neden olur.
- Özel ayrıntılı konu (abone olmak için)
  - Şu formdaki özel aboneliği açıklayan ayrıntılı konu dizgisi: `"private:`*`topic string`*`"` (örneğin: `"private:/sports/football"`). Davranış, düz konu dizgisiyle aynıdır. `private` bildirimi, belirli bir AMQP istemcisine özgü bir aboneliği, istemciler arasında paylaşılan bir abonelikten ayırır.
- Paylaşılan ayrıntılı konu (abone olmak için)
  - Şu formdaki paylaşılan aboneliği açıklayan ayrıntılı konu dizesi: `"share:`*`share name`*`:`*`topic string`*`"` (örneğin: `"share:bbc:/sports/football"`).
- V 9.3.0 Kuyruk (üretici ve tüketici için noktadan noktaya ileti alışverişi için)
  - Üretici ileti gönderir; kuyruk adı dizgisi, üreticinin kuyrukta ileti göndermesine neden olur.
  - Tüketici ileti alacak; kuyruk adı dizgisi, tüketicinin kuyruktan ileti almasına neden olur.
- V 9.3.0 Boş (geçici bir kuyrukta noktadan noktaya ileti alışverişi için)
  - Üretici, geçici bir kuyrukta ileti gönderir; Boş, üreticinin geçici bir kuyrukta ileti göndermesine neden olur.
  - Tüketicinin geçici bir kuyruktaki iletileri alması; boş olması, bir tüketicinin geçici bir kuyruktan ileti almasına neden olur.

AMQP iletilerinin IBM MQ iletileriyle ve bu iletilerden nasıl eşlendiği hakkında daha fazla bilgi için bkz. “AMQP alanlarının IBM MQ alanlarıyla (gelen iletiler) eşlenmesi” sayfa 659.

Konu dizgileri, paylaşım adları ve istemci tanıtıcıları için uzunluk üst sınırı

Konu dizgisi, paylaşım adı ve istemci tanıtıcısı 10237 bayt içinde bulunmalıdır. Buna ek olarak, istemci tanıtıcısı uzunluğu üst sınırı 256 karakterdir.

Bu uzunluk üst sınırı aşağıdakilerden birine sahip olabileceğiniz anlamına gelir:

- paylaşım adının kısa olması koşuluyla, çok uzun bir konu dizgisi
- uzun paylaşım adı, ancak kısa konu dizgisi

Konteyner Tanıtıcıları

AMQP kanalları, AMQP Open işlevine ilişkin taşıyıcı tanıtıcısının benzersiz bir AMQP istemci tanıtıcısı içermesini bekler. AMQP istemci tanıtıcısı uzunluğu üst sınırı 256 karakterdir ve tanıtıcı alfasayısal karakterler, yüzde işareti (%), eğik çizgi (/), nokta (.) ve alt çizgi (_) içerebilir.

Oturumlar

AMQP kanalları yalnızca tek bir AMQP oturumunu destekler. Birden çok AMQP oturumu yaratmayı deneyen bir AMQP istemcisi bir hata iletisi alır ve kanalla bağlantısı kesilir.

İşlemler

AMQP kanalları AMQP hareketlerini desteklemez. Yeni bir işlemi koordine etmeye çalışan bir AMQP bağlantı çerçevesi ya da yeni bir işlemi bildirmeye çalışan bir AMQP aktarım çerçevesi, bir hata iletisiyle reddedilir.

Teslim durumu

AMQP kanalları, yalnızca Kabul Edildi, Serbest Bırakıldı ya da Değiştirildi düzenleme çerçeveleri için teslim durumunu destekler. Değiştirilmiş bir durum kullanıldığında AMQP kanallarının teslim edilemeyen-here seçeneğini desteklemediğini unutmayın.

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

V 9.3.0 ALW AMQP kanallarında noktadan noktaya destek

IBM MQ AMQP kanalı, kuyruklara ileti göndermek ve kuyruklardan ileti almak için destek sağlar.

Apache Qpid™ JMS kitaplığı gibi AMQP istemcileri, AMQP ekleme çerçevesini gönderirken bir `queue` ya da `temporary-queue` yeteneği ister. Yetenekler, AMQP kanalının nesneyi bir kuyruk, geçici kuyruk ya da konu olarak tanımlamasına olanak sağlar. Bir kuyruk ya da geçici kuyruk yeteneği ya da yeteneklerden herhangi birinin olmaması durumunda, isteğin bir konu için olduğu varsayılır.

IBM MQ AMQP kanalları, aşağıdakiler için kuyruk tipi desteği sağlar:

Kuyruk alma ve gönderme
İletiler bir kuyruğa gönderilebilir ve kuyruktan tüketilebilir. İletileri tüketmek için hem zamanuyumlu hem de zamanuyumsuz kipler desteklenir.

V 9.3.0 Kuyruk göz atma iletisi
Bir kuyruğa ileti koymanın ve kuyruktan ileti almanın yanı sıra, iletilere kuyruktan da göz atılabilir.

Geçici kuyruk desteği
İletiler geçici bir kuyruğa gönderilebilir ve geçici bir kuyruktan tüketilebilir. Geçici kuyruğu yaratmak için kullanılan geçici kuyruk nesnesi de geçici kuyruğu silmek için kullanılıyorsa, geçici kuyruk silme işleminin desteklendiğini unutmayın.

SYSTEM.DEFAULT.MODEL.QUEUE , geçici bir kuyruk yaratılırken kullanılır ve geçici kuyruğun öneki `AMQP.*`olur.

SYSTEM.DEFAULT.MODEL.QUEUE varsayılan olarak geçici bir dinamik kuyruktur; ancak, SYSTEM.DEFAULT.MODEL.QUEUE kuyruğundaki **`Definition type`** özelliğini kullanarak kuyruğu kalıcı bir dinamik kuyruk olarak değiştirebilirsiniz.

Kalıcı dinamik kuyruk
Apache Qpid JMS kitaplığı gibi bir AMQP istemcisi, **`closed`** özniteliği *true*olarak ayarlanmış bir `detach` çerçevesiyle bir istek gönderdiğinde kalıcı dinamik kuyruk silinir.

Önemli:

Qpid JMS davranışı:
Bir Qpid JMS API komutunu çağırmanız gerekir; örneğin, `javax.jms.TemporaryQueue.delete()` yöntemi kullanıldıktan sonra kuyruğu yok eder ve bu işlem kuyrukta bulunan iletileri de temizler.

Böyle bir komut vermezseniz, bağlantı kapatıldığında kuyruk hala var olan iletilerle birlikte kalır.

-

Geçici dinamik kuyruk
AMQP istemcisi bağlantıyı kapattığında geçici bir dinamik kuyruk silinir.

Önemli:

Qpid JMS davranışı:

`javax.jms.TemporaryQueue.delete()` yöntemi gibi bir Qpid JMS API komutunu çağırır, JMS bağlantısını kapatır ya da bağlantı kesmeleri, kuyruk silinir ve iletiler kaybolur.

Bir JMS oturumunun tek başına kapatılması, geçici kuyruk `javax.jms.Session.createTemporaryQueue()` yöntemi kullanılarak yaratılmış olsa da geçici kuyruğun silinmesine neden olmaz.

-

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW AMQP ve IBM MQ ileti alanlarının eşlenmesi

AMQP iletileri, üstbilgi, teslim ek açıklamaları, ileti ek açıklamaları, özellikler, uygulama özellikleri, gövde ve altbilgiden oluşur.

AMQP iletileri aşağıdaki kısımlardan oluşur:

Üstbilgi

İsteğe bağlı üstbilgi, iletinin beş sabit özniteliğini içerir:

- **sürekli** -dayanıklılık gereksinimlerini belirtir
- **priority** -göreli ileti önceliği
- **ttl** -milisaniye cinsinden yaşam süresi
- **first-acquirer** -bu doğruysa, ileti başka bir bağlantı tarafından alınmadı
- **teslim-sayısı** -önceki, başarısız teslim girişimlerinin sayısı.

Teslim-ek açıklamalar

İsteğe Bağlı. Farklı amaçlanan hedef kitleler için iletinin standart olmayan üstbilgi özniteliklerini belirtir. Teslim ek açıklamaları, gönderen eşten alan eşe bilgi iletiyor.

İleti-ek açıklamalar

İsteğe Bağlı. Farklı amaçlanan hedef kitleler için iletinin standart olmayan üstbilgi özniteliklerini belirtir. Mesaj-ek açıklamalar bölümü, altyapıyı hedefleyen ve her teslim adımına yayılması gereken iletinin özellikleri için kullanılır.

Özellikler

İsteğe Bağlı. Bu kısım, MQ ileti tanımlayıcısına eşdeğerdir. Aşağıdaki sabit alanları içerir:

- **ileti-tnt** -uygulama iletisi tanıtıcısı
- **kullanıcı-kimliği** -kullanıcı yaratma kimliği
- **to** -İletinin gönderildiği düğümün adresi
- **subject** -iletinin konusu
- **yanıtla** -Yanıtın gönderileceği düğüm
- **ilinti-tnt** -uygulama ilinti tanıtıcısı
- **content-type** -MIME içerik tipi
- **content-encoding** -MIME içerik tipi. İçerik tipi için değiştirici olarak kullanılır.
- **mutlak-süre bitimi-süresi** -bu iletinin süresi dolmuş olarak değerlendirildiği zaman
- **yaratma-zamanı** -bu iletinin yaratıldığı zaman
- **grup-tnt** -bu iletinin ait olduğu grup
- **grup-sırası** -bu iletinin grup içindeki sıra numarası
- **yanıtlama grubu tanıtıcısı** -yanıt iletisinin ait olduğu grup

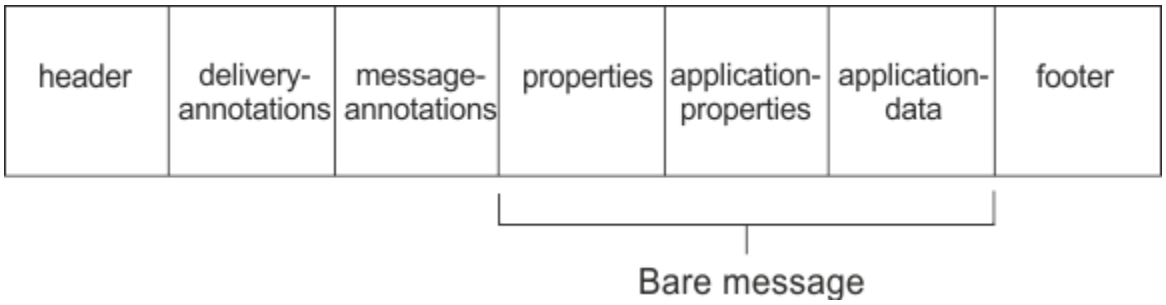
Uygulamalar-özellikler

MQ ileti özelliklerinin eşdeğeri.

Gövde
MQ kullanıcı bilgi yüküne eşdeğerdir.

Altbilgi
İsteğe Bağlı. Altbilgi, yalnızca tüm yalın ileti oluşturulduktan ya da görüldükten sonra hesaplanabilen ya da değerlendirilebilen ileti ya da teslimle ilgili ayrıntılar için kullanılır (örneğin, ileti hash 'leri, HMC ' ler, imzalar ve şifreleme ayrıntıları).

AMQP ileti biçimi aşağıdaki şekilde gösterilmiştir:



Özellikler, uygulama özellikleri ve uygulama verileri kısmı "yalın ileti"olarak bilinir. Bu, gönderen tarafından gönderilen ve sabit olan bir iletidir. Alıcı, üstbilgi, altbilgi, teslim-ek açıklamalar ve ileti-ek açıklamaları da içinde olmak üzere iletinin tamamını görür.

AMQP 1.0 ileti biçiminin tam açıklaması için https://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdfadresindeki OASIS Standardına bakın.

İlgili görevler
AMQP kanallarının oluşturulması ve kullanılması
AMQP istemcilerinin güvenliğinin sağlanması

ALW IBM MQ alanlarını AMQP alanlarıyla (giden iletiler) eşleme

Bir IBM MQ iletisi yayınlandığında ve IBM MQ bunu bir AMQP tüketicisine gönderdiğinde, IBM MQ iletisinin bazı özniteliklerini eşdeğer AMQP iletisi özniteliklerine geçirir.

üstbilgi

Üstbilgi yalnızca üstbilgideki beş alandan biri varsayılan olmayan bir değer içeriyorsa dahil edilir. Üstbilgiye yalnızca varsayılan değeri olmayan alanlar eklenir. Beş üstbilgi alanı, ayarlanmışsa, başlangıçta eşdeğer `mq_amqp.Hdr` özelliğinden türetilir ve aşağıdaki tabloda gösterildiği gibi değiştirilir:

Çizelge 101. Üstbilgi alanı eşlemeleri

Alan	Varsayılan değer	Değer
sürekli	yanlış	`MQMD.Persistence` , `MQPER_PERSISTENT`olarak ayarlandıysa true, tersi durumda false olarak ayarlanır.
öncelik	4	Ayarlanmışsa `mq_amqp.Hdr.Pri`'dan ya da ayarlanmışsa `MQMD.Priority`' dan. İkisi de ayarlanmazsa, 4 olarak ayarlayın.
ttl	geçerli değil	`MQMD.Expiry` milisaniye cinsinden. `MQMD.Expiry` değeri MQEI_UNLIMITED ise, AMQP ttl alanı için maksimum değere ayarlayın
birinci alıcı banka	yanlış	`mq_amqp.Hdr.Fac`' den (ayarlandıysa) ya da tersi durumda false (yanlış) değerini seçin.

Çizelge 101. Üstbilgi alanı eşlemeleri (devamı var)		
Alan	**Varsayılan değer**	**Değer**
teslimat-sayısı	0	`mq_amqp.Hdr.Dct`' dan (ayarlandıysa) ya da 0 'dan (tersi durumda).

teslim-ek açıklama

AMQP kanalı tarafından gerektiği gibi ayarlayın.

ileti-ek açıklama

Dahil değil.

Özellikler

Özellikler ayarlanmışsa, eşdeğer `mq_amqp.Prp` özelliklerinden değiştirilmeden gelir. İleti başlangıçta bir AMQP iletisi değilse (PutApplTipi MQAT_AMQP değilse), aşağıdaki çizelgede açıklandığı gibi bir özellikler kısmı oluşturulur:

Çizelge 102. Özellikler alanı eşlemeleri	
Ad	**Değer**
ileti-tnt	`MQMD.MsgId` ikili olarak ayarlanır.
kullanıcı-kimliği	`MQMD.UserIdentifier`' nin UTF-8 biçimi, ağ bayt sıralamasında ikili olarak ayarlanır.
Bitişi	İletinin gönderildiği kuyruk ya da bir yayın için konu dizgisi.
konu	Ayarlanmadı.
yanıtla	Boş değilse `MQMD.ReplyToQ` , ayarlanmaz.
ilinti-tnt	Boş değilse `MQMD.CorrelId` ikili olarak ayarlanır, tersi durumda ayarlanmaz.
İçerik Tipi	Ayarlanmadı.
içerik kodlaması	Ayarlanmadı.
mutlak-süre bitimi-saati	Ayarlanmadı.
oluşturma-zamanı	Bir zaman damgası oluşturmak için `MQMD.PutDate` ve `MQMD.PutTime` alanları kullanılır.
grup-tnt	Ayarlanmadı.
grup-sırası	Ayarlanmadı.
yanıtlanacak grup tanıtıcısı	Ayarlanmadı.

uygulama özellikleri

"usr" grubundaki tüm IBM MQ özellikleri **application-properties**olarak eklenir.

gövde

AMQP kanalı, IBM MQ bilgi yükünü UTF-8' e dönüştürmek için bir get (alma) işlemi gerçekleştirir.

IBM MQ bilgi yükü bir AMQP iletisi içermiyorsa, IBM MQ bilgi yükü gövdede MQFMT_STRING Biçimine (UTF-8 biçimine dönüştürme başarılı oldu) ya da tek bir ikili veri bölümü olarak ayarlanır.

Bir AMQP biçim iletisi eklenirse, bu gövde olarak ayarlanır. Gövde bir AMQP Sırası ise, AMQP iletisinden önce gelen IBM MQ üstbilgilerinin (ileti tanıtıcısında döndürülen ileti özellikleri dahil değil) başına ikili değer girilir. Tersi durumda, IBM MQ üstbilgileri atılır.

altbilgi

Altbilgi dahil edilmez.

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

İlgili başvurular

MQMD-İleti tanımlayıcı

ALW AMQP alanlarının IBM MQ alanlarıyla (gelen iletiler) eşlenmesi

AMQP kanalı bir ileti aldığında ve IBM MQ' e koyduğunda, AMQP iletisinin bazı özniteliklerini eşdeğer IBM MQ ileti özniteliklerine yatar.

Gelen AMQP iletisi eşlenirken aşağıdaki kısıtlamalar geçerlidir:

- Özellikler bölümündeki `message-id` ya da `correlation-id` alanı bir uuid ya da ulong ise, ileti reddedilir.
- Herhangi bir `message-annotations` , iletinin reddedilmesine neden olur.
- `delivery-annotations` ve `footer` bölümlerine izin verilir, ancak bunlar IBM MQ iletisine yayılmaz.

Aşağıdaki alt kısımlar, bir AMQP iletisinin IBM MQ ifadesini gösterir.

İleti tanımlayıcı

Çizelge 103. AMQP iletisine ilişkin ileti tanımlayıcı

Alan	**Değer**
StrucId	MQMD_STRUC_ID
Sürüm	MQMD_VERSION_1
Rapor	MQRO_NONE
MsgType	MQMT_DATAGRAM
Son kullanma tarihi	AMQP ileti üstbilgisindeki `ttl` alanından alınan değer
Geri bildirim	MQFB_NONE
Kodlama	MQENC_NORMAL
CodedCharSetId	1208 (UTF-8)
Biçim	Bkz. Bilgi Yükü
Öncelik	AMQP ileti üstbilgisindeki `priority` alanından alınan değer. Ayarlanırsa, en çok 9 ile sınırlıdır. Ayarlanmazsa, varsayılan değer olan 4 'ü alır.
Kalıcılık	AMQP ileti üstbilgisindeki `durable` alanı true olarak ayarlanırsa, MQPER_PERSISTAN olarak ayarlayın. Tersi durumda, MQPER_NOT_PERSISTENT olarak ayarlayın.
MagId	Kuyruk yöneticisi benzersiz bir 24 baytlık MsgIdayırır.
Birleştirme	Ayarlanmışsa, AMQP özelliklerindeki `correlation-id` alanından alınan değer. 24 baytlık bir ikili değere ayarlayın. Tersi durumda, MQCI_NONE/ olarak ayarlayın.

Çizelge 103. AMQP iletisine ilişkin ileti tanımlayıcı (devamı var)	
Alan	**Değer**
BackoutCount	0
ReplyToQ	V 9.3.0 Ayarlıysa, AMQP özelliklerindeki `reply-to` alanından alınan değer. Tersi durumda "" olarak ayarlayın.
ReplyToQMgr	""
V 9.3.0 Rapor	AMQP uygulama özelliklerinde ayarlanan JMS IBM Report özelliklerinden türetilen değer.
UserIdentifier	AMQP kanalına bağlanan kimliği doğrulanmış kullanıcının tanıtıcısına ayarlayın
AccountingToken	MQACT_NONE
ApplIdentityVerileri	Onaltılı dizgi. AMQP kanalının MQ bağlantı tanıtıcısının son 8 baytı olarak ayarlayın.
PutApplTipi	MQAT_AMQP
PutApplAdı	
PutDate	Ayarlanmışsa, AMQP özelliklerinin `creation-time` alanından alınan değer. Aksi takdirde geçerli tarihe ayarlanır.
PutTime	Ayarlanmışsa, AMQP özelliklerinin `creation-time` alanından alınan değer. Tersi durumda geçerli saate ayarlayın.
ApplOriginVerileri	""

İleti Özellikleri

İleti özelliklerinin ayarlanmasının iki nedeni vardır:

- AMQP iletisinin bazı kısımlarının, iletinin bilgi yükünü etkilemeden kuyruk yöneticisinden akmasına izin vermek için.
- `application-properties`öğesinin seçilmesine izin vermek için.

Aşağıdaki tabloda AMQP iletisinden ayarlanan özellikler gösterilmektedir:

Çizelge 104. AMQP iletisi özellikleri			
Özellik adı	**MQRFH2 adı**	**Tip**	**Açıklama**
AMQPListener	mq_amqp.Lis	MQTYPE_STRING	AMQP kanalı için tanımlayıcı bir dizgi. İletiyi oluşturmak için kullanılır, böylece ilgili taraflar iletiyi hangi sürümün koyduğunu anlayabilirler (örneğin, sorunları tanılarken hizmet ekibi). Değer, kuyruk yöneticisi tarafından doğrulanmaz ve harici olarak belgelenmemelidir.
AMQPVersion	mq_amqp.Ver	MQTYPE_STRING	AMQP iletisinin sürümü. Yoksa, "1.0" varsayılır. Değer, kuyruk yöneticisi tarafından doğrulanmadı.

Çizelge 104. AMQP iletisi özellikleri (devamı var)

Özellik adı	MQRFH2 adı	Tip	Açıklama
AMQPClient	mq_amqp.Cli	MQTYPE_STRING	API için tanımlayıcı bir dizgi. AMQP iletisini kanala göndermek için kullanılır, böylece ilgili taraflar mesajı hangi sürümün koyduğunu söyleyebilirler (örneğin, sorunları tanılarken hizmet ekibi). Değer, kuyruk yöneticisi tarafından doğrulanmaz ve harici olarak belgelenmemelidir.
AMQPDurable (MQPDurable)	mq_amqp.Hdr.Dur	MQTYPE_BOOLEAN	Ayarlandıysa, AMQP ileti üstbilgisindeki `durable` alanının değeri.
AMQPPriority	mq_amqp.Hdr.Pri	MQTYPE_INT32	Ayarlandıysa, AMQP ileti üstbilgisindeki `priority` alanının değeri.
AMQPTtl	mq_amqp.Hdr.Ttl	MQTYPE_INT64	Ayarlandıysa, AMQP ileti üstbilgisindeki `ttl` alanının değeri.
AMQPFirstAcquirer	mq_amqp.Hdr.Fac	MQTYPE_BOOLEAN	Ayarlandıysa, AMQP ileti üstbilgisindeki `first-acquirer` alanının değeri.
AMQPDeliveryCount	mq_amqp.Hdr.Dct	MQTYPE_INT64	Ayarlandıysa, AMQP ileti üstbilgisindeki `delivery-count` alanının değeri.
AMQPMsgId	mq_amqp.Prp.Mid	MQTYPE_STRING	Dizgi olarak ayarlandıysa, AMQP özelliklerindeki `message-id` alanının değeri.
		MQTYPE_BYTE_STRING	Bayt dizgisi olarak ayarlanırsa, AMQP özelliklerindeki `message-id` alanının değeri.
AMQPUserId	mq_amqp.Prp.Uid	MQTYPE_BYTE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `user-id` alanının değeri.
AMQPTo	mq_amqp.Prp.To	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `to` alanının değeri.
AMQPSubject	mq_amqp.Prp.Sub	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `subject` alanının değeri.
AMQPReplyTo	mq_amqp.Prp.Rto	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `reply-to` alanının değeri.
AMQPCorrelationId	mq_amqp.Prp.Cid	MQTYPE_STRING	Dizgi olarak ayarlanırsa, AMQP özelliklerindeki `correlation-id` alanının değeri.
		MQTYPE_BYTE_STRING	Bayt dizgisi olarak ayarlanırsa, AMQP özelliklerindeki `correlation-id` alanının değeri.
AMQPContentType	mq_amqp.Prp.Cnt	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `content-type` alanının değeri.

Çizelge 104. AMQP iletisi özellikleri (devamı var)			
Özellik adı	**MQRFH2 adı**	**Tip**	**Açıklama**
AMQPContentEncoding	mq_amqp.Prp.Cne	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `content-encoding` alanının değeri.
AMQPAbsoluteExpirySaati	mq_amqp.Prp.Aet	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `absolute-expiry-time` alanının değeri.
AMQPCreationTime	mq_amqp.Prp.Crt	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `creation-time` alanının değeri.
AMQPGroupId	mq_amqp.Prp.Gid	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `group-id` alanının değeri.
AMQPGroupSequence	mq_amqp.Prp.Gsq	MQTYPE_INT64	Ayarlanmışsa, AMQP özelliklerindeki `group-sequence` alanının değeri.
AMQPReplyToGroupId	mq_amqp.Prp.Rtg	MQTYPE_STRING	Ayarlanmışsa, AMQP özelliklerindeki `reply-to-group-id` alanının değeri.

AMQP iletisindeki uygulama özelliklerinin her biri bir IBM MQ ileti özelliği olarak ayarlanır. `application-properties` kısmı, bayt için aynı şekilde yeniden oluşturulmalıdır; bu nedenle, aşağıdaki kısıtlamalar geçerlidir:

- Bir uygulama özelliği MQSETMP geçerlilik denetimi kodu tarafından reddedilirse, reddedilecek ileti. Örneğin:
  - Özellik adı uzunluğu MQ_MAX_PROPERTY_NAME_LENGTH ile sınırlıdır.
  - Özellik adı, Java Tanıtıcıları için Java Dil Belirtimi tarafından tanımlanan kurallara uygun olmalıdır.
  - Özellik adı, ayarlanabilen belgelenmiş JMS özellikleri dışında `JMS` ya da `usr.JMS` ile başlamamalıdır.
  - Özellik adı bir SQL anahtar sözcüğü olmamalıdır.
- U+002E (".") Unicode karakterini içeren bir uygulama özelliği iletinin reddedilmesine neden olur. Özellik, JMS tarafından kullanılan "usr" özellik grubunda ifade edilmelidir.
- Yalnızca boş değerli, boole, byte, short, int, long, float, double, binary ve string özellikleri desteklenir. Başka tipte bir uygulama özelliği, iletinin reddedilmesine neden olur.

V 9.3.0 `application-properties`komutunu kullanarak aşağıdaki JMS özelliklerini ayarlayabilirsiniz:

- JMS_IBM_REPORT_EXCEPTION
- JMS_IBM_REPORT_SÜRE sonu
- JMS_IBM_REPORT_COA
- JMS_IBM_REPORT_COD
- JMS_IBM_REPORT_PAN
- JMS_IBM_REPORT_NAN
- JMS_IBM_REPORT_PASS_MSG_ID
- JMS_IBM_REPORT_PASS_CORREL_ID
- JMS_IBM_REPORT_DISCARD_MSG

Özellik adlarının ve değerlerinin eşdeğer "Sağlayıcıya özgü alanların JMS eşlenmesi" sayfa 153 ayrıntılarıyla tutarlı olduğunu ve geçerli olmayan değerlerin yoksayıldığını unutmayın.

bilgi yükü

- Tek bir ikili veri bölümü olan bir AMQP body için, ikili veriler (AMQP bitleri dışında), MQFMT_NONE biçimiyle IBM MQ bilgi yükü olarak kullanılır.
- Tek dizgi veri bölümü olan bir AMQP body için, dizgi verileri (AMQP bitleri hariç), MQFMT_STRING Biçimi ile IBM MQ bilgi yükü olarak kullanılır.
- Tersi durumda, AMQP body , bilgi yükünü olduğu gibi MQFMT_AMQP Biçimiyle oluşturur.

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW İleti sağlama güvenilirliği

Bu bölümde, MQ Light API ve Apache Qpid JMSile ilgili güvenilirlik özellikleri karşılaştırılıyor.

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW MQ Light ileti güvenilirliği

MQ Light API 'sinin, AMQP uygulamalarına ve AMQP uygulamalarından ileti teslimatı güvenilirliğini denetlemenizi sağlayan dört özelliği vardır.

Bunlar:

- “İleti hizmet kalitesi (QOS)” sayfa 663
- “Abone otomatik onayla” sayfa 664
- “Abonelik yaşam süresi” sayfa 664
- “İleti kalıcılığı” sayfa 664

İleti hizmet kalitesi (QOS)

MQ Light API , iki hizmet niteliği sunar:

- En çok bir kez
- En az bir kez

Yayıncıların ve abonelerin hangi hizmet kalitesini kullanmasını istediğinizi seçebilirsiniz.

MQ Light istemcisi kullanıyorsanız, **qos** seçeneğini *QOS_AT_MOST_ONCE* ya da *QOS_AT_LEAST_ONCE*olarak ayarlayın.

Farklı bir AMQP istemcisi kullanıyorsanız, gerçekleştirmek istediğiniz hizmetin kalitesine bağlı olarak aktarım çerçevesinin (yayıncılar için) **settled** özniteliğini ya da yok etme çerçevesini (aboneler için) *true* ya da *false*olarak ayarlayın.

Hizmet kalitesi, bir iletinin etkileşimin `sending` tarafından ne zaman atılacağını belirler.

Yayıncılık

Bir yayınlayıcı **QOS** *0* ' ı (en çok bir kez) seçerse, yayınlayıcı iletinin kopyasını atmadan önce kuyruk yöneticisinden bir alındı bildirimi beklemez.

Gönderme tamamlanmadan kuyruk yöneticisiyle bağlantı başarısız olursa, ileti aboneler tarafından alınmayabilir.

Bir yayınlayıcı **QOS** *1* ' i (en az bir kez) seçerse, yayıncı kuyruk yöneticisinin iletinin kopyasını atmadan önce iletinin abone kuyruklarına yazıldığını onaymasını bekler.

Gönderme sırasında kuyruk yöneticisiyle bağlantı başarısız olursa, yayınlayıcı kuyruk yöneticisine yeniden bağlandıktan sonra iletiyi yeniden gönderir.

abone olunması

Bir abone **QOS** *0* değerini seçerse, kuyruk yöneticisi iletinin kopyasını atmadan önce aboneden bir alındı bildirimini beklemez.

Abone iletiyi almadan önce aboneyle bağlantı başarısız olursa, bu ileti kaybolabilir.

Bir abone **QOS** *1* ' i seçerse, kuyruk yöneticisi iletinin kopyasını atmadan önce aboneden bir alındı bildirimi bekler. V 9.3.3 Kimden IBM MQ 9.3.3, performansı artırmak için kabul edilen iletiler toplu olarak kaldırılır. Daha fazla bilgi için bkz “Onaylı AMQP iletileri toplu işlerde kuyruktan kaldırılıyor” sayfa 666.

Aboneyle bağlantı, abonenin iletiyi almasından önce başarısız olursa, ileti kuyruk yöneticisi tarafından alıkonur. Kuyruk yöneticisi, kuyruk yöneticisi yeniden bağlandığında aboneye ya da abonelik paylaşılıyorsa başka bir aboneye iletiyi yeniden gönderir.

Abone otomatik onayla

Bir abone **QOS** *1* ' i (en az bir kez) seçerse, kuyruk yöneticisi kopyasını atmadan önce her iletinin alındığını kabul etmelidir. Abone, iletileri ne zaman onaylayacağına karar verebilir.

auto-confirm ayarı *true*olarak ayarlandığında MQ Light istemcisi, istemci ağ üzerinden iletiyi başarıyla aldığında her iletinin teslimini otomatik olarak onaylar.

Bu, bir ağ arızası oluşması durumunda iletinin uygulamaya yeniden teslim edilmesini sağlar. Ancak, uygulama iletiyi kabul eden MQ Light istemcisi ile iletiyi işleyen uygulama arasında başarısız olursa, uygulamanın iletiyi kaybetmesi yine de mümkündür.

auto-confirm ayarı *false*olarak ayarlandığında, MQ Light istemcisi iletinin teslimini otomatik olarak onaylamaz, ancak iletinin ne zaman alınacağına karar vermek için uygulamaya bırakır.

Bu, bir uygulamanın iletinin işlendiğini ve atılabileceğini kuyruk yöneticisine bildirmeden önce veritabanı ya da dosya gibi bir dış kaynakta güncelleme yapmasını sağlar.

Abonelik yaşam süresi

Bir uygulama abone olduğunda, aboneliğin ve bu abonelik için iletilerin saklandığı hedefin uygulama bağlantısı kesildikten sonra var olmaya devam edip etmeyeceğini belirler.

MQ Light abone olma seçeneği **ttl** , uygulamanın bağlantısı kesildikten sonra bir aboneliğin var olmaya devam ettiği süreyi (milisaniye cinsinden) belirtmek için kullanılır. Uygulama bu süreden önce yeniden bağlantı kurarsa, abonelik sürdürülür ve uygulama bu abonelikten gelen iletileri kullanmaya devam edebilir.

Uygulamanın yeniden bağlanması olmadan yaşam süresi geçerse, abonelik kaldırılır ve kalıcı iletiler olsa bile, hedefinde saklanan iletiler kaybolur.

İletileri kaybetmemek önemliyse, bir kesinti sırasında iletilerin kaybolmamasını sağlamak için uygulama için yeterince yüksek bir yaşam süresi değeri belirtmeniz gerekir.

İleti kalıcılığı

İletilerin kalıcılığı, yayınlama ve abone olma uygulamaları ve IBM MQ konu nesnelerinin yapılandırması tarafından denetlenir.

AMQP abonesi **QOS** *0* kullanıyorsa (en çok bir kez) ve sürekli olmayan bir abonelik oluşturuyorsa, AMQP kanalı, aşağıdaki metinde açıklanan diğer seçeneklerden bağımsız olarak her zaman abone kuyruğuna kalıcı olmayan iletiler koyar.

Kuyruk yöneticisi durdurulursa hem aboneliğin hem de iletilerin kaybedileceğini unutmayın.

Bir AMQP yayıncısı AMQP **durable** üstbilgisini *true*olarak ayarlarsa, AMQP kanalı kalıcı iletileri abone kuyruklarına koyar.

Kuyruk yöneticisi herhangi bir nedenle durdurulursa, kuyruk yöneticisi yeniden başlatıldığında iletiler aboneler tarafından kullanılabilir.

durable üstbilgisi ayarlanmazsa, AMQP kanalı, yayınlanan iletilerin kalıcılığını ilgili IBM MQ konu nesnesinin **DEFPSIST** özniteliğine dayalı olarak seçer.

Varsayılan olarak bu, *HAYıR* (kalıcı olmayan) **DEFPSIST** özniteliğini kullanan SYSTEM.BASE.TOPICözniteliğidir.

Uyarı: MQ Light istemcisinin sonraki sürümleri AMQP sürekli üstbilgisinin ayarlanmasını desteklemez.

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW Apache Qpid JMS ileti güvenilirliği

Apache Qpid™ JMS kitaplığının, AMQP uygulamalarına ve AMQP uygulamalarından ileti teslimi güvenilirliğini denetlemenizi sağlayan dört özelliği vardır.

Bunlar aşağıdakiler içindir:

- “Yayıncılık” sayfa 665 V 9.3.0 /noktadan noktaya ileti alışverişi için üretici
  - İleti süre bitimi
  - İleti kalıcılığı
- “abone olunması” sayfa 665
  - Abonelik dayanıklılığı
  - Oturum alındı bildirimi kipi V 9.3.0 (Tüketici noktadan noktaya ileti sistemi için de geçerlidir)

Yayıncılık

İleti süre bitimi

JMS üreticisinin yaşam süresi değerinin ayarlanması, o ileti üreticisi tarafından yayınlanan iletilere verilen süre bitimini etkiler.

Bir JMS üreticisi için yaşam süresi değerinin, iletilerin süresi dolmadan önce tüketilebilecek kadar büyük olduğundan emin olun.

Diğer bir seçenek olarak, etkin kalma süresi değerinin ayarlanmamış olması, iletinin abonelik kuyruğundan süresinin dolmasını önler.

İleti kalıcılığı

JMS ileti üreticisinin teslim kipinin ayarlanması, belirtilen konuda yayınlanan IBM MQ iletisinin kalıcılığını ayarlar.

DeliveryModekullandığınızdan emin olun.Bir kuyruk yöneticisi sona erdirildiğinde ya da bir kesinti olduğunda alıkonması gereken iletiler için KALıCı.

abone olunması

Abonelik dayanıklılığı

AMQP kanalları, JMS yaratma tüketici yöntemlerinin sürekli sürümlerini kullanarak sürekli aboneliklerin yaratılmasını destekler:

- **createDurableConsumer()**
- **createSharedDurableConsumer()**

Oturum alındı bildirimi kipi

Tüketilen bir iletinin IBM MQ abonelik kuyruğundan kaldırılmadan önce tam olarak işlendiğini garanti etmek için, **Session**kullanarak bir JMS oturumu oluşturun.CLIENT_ALINDI bildirimi kipi ve bu iletiyi ve bu oturumda daha önce alınan diğer iletileri onaylamak için **message.acknowledge()** yöntemini kullanın.

İlgili kavramlar

AMQP istemci uygulamalarının geliştirilmesi

AMQP API ' leri için IBM MQ desteği, IBM MQ yöneticisinin bir AMQP kanalı oluşturmasına olanak sağlar. Bu kanal başlatıldığında, AMQP istemci uygulamalarından gelen bağlantıları kabul eden bir kapı numarası tanımlar.

V 9.3.3 Onaylı AMQP iletileri toplu işlerde kuyruktan kaldırılıyor

Bir AMQP uygulaması QOS_AT_LEAST_ONCE (1) ileti teslimini kullanıyorsa, AMQP hizmeti, uygulamaya iletiyi gönderdikten sonra alıkoyduğu iletinin kopyasını atmadan önce uygulamadan bir alındı bildirimi bekler. IBM MQ 9.3.3' den, onaylanan iletiler, tek tek değil, toplu işler halinde kuyruktan kaldırılır ve bu da performansın artmasına neden olur.

Bu görev hakkında

Long Term Support ve Continuous Delivery öncesi IBM MQ 9.3.3için, her ileti kuyruktan ayrı ayrı kaldırılır.

IBM MQ 9.3.3' den iki sistem özelliğini **com.ibm.mq.AMQP.BATCHSZ** ve **com.ibm.mq.AMQP.BATCHINT** kullanarak, daha yüksek performans için toplu işlerde alındı bildirimlerinin işlenmesini ayarlayabilirsiniz:

com.ibm.mq.AMQP.BATCHSZ
Bu öznitelik, AMQP hizmeti iletileri kaldırmadan önce alınacak alındı bildirimi sayısı üst sınırını tanımlar. Sayı, 1-9999 aralığında olabilir. Geçersiz bir sayı ayarlanırsa ya da belirtilen sayı geçerli aralığın dışındaysa, varsayılan değer olan 50 kullanılır.

Toplu iş büyüklüğü, iletilerin aktarılmasını etkilemez. İletiler her zaman tek tek aktarılır, ancak AMQP hizmeti alındı bildirimlerini aldıktan sonra toplu işte kaldırılır. Bir toplu işin gerçek boyutu, **com.ibm.mq.AMQP.BATCHINT**ile belirtilen değerden küçük olabilir. Örneğin, **com.ibm.mq.AMQP.BATCHINT** özniteliği tarafından ayarlanan dönem sona erse bir toplu iş tamamlanır.

com.ibm.mq.AMQP.BATCHINT
Bu öznitelik, AMQP hizmetinin alınan iletileri kuyrukta tuttuğu süreyi milisaniye cinsinden tanımlar. Toplu iş dolu değilse, bu süreden sonra toplu iş temizlenir. 1-999 999 999 arasında bir milisaniye belirleyebilirsiniz. Varsayılan değer 50 'dir. Bu öznitelik için bir değer belirtmezseniz, varsayılan değer olan 50 kullanılır.

Notlar:

1. AMQP hizmetinin bir iletiyi atmadan önce bir onayı bekleyip beklemeyeceği, bir uygulamanın ileti teslimi için kullandığı aşağıdaki iki hizmet niteliğine bağlıdır:
   - QOS for QOS_AT_MOST_ONCE (0)

     Bir AMQP uygulaması bu hizmet kalitesini kullanıyorsa, iletileri kabul etmez, bu nedenle AMQP hizmeti iletileri uygulamaya gönderdikten sonra bir onay beklemeden atar.
   - QOS_AT_LEAST_ONCE (1)

     Bir AMQP uygulaması bu hizmet kalitesini kullanıyorsa, iletileri onaylar, bu nedenle AMQP hizmeti, uygulamadan bir onay alıncaya kadar uygulamaya gönderdikten sonra her iletinin bir kopyasını saklar. Uygulama, iletiyi kabul etmeden önce bağlantıyı keserse ya da bağlantıyı kaybederse, ileti diğer uygulamalar tarafından kullanılabilir duruma getirilir. AMQP hizmeti, onaylanıncaya kadar kuyruktan bir iletiyi kaldırmaz.
2. MQ Appliance **com.ibm.mq.AMQP.BATCHSZ** ve **com.ibm.mq.AMQP.BATCHINT** sistem özellikleri IBM MQ Applianceiçin geçerli değildir. IBM MQ Applianceüzerinde varsayılan 50 değeri kullanılır.

.

Yordam

Toplu işlerde alındı bildirimlerinin işlenmesini ayarlamak için **com.ibm.mq.AMQP.BATCHSZ** ve **com.ibm.mq.AMQP.BATCHINT** sistem özelliklerini kullanın.

IBM MQ 9.3.3' den kuyruk yöneticisi yaratıldığında, `amqp_java.properties` dosyası sistem özellikleri için aşağıdaki varsayılan değerleri içerir:

```
-Dcom.ibm.mq.AMQP.BATCHSIZE=50
-Dcom.ibm.mq.AMQP.BATCHINT=50
```

Tüketilen ileti hızına bağlı olarak, daha yüksek performans için toplu işlerde alındı bildirimlerinin işlenmesini hassas bir şekilde ayarlayabilirsiniz. Geçirilen bir kuyruk yöneticisi, `amqp_java.properties` dosyasında bu özelliklere sahip değil. Bu nedenle, yeni düzeye geçirilen bir kuyruk yöneticisi için ya da özellikler ayarlanmamışsa, varsayılan değerler kullanılır. Eniyilenmiş performans için değerleri ince ayarlamak üzere bu özellikleri ekleyebilirsiniz.

Aşağıdaki koşullardan biri karşılandığında, kabul edilen iletiler toplu olarak kaldırılır:

- Kabul edilen iletilerin sayısı **com.ibm.mq.AMQP.BATCHSZ**' a ulaşır.
- Toplu iş başlatıldıktan sonra **com.ibm.mq.AMQP.BATCHINT** aşıldı.
- Uygulama, önceki iki koşul yerine getirilmeden önce kuyruğun ya da konunun bağlantısını keser ya da kapatır.

ALW IBM MQ ile AMQP istemcileri için topolojiler

IBM MQile çalışacak AMQP istemcilerinizi geliştirmenize yardımcı olacak örnek topolojiler.

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW IBM MQ üzerinden iletişim kuran AMQP istemcileri

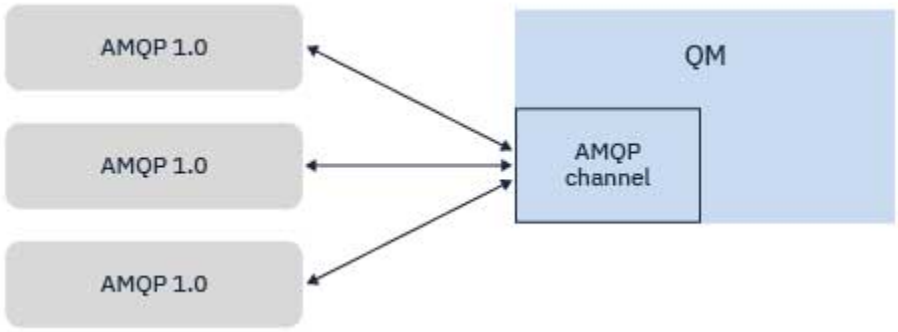
IBM MQ ' i AMQP 1.0ile uyumlu herhangi bir uygulama için ileti alışverişi sağlayıcısı olarak kullanabilirsiniz. Herhangi bir AMQP 1.0 istemcisi bir AMQP kanalına bağlanabilse de, bazı AMQP özellikleri (örneğin, işlemler ya da birden çok oturum) desteklenmez.

Bir ya da daha çok AMQP kanalı tanımlayarak, AMQP 1.0 istemcileri kuyruk yöneticisine bağlanabilir ve bir konu dizgisine ileti gönderebilir. İstemciler, örüntüyle eşleşen iletileri almak için bir konu örüntüsüne de abone olabilirler.

Aşağıdaki senaryoda, ileti gönderen ve alan uygulamalar yalnızca AMQP 1.0 uygulamalarıdır.

Uygulamalar, bir konu dizgisine abone olarak yaratılan hedeflerin kalıcı olup olmayacağını seçebilir; böylece, uygulama kuyruk yöneticisiyle bağlantısını geçici olarak kaybederse iletiler kaybolmaz.

Uygulamalar, iletilerin hedeften temizlenmeden önce ne kadar süreyle tutulacağını da seçebilir.



İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW IBM MQ uygulamalarıyla ileti alışverişi sağlayan AMQP istemcileri

AMQP kanalı tanımlayarak ve başlatarak AMQP 1.0 uygulamaları, var olan MQ uygulamaları tarafından alınan iletileri yayınlayabilir. AMQP kanalı aracılığıyla yayınlanan iletilerin tümü, MQ kuyruklarına değil, MQ konularına gönderilir. MQSUB API çağrısını kullanarak abonelik yaratan bir MQ uygulaması, MQ uygulaması tarafından kullanılan konu dizgisi ya da konu nesnesinin AMQP istemcisi tarafından yayınlanan konu dizgisiyle eşleşmesi koşuluyla, AMQP 1.0 uygulamaları tarafından yayınlanan iletileri alır.

AMQP ileti verileri, öznitelikleri ve özellikleri, MQ uygulaması tarafından alınan MQ iletisinde ayarlanır. AMQP ile MQ ileti eşlemeleri hakkında daha fazla bilgi için bkz. “AMQP alanlarının IBM MQ alanlarıyla (gelen iletiler) eşlenmesi” sayfa 659.

MQ uygulaması kalıcı bir abonelik yarattıysa, AMQP uygulaması tarafından yayınlanan iletiler, aboneliği geri alan kuyrukta saklanır. Daha sonra, uygulama aboneliğini sürdürdüğünde MQ uygulaması iletileri alır. AMQP uygulaması bir iletinin etkin kalma süresini belirtiyorsa ve MQ uygulaması canlanma süresi içinde yeniden bağlanmazsa, iletinin süresi kuyruktan sona ermiştir.

AMQP 1.0 uygulamaları, var olan MQ uygulamaları tarafından yayınlanan iletileri de kullanabilir. Uygulamanın yayınlanan konu dizgisiyle eşleşen bir konu örüntüsüyle abone olması koşuluyla, MQ uygulamaları tarafından bir MQ konusuna ya da konu dizgisine yayınlanan iletiler AMQP 1.0 uygulaması tarafından alınır.

AMQP 1.0 uygulaması abonelik için bir etkin kalma süresi değeri belirtirse ve AMQP uygulaması, etkin kalma süresinden daha uzun bir süre için bağlantıyı keserse, abonelik süresi kuyruk yöneticisinden sona erer ve abonelik kuyruğunda saklanan iletiler kaybolur.

MQMD alanları, ileti özellikleri ve uygulama verileri, AMQP uygulaması tarafından alınan AMQP iletisinde ayarlanır. MQ ile AMQP ileti eşlemeleri hakkında daha fazla bilgi için bkz. “IBM MQ alanlarını AMQP alanlarıyla (giden iletiler) eşleme” sayfa 657.

İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

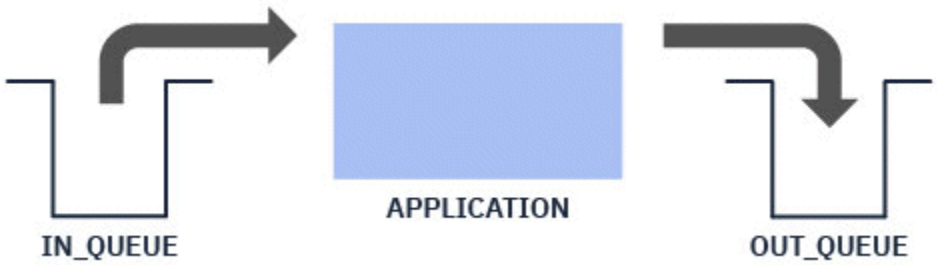
ALW AMQP istemcilerinin IBM MQ kuyruklarındaki uygulamalarla doğrudan etkileşimde bulunmaları için yapılandırılması

V 9.3.0 IBM MQ AMQP uygulaması, yayınlamayı/abone olmayı ve noktadan noktaya olmayı destekler. Noktadan noktaya özelliğini desteklemeyen AMQP istemcileri için, kuyruğa ileti göndermek ya da kuyruktan ileti almak için aşağıdaki adımları kullanın.

Genel Bakış

Örneğin, bir uygulamanın `IN_QUEUE` giriş kuyruğundan ileti aldığını ve bu iletileri bir çıkış kuyruğuna `OUT_QUEUE`koyduğunu varsayın. AMQP istemcilerinin `IN_QUEUE`'e ileti göndermeleri ve `OUT_QUEUE` ' den ileti almaları mümkündür.

Not: Uygulamanın kendisinde değişiklik yapılması gerekmez.



Bir AMQP yayıncısının bir iletiyi kuyruğa koyması için, AMQP istemcisinin yayınlayacağı konu dizgisi için, istenen kuyruğun hedefiyle birlikte bir denetim aboneliği yaratmanız gerekir; bkz. “Uygulamaya ileti gönderiliyor:” sayfa 669.

Bir AMQP abonesinin kuyruktan ileti alabilmesi için, kuyruğu, AMQP istemcisinin abone olduğu konu dizgisini gösteren bir konu nesnesinin hedefiyle aynı adı taşıyan bir diğer ad kuyruğuyla değiştirmeniz gerekir; bkz. “Uygulamadan ileti alınıyor:” sayfa 669

Uygulamaya ileti gönderiliyor:

Uygulama zaten IN_QUEUE ' den ileti alıyor ve bir AMQP istemcisinin iletileri yayınlayabilmesini istiyorsunuz; böylece, bu iletiler uygulama tarafından işlenmek üzere bu kuyruğa gider.

Bunu yapmak için, bu aboneliğin iletileri aldığı konu dizgisinin AMQP istemcisinin yayınlandığı konu dizesi olduğu yeni bir yönetim aboneliği yaratırsınız. Bu aboneliğin hedef kuyruğu, uygulamanın (IN_QUEUE) giriş kuyruğudur.

Bu denetim aboneliği için tanımlanan konu dizgisine yayınlanan iletiler, tanımlı hedefe (bu durumda IN_QUEUE) yöneltilir.

AMQP istemcisinin /application/inkonu dizgisinde yayınlayacağını varsayarak, aşağıdaki MQSC komutunu kullanarak bir yönetim aboneliği APP_INyaratabilirsiniz:

```
DEF SUB(APP_IN) TOPICSTR('/application/in') DEST('IN_QUEUE')
```

Bu nesneyi tanımladığınızda, /application/in ' e yayınlanan tüm iletiler, diğer uygulamalar tarafından bu kuyruğa konan diğer iletilerle aynı şekilde uygulama tarafından alındıkları hedefe IN_QUEUEyöneltilir.

Uygulamadan ileti alınıyor:

Uygulama, diğer istemciler tarafından alınabilecekleri ve işlenebilecekleri OUT_QUEUE' e ileti koyuyor.

Ancak bu durumda, iletilerin AMQP istemcisine teslim edilmesini istersiniz, ancak AMQP istemcileri yalnızca yayınlama/abone olma özelliğini kullanır ve iletileri doğrudan kuyruktan alamaz.

Daha önce ileti alan istemcileri abone olan AMQP istemcisiyle değiştirmek için, AMQP istemcisinin abone olduğu konu dizgisi ve diğer ad kuyruğu için bir konu nesnesi yaratmanız gerekir.

Uyarı: Diğer ad kuyruğunu tanımlarsanız ve herhangi bir AMQP istemcisi abone olmadan önce üretici uygulamayı başlatırsanız, üretici uygulamanın "kuyruk" a (şimdi bir konu) gönderdiği iletiler kaybolur çünkü abone yoktur.

Bu metinde açıklanan değişiklikler, daha önce ileti alan istemcileri yalnızca abone olan AMQP istemcisiyle değiştirir. İleti almak için AMQP ve diğer istemcilerin bir birleşimini kullanmak için daha kapsamlı değişiklikler gerekir.

AMQP istemcisinin /application/outkonu dizgisine abone olduğu varsayılarak, aşağıdaki MQSC komutunu kullanarak APP_OUT konu nesnesini tanımlayabilirsiniz:

```
DEF TOPIC(APP_OUT) TOPICSTR('/application/out')
```

Bu konu nesnesine teslim edilen iletiler, aynı konu dizgisine abone olan AMQP istemcisine teslim edilir.

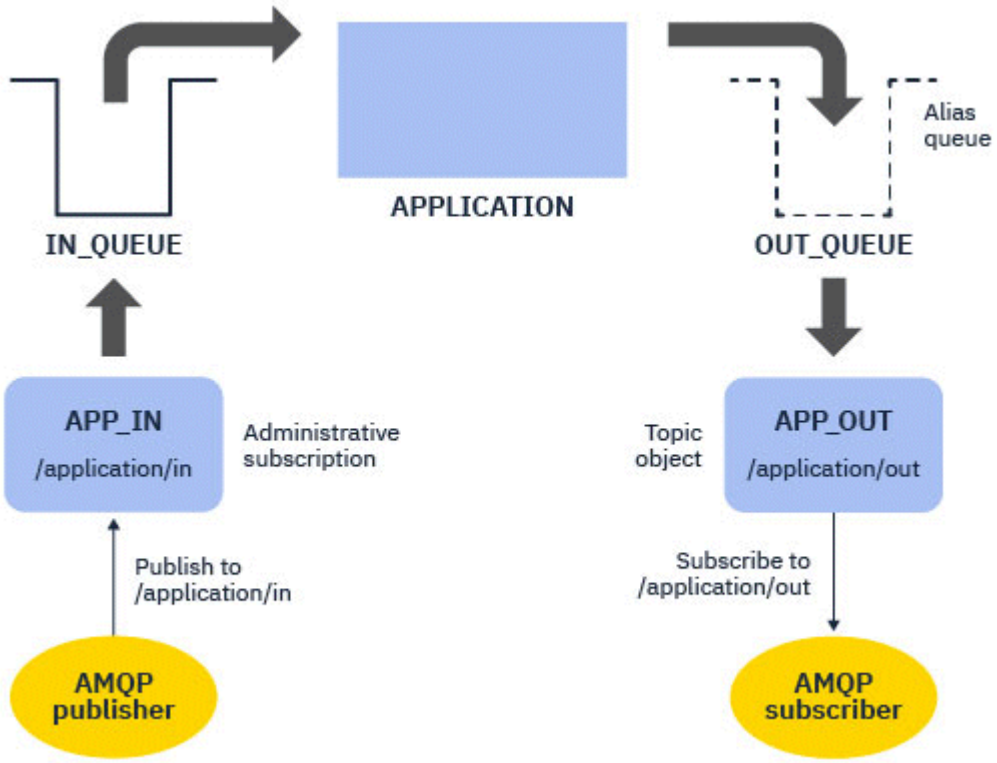
Daha sonra, uygulama tarafından OUT_QUEUE ' e gönderilen iletilerin abone olan istemciye gönderilmesi için bu yeni konu nesnesine teslim edildiğinden emin olmanız gerekir.

Bunu yapmak için, aşağıdaki MQSC komutunu kullanarak var olan OUT_QUEUE kuyruğunu, yeni yaratılan konu nesnesinin hedef tipiyle aynı adı taşıyan bir diğer ad kuyruğuyla değiştirin:

```
DEF QALIAS(OUT_QUEUE) TARGTYPE(TOPIC) TARGET(APP_OUT)
```

Şimdi, uygulama tarafından OUT_QUEUE kuyruğuna konan iletiler alınmak üzere kuyrukta beklemez; bunun yerine bu diğer ad kuyruğunun hedefine teslim edilir (yani, yeni konu nesnesi) APP_OUT.

Bu konu nesnesi `/application/out`ile gösterilen konu dizgisine abone olan AMQP istemcisi, diğer ad kuyruğundan bu konu nesnesine gönderilen iletileri alır.



İlgili görevler

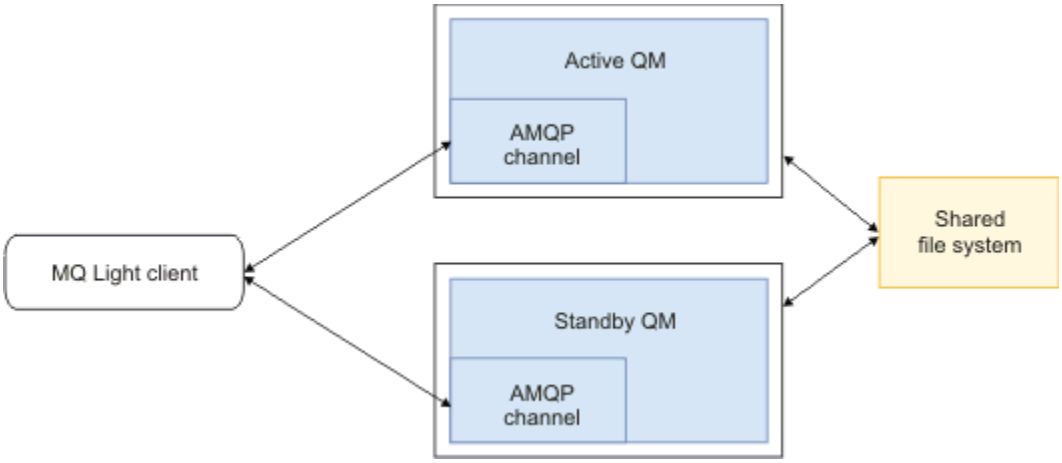
AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW Yüksek kullanılabilirlik için AMQP istemcisinin yapılandırılması

AMQP 1.0 uygulamalarını, bir IBM MQ çok eşgörünümlü kuyruk yöneticisinin etkin yönetim ortamına bağlanacak ve yüksek kullanılabilirlikli (HA) çiftindeki çok eşgörünümlü kuyruk yöneticisinin yedek yönetim ortamına geçmeyecek şekilde yapılandırabilirsiniz. Bunu yapmak için AMQP uygulamasını iki IP adresi ve kapı çiftleriyle yapılandırırsınız.

AMQP istemcisi API 'sini, istemci sunucuyla bağlantısını kaybederse çağrılan özel bir işlevle yapılandırabilirsiniz. İşlev, yedek bir IBM MQ kuyruk yöneticisi ya da özgün IP adresi gibi alternatif bir IP adresine bağlanabilir. Diğer AMQP istemcileri için, istemci birden çok bağlantı uç noktasının yapılandırılmasını destekliyorsa, uygulamayı iki anasistem-kapı çiftiyle yapılandırın ve yedek kuyruk yöneticisine geçmek için AMQP kitaplığı tarafından sağlanan yeniden bağlanma özelliklerini kullanın.



İlgili görevler

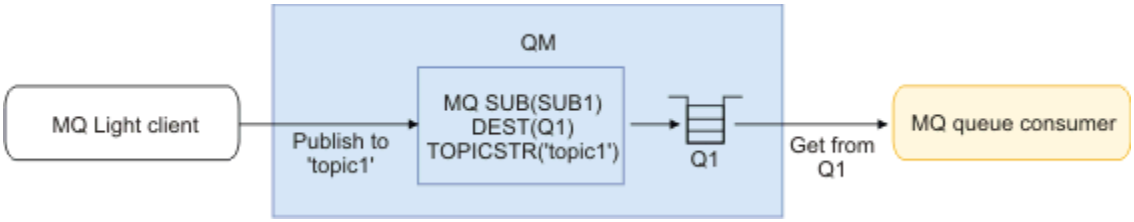
AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW AMQP istemcileri için yayınlama/abone olma yapılandırılıyor

AMQP istemcileri, iletileri var olan bir uygulama tarafından okunan bir IBM MQ kuyruğuna yönlendiren bir IBM MQ aboneliğiyle bir konuyu yayınlayabilir. Bir AMQP 1.0 uygulamasının, kuyruktan okunmak üzere yapılandırılmış var olan bir IBM MQ uygulamasına ileti göndermesini istiyorsanız, kuyruk yöneticisinde denetlenen bir IBM MQ aboneliği tanımlamanız gerekir.

Aboneliği, AMQP uygulaması tarafından kullanılan konu dizgisiyle eşleşen bir konu kalıbını kullanacak şekilde yapılandırın. Abonelik hedefini, IBM MQ uygulamasının iletileri aldığı ya da göz atdığı kuyruğun adına ayarlayın.



İlgili görevler

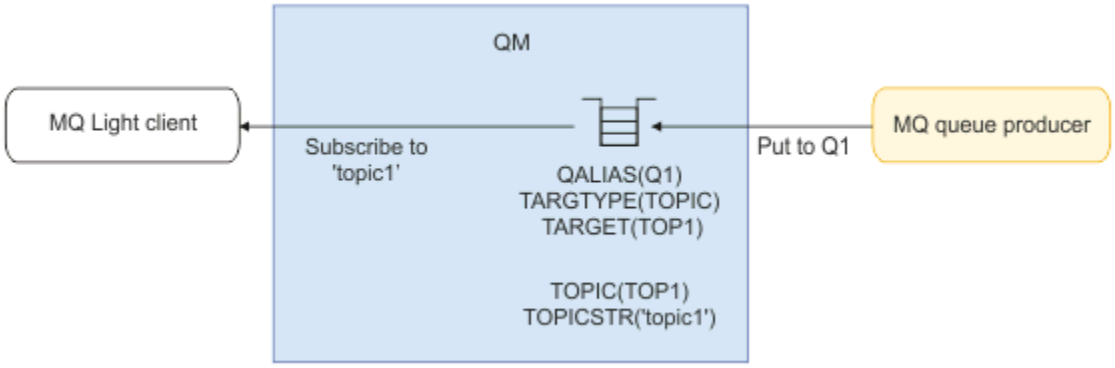
AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW IBM MQ uygulamasından ileti almak için kuyruk diğer adını kullanan AMQP istemcisi

Bir AMQP istemcisi bir konuya abone olabilir ve bir IBM MQ uygulaması tarafından diğer ad kuyruğuna konan iletileri alabilir. Bir AMQP 1.0 uygulamasının, iletileri kuyruğa koyacak şekilde yapılandırılmış var olan bir IBM MQ uygulamasından ileti almasını istiyorsanız, kuyruk yöneticisinde bir kuyruk diğer adı (QALIAS) tanımlamanız gerekir.

Kuyruk diğer adı, IBM MQ uygulamasının koyma işlemi için açtığı kuyruğun adıyla aynı olmalıdır. Kuyruk diğer adı, AMQP uygulaması tarafından abone olunan konu kalıbıyla eşleşen bir konu dizgisine sahip IBM MQ konu nesnesinin temel tipini ve temel nesnesini belirtmelidir.



İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW Bir uygulama sunucusuna istek gönderen ve bir uygulama sunucusundan yanıt alan AMQP istemcisi

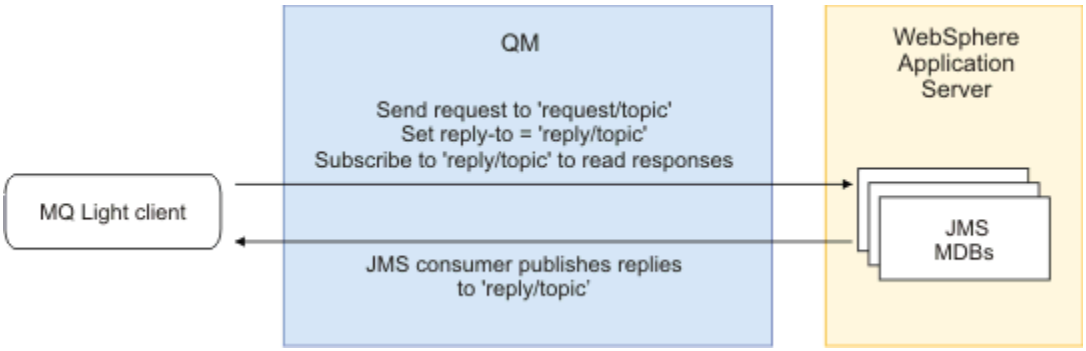
Bir AMQP istemcisi, uygulama sunucusunda çalışan, ileti odaklı bir bean 'e istek gönderebilir ve bir yanıt konusundan gelen yanıtları kullanabilir. IBM MQ , AMQP 1.0 uygulamalarının IBM MQ tarafından yayınlanan iletilerde bir yanıt konusu ayarlamasını destekler. Yanıt özniteliği kümesiyle bir AMQP iletisi yayınlandığında, yanıt alanı değeri, JMS tüketicilerinin alması için bir JMS özelliği olarak ayarlanır. Bu ayar, JMS tüketicilerinin iletiden yanıt konusunu okumasına ve AMQP istemcisine bir yanıt iletisi göndermesine olanak sağlar.

JMS özelliği: **JMSReplyTo**. AMQP yanıtlama dizgisi aşağıdaki tiplerden biri olmalıdır:

- Konu dizgisi. Örneğin, `'reply/topic'`
- `amqp://host:port/[topic-string]`biçiminde bir AMQP adresi URL . Örneğin, `amqp://localhost:5672/reply/topic`

Yanıt alanı olarak bir AMQP adresi URL belirtirseniz, **JMSReplyTo** özelliği ayarlanmadan önce URL ' nin sonundaki konu dizgisi dışındaki her şey kaldırılır.

Bir AMQP yanıt adresinden **JMSReplyTo** özelliğine yapılan eşlemelerle ilgili daha fazla bilgi için bkz. “AMQP alanlarının IBM MQ alanlarıyla (gelen iletiler) eşlenmesi” sayfa 659



İlgili görevler

AMQP kanallarının oluşturulması ve kullanılması

AMQP istemcilerinin güvenliğinin sağlanması

ALW MQ Light ve Apache Qpid JMS uygulamaları arasında birlikte çalışabilirlik

MQ Light ve Apache Qpid JMS uygulamaları benzer şekilde çalışır ve bir konuya abone olurken, aynı adlandırma kuralını izleyen IBM MQ abonelikleri oluşturun.

Özel, paylaşılmayan abonelik

Uygulama tarafından oluşturulan IBM MQ aboneliğinin adı şudur: `:private:<clientid>:<topicstring>`.

Farklı bir istemci tanıtıcısı kullanan bir uygulama, abonelik adı otomatik olarak oluşturulduğundan ve AMQP istemci tanıtıcısını içerdiğinden, başka bir uygulama tarafından oluşturulan aboneliklere erişemez.

Apache Qpid JMS ve MQ Light uygulamalarının her ikisi de özel abonelikler için bu adlandırma kuralını kullanır.

Genel olarak paylaşılan abonelikler

Bir AMQP istemcisi tarafından oluşturulan genel olarak paylaşılan IBM MQ aboneliğinin adı şudur: `:share:<sharename>:<topicstring>`.

Farklı AMQP istemci kimliklerine sahip birden çok uygulama aynı paylaşım adını ve konu dizgisini belirtirse, tek bir aboneliği paylaşırlar ve bu aboneliğe ilişkin iletileri işlemek için birlikte çalışabilirler. Bir abonelikten ileti çeken çalışan uygulamalarının sayısını ölçeklemek istiyorsanız bu örüntüyü kullanabilirsiniz.

Hem Apache Qpid JMS hem de MQ Light uygulamaları, genel olarak paylaşılan abonelikler için bu adlandırma kuralını kullanır. Apache Qpid JMSdurumunda, bunun için JMS bağlantısında bir istemci tanıtıcısı belirtilmemiş olması gerekir.

Apache Qpid JMS kitaplığı otomatik olarak bir AMQP istemci tanıtıcısı oluşturur, ancak bu istemci tanıtıcısı IBM MQ abonelik adlandırması amacıyla kullanılmaz.

Not: Genel olarak paylaşılan abonelikler, tek bir kuyruk yöneticisine kapsamaya devam etmektedir.

Özel paylaşılan abonelikler

Bir AMQP istemcisi tarafından oluşturulan özel olarak paylaşılan IBM MQ aboneliğinin adı şudur: `:privateshare:<clientid>:<sharename>:<topicstring>`.

Tek bir Apache Qpid JMS uygulamasındaki birden çok iş parçacığı aynı paylaşım adını ve konu dizgisini kullanıyorsa ve JMS bağlantısında bir istemci tanıtıcısı yapılandırıldıysa, bu iş parçacıkları aynı IBM MQ abonelik nesnesini paylaşır.

Ancak, diğer Apache Qpid JMS bağlantıları farklı bir istemci tanıtıcısı kullanmaları gerektiğinden aboneliği paylaşamaz.

MQ Light istemcileri, özel paylaşılan abonelikler kavramını desteklemez ve Apache Qpid JMS uygulaması tarafından oluşturulan özel bir paylaşılan abonelikten gelen iletileri kullanamazlar.

IBM MQ JMS abonelikleri

IBM MQ JMS abonelikleri AMQP kanallarından farklı bir adlandırma şeması kullanır. MQ Light ya da Apache Qpid JMS uygulamalarının IBM MQ JMS uygulamalarıyla abonelikleri paylaşması mümkün değildir.

İlgili kavramlar

AMQP istemci uygulamalarının geliştirilmesi

AMQP API ' leri için IBM MQ desteği, IBM MQ yöneticisinin bir AMQP kanalı oluşturmasına olanak sağlar. Bu kanal başlatıldığında, AMQP istemci uygulamalarından gelen bağlantıları kabul eden bir kapı numarası tanımlar.

ALW IBM MQ AMQP dinleyici denetim özellikleri

Çok iş parçacıklı bir uygulamada daha iyi performans için AMQP özellikler dosyasında bir özellik yapılandırarak AMQP hizmetinin kullanması gereken işçi iş parçacıklarının sayısını ayarlayabilirsiniz.

Aşağıdaki özellikler dosyalarında AMQP dinleyici hizmeti özelliklerini yapılandırabilirsiniz:

- Windows Windows sistemlerinde: `amqp_win.properties` .
- Linux AIX AIX and Linux sistemlerinde: `amqp_unix.properties` .

Yapılandırabileceğiniz özellikler şunlardır:

Çizelge 105. AMQP dinleyici hizmeti özellikleri

Özellik	Açıklama
com.ibm.mq.MQXR.Workers	AMQP dinleyici hizmetinin yarattığı sunucu çalışanı iş parçacıklarının sayısı. Bu değer belirlenmezse, varsayılan olarak sistemdeki mantıksal işlemci sayısına eşit olur.
MQIBindType	AMQP hizmetine ilişkin bağ tanımı tipi: FASTPATH, SHAREDya da YALITILMIŞ. Varsayılan değer FASTPATH' dır.

AMQP dinleyici hizmeti, istemci bağlantısı iş yükünü çok sayıda işçi iş parçacıklarında dengeler. AMQP hizmetinin kullanması gereken işçi iş parçacıklarının sayısı **com.ibm.mq.MQXR.Workers** özelliği kullanılarak belirtilebilir.

IBM MQ kuyruk yöneticisi, çok iş parçacıklı bir uygulamada daha iyi performans elde etmek için işçi iş parçacıklarının sayısını ayarlayabilir. Genellikle, işçi iş parçacıklarının sayısı sistemdeki mantıksal işlemci sayısıyla eşleştiğinde en iyi performans elde edilir. Ancak, belirli makine yapılandırmaları ve istemci yükleme özellikleri için her zaman bu durum olmayabilir, bu nedenle işçi iş parçacıklarının sayısı için en uygun değeri bulmak için bir ayarlama öğesi gerekebilir.

Ayarlamadan önce, müşteri uygulamalarınızın ve bunların iş yüklerinin doğasını iyice anladığınızdan emin olun. Uygulamanızın performansının farklı iş parçacığı sayılarıyla ve karşılaştırmalı değerlendirmeyle ölçülmesi, çalışan iş parçacıklarının sayısı için en uygun değerin belirlenmesine yardımcı olacaktır.

Not: MQ Appliance Bu özellikler IBM MQ Applianceiçin geçerli değildir. IBM MQ Applianceüzerinde varsayılan değerler kullanılır.

IBM MQ ile REST uygulamaları geliştirilmesi

İleti göndermek ve almak için REST uygulamaları geliştirebilirsiniz. IBM MQ , platforma ve yeteneğe bağlı olarak farklı REST API ' lerini destekler.

Aşağıdaki seçenekler, içinden ileti göndermeyi ve ileti almayı seçebileceğiniz IBM MQ desteklenen seçeneklerdir IBM MQ:

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

IBM MQ messaging REST API

IBM MQ iletilerini düz metin biçiminde göndermek, almak ve bu iletilere göz atmak için messaging REST API komutunu kullanabilirsiniz. messaging REST API varsayılan olarak etkindir.

Ortak ileti özelliklerini ayarlamak için kullanılabilecek farklı HTTP üstbilgileri için destek sağlanır.

messaging REST API , IBM MQ güvenliğiyle tam olarak bütünleştirilmiştir. messaging REST API' i kullanmak için kullanıcıların kimliklerinin mqweb sunucusunda doğrulanması ve MQWebUser rolünün bir üyesi olması gerekir.

Daha fazla bilgi için bkz. “REST API kullanılarak ileti alışverişi” sayfa 676. Ayrıca bkz. Tutorial: İleti alışverişi REST API 'sini kullanmaya ilişkin Go ve Node.js örneklerini içeren IBM Developer üzerindeki IBM MQ ileti sistemiyle REST API çalışmaya başlayın.

IBM z/OS Bağlantı EE

IBM z/OS Connect EE, CICS ya da IMS işlemleri ve IBM MQ kuyrukları ve konuları gibi var olan z/OS varlıkları üzerinde REST API ' leri oluşturmanıza olanak sağlayan bir z/OS ürünüdür. Var olan z/OS varlığı kullanıcıdan gizlendi. Bu, varlıklarınızı değiştirmeden ya da bunları kullanan var olan uygulamalardan herhangi birini değiştirmeden varlıklarınızı REST ' ye etkinleştirmenizi sağlar.

IBM z/OS Connect EE, REST API ' leri tarafından kullanılan JSON verileri ile birçok ana bilgisayar uygulaması tarafından beklenen COBOL gibi daha geleneksel dil yapıları arasında çeviri yapmak için otomatik veri dönüştürme sağlar.

Eclipse tabanlı IBM z/OS Connect EE API araç takımı, sorgu değiştirgelerinden ve URL yol bölümlerinden yararlanan kapsamlı bir RESTful API oluşturmak için kullanılabilir ve JSON biçimini IBM z/OS Connect EE çalıştırma zamanı boyunca akar.

IBM z/OS Connect EE, IBM MQ hizmet sağlayıcısı aracılığıyla IBM MQ kuyruklarını ve konularını RESTful API ' leri olarak göstermek için kullanılabilir. İki farklı hizmet tipi desteklenir:

- Tek yönlü hizmetler: Bunlar, bir kuyruk ya da konu üzerinde tek bir IBM MQ işleminin gerçekleştirilmesini sağlayan bir REST API 'si sağlar. Tam yapılandırmaya bağlı olarak bir HTTP isteği, bir iletinin bir kuyruğa gönderilmesine ya da bir konuya yayınlanmasına neden olabilir ya da HTTP isteği, bir iletinin kuyruktan yıkıcı bir şekilde alınmasına neden olabilir.
- İki yönlü hizmetler: Bunlar, arka uç istek yanıt biçemi uygulaması tarafından kullanılan bir kuyruk çiftinin üzerinde bir REST API 'si sağlar. Çağıranlar, iki yönlü hizmet için bir HTTP isteği yayınlar. HTTP istek bilgi yükü JSON ' dan geleneksel bir dil yapısına dönüştürülür ve arka uç uygulaması ve yanıt kuyruğuna konan bir yanıt tarafından işlendiği bir istek kuyruğuna yerleştirilir. Bu yanıt hizmet tarafından alınır, geleneksel dil yapısından JSON 'a dönüştürülür ve POST yanıt gövdesi olarak çağırana geri gönderilir.

IBM z/OS Connect EE hakkında daha fazla bilgi için bkz. z/OS Connect EE.

IBM MQ hizmet sağlayıcısına ilişkin daha fazla bilgi için IBM MQ hizmet sağlayıcısını kullanmabaşlıklı konuya bakın.

IBM Integration Bus

IBM Integration Bus , destekledikleri ileti biçimleri ve iletişim kurallarından bağımsız olarak uygulamaları ve sistemleri birbirine bağlamak için kullanılabilen IBM' in önde gelen bütünleştirme teknolojisidir.

IBM Integration Bus her zaman IBM MQ ' i desteklemiştir ve IBM MQüzerinde RESTful arabirimi oluşturmak için kullanılabilecek *HTTPInput* ve *HTTPRequest* düğümlerini ve veritabanları gibi diğer birçok sistemi sağlar.

IBM Integration Bus , IBM MQüzerinde basit bir REST arabirimi sağlamaktan çok daha fazlasını yapmak için kullanılabilir. Yetenekleri, REST APIürününün bir parçası olarak gelişmiş bilgi yükü işleme, bilgi yükü zenginleştirme ve diğer birçok geliştirmeyi sağlamak için kullanılabilir.

Daha fazla bilgi için, XML bilgi yükü bekleyen bir IBM MQ uygulamasının üzerinde bir JSON over REST arabirimini gösteren teknoloji örneğine bakın.

DataPower

DataPower ağ geçidi, IBM MQda dahil olmak üzere çeşitli sistemlere güvenlik, denetim, bütünleştirme ve optimize edilmiş erişim sağlanmasına yardımcı olan tek bir çok kanallı ağ geçididir. Hem donanım hem de sanal form faktörlerinde gelir.

DataPower ' in sağladığı hizmetlerden biri, bir iletişim kuralında giriş alabilen ve farklı bir iletişim kuralında çıkış oluşturabilen çok protokollü bir ağ geçididir. Özellikle DataPower , HTTP(S) verilerini kabul edecek ve bunu bir istemci bağlantısı üzerinden IBM MQ ' e yöneltecek şekilde yapılandırılabilir; bu, IBM MQüzerinde bir REST arabirimi oluşturmak için kullanılabilir. REST arabirimini geliştirmek için dönüştürme gibi diğer DataPower hizmetleri de kullanılabilir.

Daha fazla bilgi için bkz. Multi-Protocol Gateway.

REST API kullanılarak ileti alışverişi

Basit noktadan noktaya iletişim ve yayınlama ileti sistemi gerçekleştirmek için messaging REST API olanağını kullanabilirsiniz. Bir konuya ileti yayınlayabilir, kuyruğa ileti gönderebilir, kuyruktaki iletilere göz atabilir ve kuyruktan yıkıcı bir şekilde ileti alabilirsiniz. Bilgiler, messaging REST API ' e düz metin biçiminde gönderilir ve bu bilgilerden alınır.

Başlamadan önce

Not:

- messaging REST API varsayılan olarak etkindir. Tüm ileti alışverişini önlemek için messaging REST API ' i devre dışı bırakabilirsiniz. messaging REST APIürününü etkinleştirme ya da devre dışı bırakma hakkında daha fazla bilgi için bkz. messaging REST APIürününü yapılandırma.
- messaging REST API , IBM MQ güvenliğiyle bütünleştirilmiştir. messaging REST API' i kullanmak için kullanıcıların kimliklerinin mqweb sunucusunda doğrulanması ve MQWebUser rolünün bir üyesi olması gerekir. Kullanıcının belirtilen kuyruğa ya da konuya erişim yetkisi de olmalıdır. REST APIgüvenliği hakkında daha fazla bilgi için bkz. IBM MQ Console ve REST API güvenlik.
- messaging REST APIile Advanced Message Security (AMS) kullanıyorsanız, tüm iletilerin iletiyi postalayan kullanıcının bağlamı değil, mqweb sunucusunun bağlamı kullanılarak şifrelendiğini unutmayın.
- Bir iletiyi alırken ya da bir iletiye göz atarken yalnızca IBM MQ `MQSTR` ya da `JMS TextMessage` biçimli iletiler desteklenir. Daha sonra, tüm iletiler eşitleme noktası altında yıkıcı bir şekilde alınır ve işlenemeyen iletiler kuyrukta bırakılır. IBM MQ kuyruğu, bu zehirli iletileri başka bir hedefe taşımak için yapılandırılabilir. Daha fazla bilgi için bkz. “IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 225.
- messaging REST API , işlemsel destek içeren iletilerin yalnızca bir kez teslim edilmesini sağlamaz. Bir HTTP POST komutu verilirse ve istemci HTTP yanıtı almadan önce bağlantı başarısız olursa, istemci iletinin belirtilen kuyruğa mı gönderildiğini, yoksa belirtilen konuya mı yayınlandığını hemen bilemez. HTTP DELETE komutu verilirse ve istemci HTTP yanıtı almadan önce bağlantı başarısız olursa, yıkıcı geri alma işleminin geri alınmasının bir yolu olmadığı için bir ileti kuyruktan yıkıcı bir şekilde alınmış ve kaybolmuş olabilir.
- V 9.3.0 IBM MQ 9.3.0içinden, gelen dizelerdeki yeni satırlar artık HTTP POST işlemi tarafından kaldırılmaz. Daha önceki sürümleri kullanan REST uygulamaları, REST API kullanılarak gönderilen ya da yayınlanan iletilerde yeni satırlar kullanmamalıdır; bunlar kaybedilir.

Yordam

- “messaging REST API ile çalışmaya başlama” sayfa 677
- “messaging REST API olanağının kullanılması” sayfa 679
- REST API hata işleme
- REST API keşif
- REST API ulusal dil desteği

İlgili başvurular

İleti alışverişi REST API başvurusu

İlgili bilgiler
Eğitmen: IBM MQ ileti sistemiyle çalışmaya başlayın REST API

messaging REST API ile çalışmaya başlama

messaging REST API ile hızlı bir şekilde çalışmaya başlayın ve cURLkomutunu kullanarak birkaç örnek komutu deneyin.

Başlamadan önce

messaging REST APIürününü kullanmaya başlamanız için bu görevdeki örnekler aşağıdaki gereksinimlere sahip olur:

- Bu örnekler, bir kuyruktaki iletileri koymak ve kuyruktan ileti almak üzere REST istekleri göndermek için cURL ' yi kullanır. Bu nedenle, bu görevi tamamlamak için sisteminizde cURL kurulu olmalıdır.
- Örnekler `QM1`kuyruk yöneticisini kullanır. Aynı adı taşıyan bir kuyruk yöneticisi yaratın ya da sisteminizde var olan bir kuyruk yöneticisini kullanın. Kuyruk yöneticisi, mqweb sunucusuyla aynı makinede olmalıdır.
- Bu görevi tamamlamak için, **dspmqweb** komutunu kullanabilmek üzere belirli ayrıcalıklara sahip bir kullanıcı olmanız gerekir:
  - z/OS z/OS' da **dspmqweb** komutunu çalıştırma ve `mqwebuser.xml` dosyasına yazma erişiminiz olmalıdır.
  - Multi Diğer tüm işletim sistemlerinde ayrıcalıklı kullanıcıolmanız gerekir.

IBM i IBM iüzerinde, komutlar QSHELL içinde çalıştırılmalıdır.

Yordam

1. messaging REST APIiçin mqweb sunucusunun yapılandırıldığından emin olun:
   - mqweb sunucusunu administrative REST API, administrative REST API for MFT, messaging REST APIya da IBM MQ Consoletarafından kullanılmak üzere yapılandırdığınızdan emin olun.mqweb sunucusunu temel bir kayıt dosyasıyla yapılandırma hakkında daha fazla bilgi için bkz. mqweb sunucusu için temel yapılandırma.
   - mqweb sunucusu zaten yapılandırıldıysa, mqweb sunucusu için temel yapılandırmanın5. adımında ileti alışverişini etkinleştirmek için uygun kullanıcıları eklediğinizden emin olun.
     - mqweb sunucusu yapılandırmasında **mqRestMessagingAdoptWebUserContext** `true` olarak ayarlanırsa, messaging REST API kullanıcıları `MQWebUser` rolünün bir üyesi olmalıdır. `MQWebAdmin` ve `MQWebAdminRO` rolleri messaging REST APIiçin geçerli değildir. Kullanıcıların, OAM ya da RACFaracılığıyla ileti alışverişi için kullanılan kuyruklara ve konulara erişme yetkisi de olmalıdır.
     - mqweb sunucusu yapılandırmasında **mqRestMessagingAdoptWebUserContext** `false` olarak ayarlanırsa, mqweb sunucusunu başlatmak için kullanılan kullanıcı kimliğinin OAM ya da RACFaracılığıyla ileti alışverişi için kullanılan kuyruklara erişme yetkisi olmalıdır.
2. z/OS
   z/OS' ta, **dspmqweb** komutunu kullanabilmek için WLP_USER_DIR ortam değişkenini ayarlayın. Şu komutu girerek değişkeni mqweb sunucusu yapılandırmanızı gösterecek şekilde ayarlayın:

   ```
   export WLP_USER_DIR=WLP_user_directory
   ```

   burada *WLP_user_directory* , `crtmqweb`' e geçirilen dizinin adıdır. Örneğin:

   ```
   export WLP_USER_DIR=/var/mqm/web/installation1
   ```

   Daha fazla bilgi için mqweb sunucusu yaratılmasıbaşlıklı konuya bakın.
3. Aşağıdaki komutu girerek REST API URL 'sini belirleyin:

```
dspmqweb status
```

Aşağıdaki adımlardaki örnekler, REST API URL ' inizin varsayılan URL `https://localhost:9443/ibmmq/rest/v2/`olduğunu varsayar. URL 'iniz varsayılana göre farklıysa, aşağıdaki adımlarda URL ' yi değiştirin.

4. `MSGQ`kuyruk yöneticisinde `QM1`bir kuyruk oluşturun. Bu kuyruk ileti alışverişi için kullanılır. Aşağıdaki yöntemlerden birini kullanın:

   - administrative REST APIkullanıcısının `mqsc` kaynağında bir POST isteği kullanın ve `mqadmin` kullanıcısı olarak kimlik doğrulamasını yapın:

     ```
     curl -k https://localhost:9443/ibmmq/rest/v2/admin/action/qmgr/QM1/mqsc -X POST -u
     mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/
     json" --data "{\"type\": \"runCommandJSON\",\"command\": \"define\", \"qualifier\":
     \"qlocal\",\"name\": \"MSGQ\"}"
     ```

   - MQSC komutlarını kullan:

     z/OS z/OSsistemlerinde **runmqsc** komutu yerine bir 2CR kaynağı kullanın. Daha fazla bilgi için bkz. IBM MQ for z/OSüzerinde MQSC ve PCF komutlarını yayınlayabileceğiniz kaynaklar.

     a. Aşağıdaki komutu girerek kuyruk yöneticisi için **runmqsc** komutunu başlatın:

        ```
        runmqsc QM1
        ```

     b. Kuyruğu yaratmak için **DEFINE QLOCAL** MQSC komutunu kullanın:

        ```
        DEFINE QLOCAL(MSGQ)
        ```

     c. Şu komutu girerek **runmqsc** ' den çıkın:

        ```
        end
        ```

5. Grant authority for the user that you added to the `mqwebuser.xml` in step 5 of mqweb sunucusu için temel yapılandırma to access the queue `MSGQ`. `myuser` ' in kullanıldığı kullanıcının yerine koy:

   - z/OS z/OS'ta:

     a. Kullanıcıya kuyruk için erişim ver:

        ```
        RDEFINE MQQUEUE hlq.MSGQ UACC(NONE)
        PERMIT hlq.MSGQ CLASS(MQQUEUE) ID(MYUSER) ACCESS(UPDATE)
        ```

     b. Kuyruktaki tüm bağlamı ayarlamak için mqweb tarafından başlatılan görev kullanıcı kimliğine erişim verin:

        ```
        RDEFINE MQADMIN hlq.CONTEXT.MSGQ UACC(NONE)
        PERMIT hlq.CONTEXT.MSGQ CLASS(MQADMIN) ID(mqwebStartedTaskID) ACCESS(CONTROL)
        ```

   - Multi Diğer tüm işletim sistemlerinde, kullanıcı mqm grubundaysa, yetki zaten verilir. Tersi durumda, aşağıdaki komutları girin:

     a. Aşağıdaki komutu girerek kuyruk yöneticisi için **runmqsc** komutunu başlatın:

        ```
        runmqsc QM1
        ```

     b. Kullanıcıya göz atma, sorma, alma ve kuyruğa alma yetkilerini vermek için **SET AUTHREC** MQSC komutunu kullanın:

        ```
        SET AUTHREC PROFILE(MSGQ) OBJTYPE(QUEUE) +
        PRINCIPAL(myuser) AUTHADD(BROWSE, INQ, GET, PUT)
        ```

c. Şu komutu girerek **runmqsc** ' den çıkın:

```
end
```

6. `message` kaynağında bir POST isteği kullanarak `MSGQ` kuyruk yöneticisinde `QM1` `Hello World!` kuyruğuna içerik içeren bir ileti koyun. `myuser` ve `mypassword`için `mqwebuser.xml` içindeki kullanıcı kimliğinizi ve parolanızı yerine kullanın:

   Temel kimlik doğrulaması kullanılır ve cURL REST isteğinde isteğe bağlı bir `ibm-mq-rest-csrf-token` HTTP üstbilgisi ayarlanır. POST, PATCH ve DELETE istekleri için bu ek üstbilgi gereklidir.

```
curl -k https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/MSGQ/message -X
POST -u myuser:mypassword -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: text/
plain;charset=utf-8" --data "Hello World!"
```

7. `message` kaynağında bir DELETE isteği kullanarak, `MSGQ` kuyruk yöneticisinde `QM1`kuyruğundaki `Hello World!` kuyruğundan iletiyi yıkıcı bir şekilde alın. `myuser` ve `mypassword`için `mqwebuser.xml` içindeki kullanıcı kimliğinizi ve parolanızı yerine kullanın:

```
curl -k https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/MSGQ/message -X DELETE
-u myuser:mypassword -H "ibm-mq-rest-csrf-token: value"
```

   `Hello World!` iletisi döndürülür.

Sonraki adım

- Örnekler, isteği korumak için temel kimlik doğrulamasını kullanır. Bunun yerine, belirteç tabanlı kimlik doğrulamasını ya da istemci tabanlı kimlik doğrulamasını kullanabilirsiniz. Daha fazla bilgi için bkz. REST API ve IBM MQ Consoleile istemci sertifikası kimlik doğrulamasını kullanma ve REST APIile belirteç tabanlı kimlik doğrulamasını kullanma.
- messaging REST API olanağını kullanma ve sorgu değiştirgeleriyle URL oluşturma hakkında daha fazla bilgi edinin: “messaging REST API olanağının kullanılması” sayfa 679.
- messaging REST APIkullandığınızda, başarımı en iyi duruma getirmek için kuyruk yöneticisiyle bağlantılar havuza gönderilir. Havuz boyutu üst sınırını ve havuzdaki tüm bağlantılar kullanımdayken hangi işlemin gerçekleştirildiğini yapılandırabilirsiniz: messaging REST APIyapılandırması.
- Kullanılabilir messaging REST API kaynakları ve kullanılabilir tüm sorgu parametreleri için başvuru bilgilerine göz atın: messaging REST API reference.
- IBM MQ yönetimi için bir RESTful arabirimi olan administrative REST APIdosyasını keşfedin: REST APIkullanarak yönetim.
- Tarayıcı tabanlı bir GUI olan IBM MQ Console' yi keşfedin: IBM MQ Consolekullanarak yönetim.

messaging REST API olanağının kullanılması

messaging REST APIkullandığınızda, IBM MQ iletilerini göndermek ve almak için URL ' lerde HTTP yöntemlerini çağırırsınız. HTTP yöntemi, örneğin POST, URLile gösterilen nesne üzerinde gerçekleştirilecek işlemin tipini gösterir. İşlemle ilgili daha fazla bilgi sorgu değiştirgelerinde kodlanmış olabilir. İşlemin gerçekleştirilmesiyle ilgili bilgiler, HTTP yanıtının gövdesi olarak döndürülebilir.

Başlamadan önce

messaging REST APIprogramını kullanmadan önce şunları göz önünde bulundurun:

- messaging REST APIkullanabilmek için mqweb sunucusuyla kimlik doğrulaması yapmalısınız. HTTP temel kimlik doğrulamasını, istemci sertifikası kimlik doğrulamasını ya da belirteç tabanlı kimlik doğrulamasını kullanarak kimlik doğrulaması yapabilirsiniz. Bu kimlik doğrulama yöntemlerinin nasıl kullanılacağı hakkında daha fazla bilgi için bkz. IBM MQ Console ve REST API güvenlik.
- REST API büyük ve küçük harfe duyarlıdır. Örneğin, kuyruk yöneticisi `qmgr1`olarak adlandırılırsa, aşağıdaki URL ' de HTTP POST işlemi hatayla sonuçlanır.

```
/ibmmq/rest/v2/messaging/qmgr/QMGR1/queue/Q1/message
```

- V 9.3.3 messaging REST APIile uzak bir kuyruk yöneticisine bağlanıyorsanız, kuyruk yöneticisi adı yerine kuyruk yöneticisi bağlantısının benzersiz adını kullanmalısınız.
- IBM MQ nesne adlarında kullanılabilecek karakterlerin tümü doğrudan URLiçinde kodlanamaz. Bu karakterleri doğru kodlamak için uygun URL kodlamasını kullanmanız gerekir:
  - Eğik çizgi %2Folarak kodlanmalıdır.
  - Yüzde işareti %25olarak kodlanmalıdır.
  - Bir dönem %2Eolarak kodlanmalıdır.
  - Bir soru işareti %3Folarak kodlanmalıdır.
- Bir iletiyi alırken ya da göz atarken yalnızca IBM MQ `MQSTR` ve `JMS TextMessage` biçimlenmiş iletiler desteklenir. Daha sonra, tüm iletiler eşitleme noktası altında yıkıcı bir şekilde alınır ve işlenemeyen iletiler kuyrukta bırakılır. IBM MQ kuyruğu, bu zehirli iletileri başka bir hedefe taşımak için yapılandırılabilir. Daha fazla bilgi için bkz. “IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 225.

Bu görev hakkında

Bir IBM MQ kuyruk nesnesinde ileti alışverişi işlemi gerçekleştirmek için REST API komutunu kullandığınızda, öncelikle o nesneyi temsil edecek bir URL oluşturmanız gerekir. Her URL , isteğin gönderileceği anasistem adını ve kapıyı açıklayan bir önekle başlar. URL ' nin geri kalanı, kaynak olarak bilinen belirli bir nesneyi ya da o nesneye giden rotayı tanımlar.

Kaynak üzerinde gerçekleştirilecek ileti alışverişi işlemi, URL ' nin sorgu değiştirgelerine gerek duyup duymadığını tanımlar. Ayrıca, kullanılan HTTP yöntemini ve URL' ye ek bilgilerin gönderilip gönderilmediğini ya da bu yöntemden döndürülüp döndürülmediğini de tanımlar. Ek bilgiler HTTP isteğinin bir kısmını oluşturabilir ya da HTTP yanıtının bir parçası olarak döndürülebilir.

URL' yi oluşturduktan sonra, IBM MQadresine HTTP isteği gönderebilirsiniz. İsteği, seçtiğiniz programlama dilinde yerleşik olan HTTP uygulamasını kullanarak gönderebilirsiniz. İsteği, cURLgibi komut satırı araçlarını ya da bir web tarayıcısını ya da web tarayıcısı eklentilerini kullanarak da gönderebilirsiniz.

Önemli: En az “1.a” sayfa 680 ve “1.b” sayfa 680adımlarını gerçekleştirmeniz gerekir.

Yordam

1. URL' yi oluşturun:
   a) Aşağıdaki komutu girerek URL önekini belirleyin:

   ```
   dspmqweb status
   ```

   Kullanmak istediğiniz URL , `/ibmmq/rest/` sözcük grubunu içerir.

   b) İleti alışverişi için kullanılacak kuyruk ve ilişkili kuyruk yöneticisi kaynaklarını URL yoluna ekleyin.

   İleti sistemi başvurusunda, değişken bölümleri URL ' de *{ }*çevreleyen kaşlı ayraçlarla tanımlanabilir. Ek bilgi için bkz. /messaging/qmgr/{qmgrName}/queue/{queueName}/message.

   Örneğin, *QM1*kuyruk yöneticisiyle ilişkilendirilmiş *Q1* kuyruğuyla etkileşimde bulunmak için, aşağıdaki URLönekini yaratmak üzere URL önekine `/qmgr` ve `/queue` ekleyin:

   ```
   https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message
   ```

   İpucu: V 9.3.3 Kuyruk yöneticisi uzak bir kuyruk yöneticisiyse, kuyruk yöneticisi adı yerine kuyruk yöneticisi için benzersiz adı kullanmalısınız. Uzak kuyruk yöneticisinin messaging REST APIile kullanılabilmesi için önce yapılandırılması gerekir. Daha fazla bilgi için bkz “messaging REST API ile kullanılacak uzak kuyruk yöneticisini ayarlama” sayfa 681.

   c) İsteğe bağlı: URL' ye isteğe bağlı bir sorgu parametresi ekleyin.

   Soru işareti ekle,?, sorgu parametresi, eşittir işareti = ve URLiçin bir değer.

Örneğin, sonraki iletinin kullanılabilir olması için en fazla 30 saniye beklemek için aşağıdaki URL' yi oluşturun:

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?wait=30000
```

d) İsteğe bağlı: URL' ye isteğe bağlı başka sorgu parametreleri ekleyin.

URLadresine bir ve işareti ekleyin ve adım 1c' yi yineleyin.

2. URLüzerinde ilgili HTTP yöntemini çağırın. İsteğe bağlı ileti bilgi yükünü belirtin ve kimlik doğrulaması için uygun güvenlik kimlik bilgilerini sağlayın. Örneğin:
   - Seçtiğiniz programlama dilinin HTTP/REST uygulamasını kullanın.
   - REST istemci tarayıcısı eklentisi ya da cURLgibi bir araç kullanın.

V 9.3.3 messaging REST API ile kullanılacak uzak kuyruk yöneticisini ayarlama

İleti alışverişi için uzak kuyruk yöneticilerine bağlanmak üzere messaging REST API olanağını kullanabilirsiniz. Uzak kuyruk yöneticisine bağlanmadan önce, uzak kuyruk yöneticisi yapılanışını ayarlamanız gerekir. Daha sonra, yapılanış bilgilerinde tanımlanan benzersiz adı kullanarak uzak kuyruk yöneticisine bağlanabilirsiniz.

Başlamadan önce

- mqweb sunucusunu administrative REST API, administrative REST API for MFT, messaging REST APIya da IBM MQ Consoletarafından kullanılmak üzere yapılandırdığınızdan emin olun.mqweb sunucusunu temel bir kayıt dosyasıyla yapılandırma hakkında daha fazla bilgi için bkz. mqweb sunucusu için temel yapılandırma.
- mqweb sunucusu zaten yapılandırıldıysa, mqweb sunucusu için temel yapılandırmanın5. adımında ileti alışverişini etkinleştirmek için uygun kullanıcıları eklediğinizden emin olun. messaging REST API kullanıcıları, `MQWebUser` rolünün bir üyesi olmalıdır. `MQWebAdmin` ve `MQWebAdminRO` rolleri messaging REST APIiçin geçerli değildir.
  - **`mqRestMessagingAdoptWebUserContext`** , mqweb sunucusu yapılandırmasında `true` olarak ayarlanırsa, `MQWebUser` rolündeki kullanıcıların OAM ya da RACFaracılığıyla ileti alışverişi için kullanılan kuyruklara ve konulara erişme yetkisi olmalıdır.
  - mqweb sunucusu yapılanışında **`mqRestMessagingAdoptWebUserContext`** `false` olarak ayarlanırsa, mqweb sunucusunu başlatmak için kullanılan kullanıcı kimliğinin, OAM ya da RACFaracılığıyla ileti alışverişi için kullanılan kuyruklara ve konulara erişme yetkisi olmalıdır.
- messaging REST API ' in uzak kuyruk yöneticilerine bağlanacak şekilde yapılandırıldığından emin olun. Daha fazla bilgi için bkz. messaging REST APIiçin bağlantı kipini yapılandırma.

Bu görev hakkında

messaging REST APIkomutunu kullanarak uzak kuyruk yöneticilerine bağlanabilirsiniz. Uzak kuyruk yöneticisi, başka bir sistemde kuyruk yöneticisi, başka bir kuruluşta kuyruk yöneticisi ya da mqweb sunucusuyla aynı kuruluşta kuyruk yöneticisi olabilir.

Uzak bir kuyruk yöneticisine bağlanmak için aşağıdaki yapılandırma adımlarını tamamlamanız gerekir:

- Bir sunucu bağlantısı kanalı ve dinleyici yapılandırın.
- Kuyruk yöneticisine erişmek için uygun bir kullanıcıya yetki verin.
- Kuyruk yöneticisine ilişkin bağlantı bilgilerini içeren bir CCDT dosyası yaratın.
- **`setmqweb remote`** komutunu kullanarak messaging REST API ' e bağlantı bilgilerini ekleyin.

Daha sonra, kuyruk yöneticisi adı yerine URL kaynağında benzersiz adı belirterek uzak kuyruk yöneticisini kullanabilirsiniz.

Uzak kuyruk yöneticilerinizi bir kuyruk yöneticisi grubunun parçası olarak da yapılandırabilirsiniz. Daha fazla bilgi için bkz "İleti alışverişi REST API ile kullanılacak bir kuyruk yöneticisi grubu ayarlama" sayfa 684.

Yordam

1. Uzak kuyruk yöneticisinde, kuyruk yöneticisiyle uzak bağlantılara izin vermek için bir sunucu bağlantısı kanalı yaratın. Komut satırında **DEFINE CHANNEL** MQSC komutunu kullanarak sunucu bağlantısı kanalları yaratabilirsiniz.
   Örneğin, QM1kuyruk yöneticisi için bir sunucu bağlantısı kanalı `QM1.SVRCONN` yaratmak üzere şu komutu girin:

   ```
   DEFINE CHANNEL(QM1.SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
   ```

   DEFINE CHANNEL ve kullanılabilir seçenekler hakkında daha fazla bilgi için bkz. DEFINE CHANNEL.
2. Uygun bir kullanıcının kuyruk yöneticisine erişme yetkisine sahip olduğunu doğrulayın. Bu kullanıcının, ileti alışverişi için kullandığınız tüm kuyruklara ya da konulara erişim yetkisi de olmalıdır. Kullanıcının kuyruk yöneticisi üzerinde `connect`, `inquire`, `alternate user`ve `set context` yetkisi olması gerekir. UNIX, Linux, and Windows üzerinde, komut satırındaki **setmqaut** denetim komutunu kullanın. z/OSüzerinde, yetkili kullanıcıya kuyruk yöneticisine erişim vermek için RACF profillerini tanımlayın. Örneğin, UNIX, Linux, and Windowsişletim sistemlerinde bir kullanıcıya (`exampleUser`) QM1kuyruk yöneticisine erişim yetkisi vermek için aşağıdaki komutu girin:

   ```
   setmqaut -m QM1 -t qmgr -p exampleUser +connect +inq +altusr +setall
   ```

   Hangi kullanıcının yetkilendirilmesi gerektiğine ilişkin daha fazla bilgi için bkz. "messaging REST API tarafından kullanılan güvenlik birincil kullanıcısının belirlenmesi" sayfa 687.
3. ALW
   Uzak kuyruk yöneticisinde dinleyici yoksa, komut satırında **DEFINE LISTENER** MQSC komutunu kullanarak gelen ağ bağlantılarını kabul edecek bir dinleyici yaratın.
   Örneğin, uzak kuyruk yöneticisi QM1için 1414 numaralı kapıda bir dinleyici `REMOTE.LISTENER` yaratmak üzere şu komutu girin:

   ```
   runmqsc QM1
   DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
   end
   ```
4. Komut satırında **START LISTENER** MQSC komutunu kullanarak dinleyicinin çalıştığını doğrulayın:

   ALW Örneğin, QM1kuyruk yöneticisine ilişkin `REMOTE.LISTENER` dinleyicisini başlatmak için AIX, Linux, and Windows üzerinde şu komutu girin:

   ```
   runmqsc QM1
   START LISTENER(REMOTE.LISTENER)
   end
   ```

   z/OS Örneğin, z/OS' da dinleyiciyi başlatmak için şu komutu girin:

   ```
   runmqsc QM1
   START LISTENER TRPTYPE(TCP) PORT(1414)
   end
   ```

   z/OSüzerinde bir dinleyici başlatmadan önce kanal başlatıcı adres alanı başlatılmalıdır.
5. messaging REST API ' i barındıran mqweb sunucusunun çalıştığı sistemde, kuyruk yöneticisi bağlantı bilgilerini içeren bir JSON CCDT dosyası yaratın ya da güncelleyin.

   CCDT dosyası `name`, `clientConnection`ve `type` bilgilerini içermelidir. İsteğe bağlı olarak `transmissionSecurity` bilgileri gibi ek bilgiler ekleyebilirsiniz. Tüm CCDT kanal özniteliği tanımlamalarına ilişkin ek bilgi için CCDT kanal özniteliği tanımlamalarının tam listesibaşlıklı konuya bakın.

Aşağıdaki örnek, uzak kuyruk yöneticisi bağlantısı için temel bir JSON CCDT dosyasını göstermektedir. Kanalın adını, “1” sayfa 682. adımda oluşturulan örnek sunucu-bağlantı kanalı ile aynı ada ayarlar. Bağlantı kapısı, dinleyici tarafından kullanılan kapıyla aynı değere ayarlanır. Bağlantı anasistemi, uzak kuyruk yöneticisinin (QM1) çalıştığı sistemin anasistem adına ayarlanır.

```
{
  "channel": [{
      "name": "QM1.SVRCONN",
      "clientConnection": {
        "connection": [{
            "host": "example.com",
            "port": 1414
          }],
        "queueManager": "QM1"
      },
      "type": "clientConnection"
    }]
}
```

6. messaging REST API' i barındıran mqweb sunucusunu çalıştıran kuruluşta, uzak kuyruk yöneticisi bilgilerini mqweb sunucusu yapılandırmasına eklemek için **setmqweb remote** komutunu kullanın.

   En az aşağıdaki parametreleri belirtmeniz gerekir:

   - **-qmgrName**, burada kuyruk yöneticisinin adını belirtirsiniz.
   - **-ccdtURL**, burada kuyruk yöneticisi için CCDT URL ' yi belirtirsiniz.
   - **-uniqueName**, burada kuyruk yöneticisi için benzersiz bir ad belirtirsiniz. Benzersiz ad, aynı ada sahip olabilecek uzak kuyruk yöneticilerini ayırt etmek için kullanılır ve bu nedenle başka bir kuyruk yöneticisini tanımlamak için var olmaması gerekir.

   Uzak kuyruk yöneticisi bağlantısı için kullanılacak kullanıcı adı ve parola ya da güvenli depo ve anahtar deposu ayrıntıları gibi diğer bazı seçenekleri de belirtebilirsiniz. **setmqweb remote** komutuyla belirtilebilecek parametrelerin tam listesi için bkz. setmqweb remote.

   Örneğin, QM1uzak kuyruk yöneticisini örnek CCDT dosyasıyla eklemek için şu komutu girin:

```
setmqweb remote add -uniqueName "RemoteQM1" -qmgrName "QM1" -ccdtURL "c:\myccdts\ccdt.json"
```

Sonuçlar

Uzak kuyruk yöneticisi, kuyruk yöneticisi adı yerine URL kaynağındaki benzersiz ad kullanılarak messaging REST API ile birlikte kullanılabilir.

Örnek

Aşağıdaki örnek, QM1kuyruk yöneticisi için uzak kuyruk yöneticisi bağlantısını ayarlar. IBM MQ Console , exampleUseradlı kullanıcıya verilen yetkiye dayalı olarak kuyruk yöneticisini denetleme yetkisine sahiptir. Kuyruk yöneticisi bağlantısını yapılandırmak için **setmqweb remote** kullanıldığında bu kullanıcının kimlik bilgileri IBM MQ Console ' e sağlanır.

1. QM1 uzak kuyruk yöneticisinin bulunduğu sistemde, bir sunucu bağlantısı kanalı ve bir dinleyici yaratılır. Dinleyici başlatılır ve `exampleUser` kullanıcısının kuyruk yöneticisine bağlanması ve ileti alışverişi için kullanılan bir kuyruğa erişmesi için yetki verilir:

```
runmqsc QM1
#Define the server connection channel that will accept connections from the Console
DEFINE CHANNEL(QM1.SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
# Define the listener to use for the connection from the Console
DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
# Start the listener
START LISTENER(REMOTE.LISTENER)
end

#Set mq authorization for exampleUser to access the queue manager and a queue for messaging
setmqaut -m QM1 -t qmgr -p exampleUser +connect +inq +setall +dsp
setmqaut -m QM1 -t queue -p exampleUser -n EXAMPLEQ +put +get +browse +inq
```

2. mqweb sunucusunun çalıştığı sistemde, aşağıdaki bağlantı bilgileriyle bir `QM1_ccdt.json` dosyası yaratılır:

```
{
  "channel": [{
      "name": "QM1.SVRCONN",
      "clientConnection": {
        "connection": [{
            "host": "example.com",
            "port": 1414
          }],
        "queueManager": "QM1"
      },
      "type": "clientConnection"
    }]
}
```

3. mqweb sunucusunun çalıştığı sistemde, QM1 kuyruk yöneticisine ilişkin bağlantı bilgileri mqweb sunucusuna eklenir. `exampleUser` kimlik bilgileri bağlantı bilgilerinde bulunur:

```
setmqweb remote add -uniqueName "MACHINEAQM1" -qmgrName "QM1" -ccdtURL
"c:\myccdts\QM1_ccdt.json" -username "exampleUser" -password "password"
```

4. messaging REST API , uzak kuyruk yöneticisine QM1 , URLkaynağındaki kuyruk yöneticisi adı yerine kuyruk yöneticisi bağlantısına ilişkin benzersiz adı kullanarak bağlanabilir:

```
curl -k https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MACHINEAQM1/queue/EXAMPLEQ/
message -X POST -u myuser:mypassword -H "ibm-mq-rest-csrf-token: value" -H "Content-Type:
text/plain;charset=utf-8" --data "Hello World!"
```

V 9.3.3 İleti alışverişi REST API ile kullanılacak bir kuyruk yöneticisi grubu ayarlama

İleti alışverişi için kuyruk yöneticisi gruplarına bağlanmak üzere messaging REST API kullanabilirsiniz. Bir kuyruk yöneticisi grubuna bağlanmadan önce, gruba ilişkin uzak kuyruk yöneticisi yapılanışını ayarlamanız gerekir. Daha sonra, yapılanış bilgilerinde tanımlanan benzersiz adı kullanarak kuyruk yöneticisi grubuna bağlanabilirsiniz.

Başlamadan önce

- mqweb sunucusunu administrative REST API, administrative REST API for MFT, messaging REST APIya da IBM MQ Consoletarafından kullanılmak üzere yapılandırdığınızdan emin olun.mqweb sunucusunu temel bir kayıt dosyasıyla yapılandırma hakkında daha fazla bilgi için bkz. mqweb sunucusu için temel yapılandırma.
- mqweb sunucusu zaten yapılandırıldıysa, mqweb sunucusu için temel yapılandırmanın5. adımında ileti alışverişini etkinleştirmek için uygun kullanıcıları eklediğinizden emin olun. messaging REST API kullanıcıları, `MQWebUser` rolünün bir üyesi olmalıdır. `MQWebAdmin` ve `MQWebAdminRO` rolleri messaging REST APIiçin geçerli değildir.
  - mqweb sunucusu yapılandırmasında **`mqRestMessagingAdoptWebUserContext`** `true` olarak ayarlanırsa, `MQWebUser` rolündeki kullanıcıların ileti alışverişi için kullanılan kuyruklara ve konulara erişme yetkisi olmalıdır. Bu kullanıcılara OAM ya da RACFaracılığıyla yetki verebilirsiniz.
  - mqweb sunucusu yapılanışında **`mqRestMessagingAdoptWebUserContext`** `false` olarak ayarlanırsa, mqweb sunucusunu başlatan kullanıcı kimliğinin ileti alışverişi için kullanılan kuyruklara ve konulara erişme yetkisi olmalıdır. Bu kullanıcıya OAM ya da RACFaracılığıyla yetki verebilirsiniz.
- messaging REST API ' in uzak kuyruk yöneticilerine bağlanacak şekilde yapılandırıldığından emin olun. Daha fazla bilgi için bkz. messaging REST APIiçin bağlantı kipini yapılandırma

Bu görev hakkında

Kuyruk yöneticisi grubu, uygulamaları grup içindeki herhangi bir kuyruk yöneticisine bağlamanızı sağlar. Grup, bir istemci kanal tanımlama çizelgesinde (CCDT) bir bağlantı kümesi olarak tanımlanır. MQCONN ya

da MQCONNX çağrısı kullandığınızda, kuyruk yöneticisi adına bir yıldız işareti koyarak gruba gönderme yaparsınız. messaging REST APIile, kuyruk yöneticisi grubuyla ilişkili benzersiz adı kullanarak gruba gönderme yapabilirsiniz. Benzersiz ad, kuyruk yöneticisi adı yerine URL kaynağında bulunur. Kuyruk yöneticisi gruplarıyla ilgili daha fazla bilgi için bkz. “CCDT ' deki kuyruk yöneticisi grupları” sayfa 885.

Uzak kuyruk yöneticilerinizi ayrı ayrı da yapılandırabilirsiniz. Daha fazla bilgi için bkz “messaging REST API ile kullanılacak uzak kuyruk yöneticisini ayarlama” sayfa 681.

Yordam

1. Gruptaki uzak kuyruk yöneticilerinin her birinde, kuyruk yöneticisiyle uzak bağlantılara izin vermek için bir sunucu bağlantısı kanalı yaratın. Sunucu bağlantısı kanalları yaratmak için komut satırında **DEFINE CHANNEL** MQSC komutunu kullanabilirsiniz.
   Örneğin, QM1kuyruk yöneticisi için bir sunucu bağlantısı kanalı `QM1.SVRCONN` yaratmak üzere şu komutu girin:

   ```
   DEFINE CHANNEL(QM1.SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
   ```

   DEFINE CHANNEL ve kullanılabilir seçenekler hakkında daha fazla bilgi için bkz. DEFINE CHANNEL.
2. Gruptaki uzak kuyruk yöneticilerinin her birinde, uygun bir kullanıcının kuyruk yöneticisine erişme yetkisine sahip olduğundan emin olun. Bu kullanıcının, ileti alışverişi için kullandığınız tüm kuyruklara ya da konulara erişim yetkisi de olmalıdır. Kullanıcının kuyruk yöneticisi üzerinde `connect`, `inquire`, `alternate user`ve `set context` yetkisi olması gerekir. UNIX, Linux, and Windows üzerinde, komut satırındaki **setmqaut** denetim komutunu kullanın. z/OSüzerinde, yetkili kullanıcıya kuyruk yöneticisine erişim vermek için RACF profillerini tanımlayın.
   Örneğin, UNIX, Linux, and Windowsişletim sistemlerinde bir kullanıcıya (`exampleUser`) QM1: kuyruk yöneticisine erişim yetkisi vermek için şu komutu girin:

   ```
   setmqaut -m QM1 -t qmgr -p exampleUser +connect +inq +altusr +setall
   ```

   Hangi kullanıcının yetkilendirilmesi gerektiğine ilişkin daha fazla bilgi için bkz. “messaging REST API tarafından kullanılan güvenlik birincil kullanıcısının belirlenmesi” sayfa 687.
3. ALW
   Gruptaki uzak kuyruk yöneticilerinin her birinde dinleyici yoksa, gelen ağ bağlantılarını kabul edecek dinleyiciler yaratın. Dinleyici yaratmak için komut satırında **DEFINE LISTENER** MQSC komutunu kullanabilirsiniz.
   Örneğin, uzak kuyruk yöneticisi QM1için 1414 numaralı kapıda bir dinleyici `REMOTE.LISTENER` yaratmak üzere şu komutu girin:

   ```
   runmqsc QM1
   DEFINE LISTENER(REMOTE.LISTENER) TRPTYPE(TCP) PORT(1414)
   end
   ```
4. Gruptaki uzak kuyruk yöneticilerinin her birinde, dinleyicinin komut satırında **START LISTENER** MQSC komutunu kullanarak çalıştığından emin olun.

   ALW Örneğin, QM1kuyruk yöneticisine ilişkin `REMOTE.LISTENER` dinleyicisini başlatmak için AIX, Linux, and Windows üzerinde şu komutu girin:

   ```
   runmqsc QM1
   START LISTENER(REMOTE.LISTENER)
   end
   ```

   z/OS Örneğin, z/OS' da dinleyiciyi başlatmak için şu komutu girin:

   ```
   runmqsc QM1
   START LISTENER TRPTYPE(TCP) PORT(1414)
   end
   ```

   z/OSüzerinde bir dinleyici başlatmadan önce kanal başlatıcı adres alanı başlatılmalıdır.

5. messaging REST API ' u barındıran mqweb sunucusunun çalıştığı sistemde bir JSON CCDT dosyası yaratın. Bu JSON dosyası, gruptaki her kuyruk yöneticisi için bağlantı bilgilerini içerir.

   CCDT dosyası, her kuyruk yöneticisi bağlantısına ilişkin `name`, `clientConnection`ve `type` bilgilerini içermelidir. İsteğe bağlı olarak `transmissionSecurity` bilgileri gibi ek bilgiler ekleyebilirsiniz. Tüm CCDT kanal özniteliği tanımlamalarına ilişkin ek bilgi için CCDT kanal özniteliği tanımlamalarının tam listesibaşlıklı konuya bakın.

   Aşağıdaki örnek, iki kuyruk yöneticisi bağlantısı için temel bir JSON CCDT dosyasını göstermektedir. İlk bağlantı QM1kuyruk yöneticisi içindir. `QM1.SVRCONN`sunucu bağlantısı kanalına, `1414`kapısında bir dinleyiciye sahiptir ve `QM1.example.com`anasisteminde çalışır. İkinci bağlantı, QM2kuyruk yöneticisi içindir. `QM2.SVRCONN`sunucu bağlantısı kanalına, `1415`kapısında bir dinleyiciye sahiptir ve `QM2.example.com`anasisteminde çalışır. Ancak, bağlantılar QMGRP kuyruk yöneticisi grubunun bir parçası olduğundan, her iki bağlantıya ilişkin **queueManager** alanı kuyruk yöneticisi grubunun adına ayarlanır.

```
{
  "channel": [{
      "name": "QM1.SVRCONN",
      "clientConnection": {
        "connection": [{
            "host": "QM1.example.com",
            "port": 1414
          }],
        "queueManager": "QMGRP"
      },
      "type": "clientConnection"
   }],
   "channel": [{
      "name": "QM2.SVRCONN",
      "clientConnection": {
        "connection": [{
            "host": "QM2.example.com",
            "port": 1415
          }],
        "queueManager": "QMGRP"
      },
      "type": "clientConnection"
    }]
}
```

6. messaging REST API' u barındıran mqweb sunucusunu çalıştıran kuruluştan, kuyruk yöneticisi grubunu mqweb sunucusu yapılandırmasına eklemek için **setmqweb remote** komutunu kullanın.

   En az aşağıdaki parametreleri belirtmeniz gerekir:

   - **-qmgrName**, burada kuyruk yöneticisi grubu için grup adını belirtirsiniz.
   - **-ccdtURL**, burada kuyruk yöneticileri için CCDT URL değerini belirtirsiniz.
   - **-uniqueName**, burada kuyruk yöneticisi grubunu tanıtmak için benzersiz bir ad belirtirsiniz.
   - **-group**, kuyruk yöneticisi bilgilerini bir gruba ilişkin olarak ayarlamak için.

   Bağlantı için kullanılacak kullanıcı adı ve parola ya da güvenli depo ve anahtar deposu ayrıntıları gibi diğer birkaç seçeneği de belirtebilirsiniz. **setmqweb remote** komutuyla belirtilebilecek parametrelerin tam listesi için bkz. setmqweb remote.

   Örneğin, örnek CCDT dosyasıyla birlikte QMGRP kuyruk yöneticisi grubunu eklemek için şu komutu girin:

```
setmqweb remote add  -uniqueName "MyQMGRP" -qmgrName "QMGRP" -ccdtURL
"c:\myccdts\group_ccdt.json" -group
```

Sonuçlar

Uzak kuyruk yöneticisi grubu, URLkaynağındaki benzersiz ad kullanılarak messaging REST API ile birlikte kullanılabilir. İsteği tamamlamak için gruptan bir kuyruk yöneticisi seçilir ve isteği tamamlayan kuyruk yöneticisine ilişkin bilgiler `ibm-mq-resolved-qmgr`yanıt üstbilgisinde döndürülür.

V 9.3.3 messaging REST API tarafından kullanılan güvenlik birincil kullanıcısının belirlenmesi

messaging REST APIkullandığınızda, ileti alışverişi için bağlanmak istediğiniz kuyruk yöneticilerine, kuyruklara ve konulara erişmek için uygun bir kullanıcının yetkisi olmalıdır. Yetkilendirilmesi gereken kullanıcı, mqweb sunucunuzun nasıl yapılandırıldığına ve messaging REST APIile uzak kuyruk yöneticilerini kullanıp kullanmadığınıza bağlıdır.

Varsayılan olarak, kuyruk yöneticisine erişimi yetkilendirmek için kullanılan güvenlik birincil kullanıcısı, messaging REST APIçalıştıran mqweb sunucusunu başlatan kullanıcıdır. Kuyruklara ve konulara erişim yetkisi vermek için kullanılan güvenlik birincil kullanıcısı, messaging REST API' de oturum açan kullanıcıdır. Ancak, mqweb sunucunuz ya da uzak kuyruk yöneticisi bağlantınız farklı bir güvenlik birincil kullanıcısı kullanılacak şekilde yapılandırılmış olabilir.

Kuyruk yöneticisine bağlanmak için kullanılan güvenlik birincil kullanıcısının saptanması

Yerel kuyruk yöneticisi bağlantıları için, kuyruk yöneticisine bağlanmak için kullanılan güvenlik birincil kullanıcısı, messaging REST API' u çalıştıran mqweb sunucusunu başlatan kullanıcıdır. Uzak kuyruk yöneticisi bağlantılarında, messaging REST API kuyruk yöneticisine erişimi öncelik sırasına göre yetkilendirmek için aşağıdaki güvenlik birincil kullanıcıları kullanılır. Uzak kuyruk yöneticisi yapılanışında kullanıcılar birden çok şekilde belirtilirse, listedeki ilk kullanıcı yetki için kullanılır.

1. Güvenlik birincil kullanıcısı, bir güvenlik çıkışından benimsenen bir kullanıcı bağlamdır.
2. Güvenlik birincil kullanıcısı, uzak kuyruk yöneticisine bağlanmak için kullanılan sunucu bağlantısı kanalındaki bir CHLAUTH kuralında benimsenen bir kullanıcı bağlamdır.
3. Güvenlik birincil kullanıcısı, messaging REST APIuzak kuyruk yöneticisi yapılandırmasında bulunan kullanıcı kimliğidir. Bu kullanıcı kimliği, kuyruk yöneticisini **setmqweb remote** komutuyla eklediğinizde isteğe bağlı olarak kuyruk yöneticisi bağlantı bilgilerine eklenir.
4. Güvenlik birincil kullanıcısı, messaging REST API' i çalıştıran mqweb sunucusunu başlatan kullanıcıdır.

messaging REST APIile kullanmak üzere uzak kuyruk yöneticilerini ayarlama hakkında daha fazla bilgi için bkz. “messaging REST API ile kullanılacak uzak kuyruk yöneticisini ayarlama” sayfa 681.

Kuyruklara ve konulara bağlanmak için kullanılan güvenlik birincil kullanıcısının saptanması

messaging REST APIkullanırken kuyruklara ve konulara yönelik bağlantıları yetkilendirmek için hangi güvenlik birincil kullanıcısının kullanıldığını belirlemek üzere mqweb sunucusu yapılandırmasında bir özellik ayarlayabilirsiniz. Bu özellik **mqRestMessagingAdoptWebUserContext** özelliğidir. **dspmqweb properties** komutunu kullanarak bu özelliğin neye ayarlandığı görüntüleyebilirsiniz.

- **mqRestMessagingAdoptWebUserContext** true olarak ayarlanırsa, messaging REST API yetkilendirme için messaging REST API ' da oturum açan kullanıcının kullanıcı kimliğini kullanır. Bu nedenle, messaging REST API ile kullanılmak üzere mqweb sunucusu yapılandırmasında var olan kullanıcı kimliği ya da kullanıcı kimlikleri, kuyruklara ve konulara erişim yetkisi olması gereken güvenlik birincil kullanıcılarıdır.
- **mqRestMessagingAdoptWebUserContext** false olarak ayarlanırsa, messaging REST API yetkilendirme için messaging REST API ' yi barındıran mqweb sunucusunu başlatan kullanıcının kullanıcı kimliğini kullanır. Bu nedenle, messaging REST API ' u barındıran mqweb sunucusunu başlatan kullanıcı kimliğiyle aynı olan bir kullanıcı kimliğinin kuyruklara ve konulara erişim yetkisi olmalıdır.

  Kuyruklarınız ve konularınız uzak bir kuyruk yöneticisindeyse, yetki için kullanılan güvenlik birincil kullanıcısı, kuyruk yöneticisi yapılanışındaki ayarlara göre belirlenebilir. Öncelik sırasına göre aşağıdaki güvenlik birincil kullanıcıları kullanılabilir:

  1. Güvenlik birincil kullanıcısı, bir güvenlik çıkışından benimsenen bir kullanıcı bağlamdır.

2. Güvenlik birincil kullanıcısı, uzak kuyruk yöneticisine bağlanmak için kullanılan sunucu bağlantısı kanalındaki bir CHLAUTH kuralında benimsenen bir kullanıcı bağlamdır. Örneğin, MCAUSER parametresini kullanmak için sunucu bağlantısı kanalında bir CHLAUTH kuralı yapılandırabilirsiniz. Daha sonra, tüm bağlantılar kuyruk yöneticisini kullanma yetkisi olan bir kullanıcı kimliğiyle eşlenir.
3. Güvenlik birincil kullanıcısı, kuyruk yöneticisinin AUTHINFO öğesinden benimsenmiş bir kullanıcı bağlamdır. Kuyruk yöneticisinin CONNAUTH özniteliği tarafından başvurulan AUTHINFO nesnesi **ADOPTCTX(yes)** kullanacak şekilde yapılandırıldıysa, kuyruk yöneticisiyle bağlantıları yetkilendirmek için kullanılan güvenlik birincil kullanıcısı da kuyrukları ve konuları yetkilendirmek için kullanılır. Örneğin, bu güvenlik birincil kullanıcısı, **setmqweb remote** komutunun bir parçası olarak uzak kuyruk yöneticisi bağlantı bilgisinde bulunan kullanıcı kimliği olabilir.

İlgili bilgiler

CHLAUTH.

CONNAUTH

dspmqweb özellikleri

IBM MQ ile MQI uygulamaları geliştirilmesi

IBM MQ , C, Visual Basic, COBOL, Assembler, RPG, pTALve PL/I için destek sağlar. Bu yordamsal diller, ileti kuyruklama hizmetlerine erişmek için ileti kuyruğu arabirimini (MQI) kullanır.

Uygulamalarınızı seçtiğiniz dilde nasıl yazacağınıza ilişkin ayrıntılı bilgi için alt konulara bakın.

Yordam dillerine ilişkin çağrı arabirimine genel bakış için bkz. Çağrı açıklamaları. Bu konu, MQI çağrılarının bir listesini içerir ve her arama, bu dillerin her birinde çağrıları nasıl kodladığınızı gösterir.

IBM MQ , uygulamalarınızı yazmanıza yardımcı olacak veri tanımlama dosyalarını sağlar. Tam açıklama için bkz. “IBM MQ veri tanımlama dosyaları” sayfa 688.

Programlarınızı kodlayacak yordam dilini seçmenize yardımcı olmak için, programlarınızın işleyeceği ileti uzunluğu üst sınırını göz önünde bulundurun. Programlarınız yalnızca bilinen bir uzunluk üst sınırına sahip iletileri işlerse, bunları desteklenen dillerden herhangi birinde kodlayabilirsiniz. Programların işlemesi gereken ileti uzunluğu üst sınırını bilmiyorsanız, seçtiğiniz dil bir CICS, IMSya da toplu iş uygulaması yazıp yazmadığınıza bağlıdır:

IMS ve toplu iş

C, PL/I ya da çevirici dilindeki programları, isteğe bağlı bellek miktarlarını elde etmek ve serbest bırakmak için bu dillerin sunduğu olanakları kullanacak şekilde kodlayın. Diğer bir yöntem olarak, programlarınızı COBOL dilinde kodlayabilir, ancak depolama almak ve yayınlamak için çevirici dili, PL/I ya da C alt yordamlarını kullanabilirsiniz.

CICS

Programları CICStarafından desteklenen herhangi bir dilde kodlayın. EXEC CICS arabirimi, gerekirse bellek yönetimine ilişkin çağrıları sağlar.

İlgili kavramlar

“Nesne yönelimli uygulamalar” sayfa 15

IBM MQ , JMS, Java, C + + ve .NETiçin destek sağlar. Bu diller ve çerçeveler, IBM MQ çağrıları ve yapılarıyla aynı işlevselliği sağlayan sınıfları sağlayan IBM MQ Nesne Modeli 'ni kullanır.

Teknik genel bakış

“Uygulama geliştirme kavramları” sayfa 6

IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

İlgili başvurular

Uygulama başvurusu geliştirilmesi

IBM MQ veri tanımlama dosyaları

IBM MQ , uygulamalarınızı yazmanıza yardımcı olacak veri tanımlama dosyalarını sağlar.

Veri tanımlama dosyaları şu şekilde de bilinir:

Dil	Veri tanımlamaları
C	Dosyaları ya da üstbilgi dosyalarını içer
Visual Basic	Modül dosyaları (yalnızca 32 bit sürümler)
COBOL	Dosyaları kopyala
Çevirici	Makrolar
PL/I	Dosyaları içer

Kanal çıkışlarını yazmanıza yardımcı olacak veri tanımlama dosyaları IBM MQ COPY, header, include, module filesbaşlıklı konuda açıklanmıştır.

Kurulabilir hizmet çıkışlarını yazmanıza yardımcı olacak veri tanımlama dosyaları “Kullanıcı çıkışları, API çıkışları ve IBM MQ kurulabilir hizmetleri” sayfa 896içinde açıklanmıştır.

C + + üzerinde desteklenen veri tanımlama dosyaları için bkz. C++ kullanarak.

IBM i

RPG ' de desteklenen veri tanımlama dosyaları için bkz. IBM i Application Programming Reference (ILE/RPG).

Veri tanımlama dosyalarının adları CMQ önekini ve programlama dili tarafından belirlenen bir soneki içerir:

Sonek	Dil
a	Çevirici dili
b	Visual Basic
c	C
l	COBOL (ilk kullanıma hazırlanmış değerler olmadan)
p	PL/I
v	COBOL (varsayılan değerler ayarlanmış olarak)

Kuruluş kitaplığı

thlqual adı, z/OSüzerindeki kuruluş kitaplığının üst düzey niteleyicidir.

Bu konuda, aşağıdaki başlıklar altında IBM MQ veri tanımlama dosyaları tanıtılmaktadır:

- “C dili dosyaları içerir” sayfa 689
- “Visual Basic modül dosyaları” sayfa 690
- “COBOL kopyalama dosyaları” sayfa 690
- z/OS “System/390 çevirici dili makroları” sayfa 691
- z/OS “PL/I içerme dosyaları” sayfa 691

C dili dosyaları içerir

IBM MQ C içerme dosyaları C üstbilgi dosyalarındalistelenir. Bunlar aşağıdaki dizinlere ya da kitaplıklara kurulur:

Hizmet olarak sunulan	Kuruluş dizini ya da kitaplığı
IBM i IBM i	QMQM/H

Hizmet olarak sunulan	Kuruluş dizini ya da kitaplığı
Linux AIX AIX and Linux	*MQ_INSTALLATION_PATH*/inc/
Windows Windows	*MQ_INSTALLATION_PATH*\Araçlar \c\include
z/OS z/OS	**thlqual**.SCSQC370

Burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Not: AIX and Linuxiçin, içerme dosyaları sembolik olarak `/usr/include`eile bağlantılıdır.

Dizinlerin yapısıyla ilgili daha fazla bilgi için Planning file system support(Dosya sistemi desteğini planlama) başlıklı konuya bakın.

Visual Basic modül dosyaları

IBM MQ for Windows , dört Visual Basic modülü dosyası sağlar.

Bunlar, Visual Basic modül dosyalarında listelenir ve şu adrese kurulur:

```
MQ_INSTALLATION_PATH\Tools\Samples\VB\Include
```

COBOL kopyalama dosyaları

COBOL için IBM MQ , adlandırılmış değişmezleri içeren ayrı kopya dosyaları ve her bir yapı için iki kopya dosyası sağlar.

Her biri ilk değerlerle birlikte ve başlangıç değerleri olmadan sağlandığından, her yapı için iki kopya dosyası vardır:

- Bir COBOL programının ÇALIŞMA-DEPOLAMA bölümünde, yapı alanlarını varsayılan değerlerle başlatan dosyaları kullanın. Bu yapılar, V harfi (değerler) ile sonekli adlara sahip kopya dosyalarında tanımlanır.
- Bir COBOL programının BAğLANTı BöLüMüNDE, ilk değerleri olmayan yapıları kullanın. Bu yapılar, adının sonuna L harfi (bağ) eklenmiş olan kopya dosyalarında tanımlanır.

IBM i IBM i için veri ve arabirim tanımlamalarını içeren kopyalama dosyaları, MQI 'a prototip atanmış çağrılar kullanılarak ILE COBOL programları için sağlanır. Dosyalar, QMQM/QCBLLESRC içinde, soneki L (ilk değerleri olmayan yapılar için) ya da soneki V (ilk değerleri olan yapılar için) olan üye adlarıyla bulunur.

IBM MQ COBOL kopyalama dosyaları COBOL COPY dosyalarıiçinde listelenir. Bunlar aşağıdaki dizinlere kurulur:

Hizmet olarak sunulan	Kuruluş dizini ya da kitaplığı
Linux AIX AIX and Linux	*MQ_INSTALLATION_PATH*/inc/
IBM i IBM i	QMQM/QCBLLESRC
Windows Windows	*MQ_INSTALLATION_PATH*\Tools\cobol\copybook (Micro Focus COBOL için) *MQ_INSTALLATION_PATH*\Tools\cobol\copybook\VAcobol (IBM VisualAge COBOL için)
z/OS z/OS	**thlqual**.SCSQCOBC

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Programınıza yalnızca gereksinim duyduğunuz dosyaları ekleyin. Bunu, level-01 bildiriminden sonra bir ya da daha çok COPY deyimiyle yapın. Bu, gerekirse bir programa yapıların birden çok sürümünü ekleyebileceğiniz anlamına gelir. CMQV ' nin büyük bir dosya olduğunu unutmayın.

Aşağıda, CMQMDV kopya dosyasını dahil etmek için bir COBOL kodu örneği verilmiştir:

```
01 MQM-MESSAGE-DESCRIPTOR.
COPY CMQMDV.
```

Her yapı bildirimi bir level-01 öğesiyle başlar; yapı bildiriminin geri kalanında kopyalanacak bir COPY deyiminin ardından level-01 bildirimini kodlayarak yapının birkaç eşgörünümünü bildirebilirsiniz. Uygun yönetim ortamına gönderme yapmak için IN anahtar sözcüğünü kullanın.

Aşağıda, CMQMDV ' nin iki eşgörünümünü içermek için bir COBOL kodu örneği verilmiştir:

```
* 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.
```

4 baytlık sınırlardaki yapıları hizalayın. COPY deyimini, level-01 öğesi olmayan bir öğeyi izleyen bir yapıyı içermek için kullanırsanız, yapının level-01 öğesinin başlangıcından itibaren 4 baytlık bir kat olmasına dikkat edin. Bunu yapmazsanız, uygulamanızın performansını düşürebilirsiniz.

Yapılar, MQI 'da kullanılan veri tiplerindeaçıklanmıştır. Yapılardaki alanların açıklamaları, önek içermeyen alanların adlarını gösterir. COBOL programlarında, COBOL bildirimlerinde gösterildiği gibi, alan adlarına yapının adını ve ardından bir tire işareti ekleyin. Yapı kopyalama dosyalarındaki alanların başına bu şekilde önek eklenir.

Yapı kopyalama dosyalarındaki bildirimlerdeki alan adları büyük harfli. Bunun yerine büyük ve küçük harf karışık kullanabilirsiniz. Örneğin, MQGMO yapısının *StrucId* alanı, COBOL bildiriminde ve kopyalama dosyasında MQGMO-STRUCID olarak gösterilir.

V soneki yapıları, tüm alanlar için başlangıç değerleriyle bildirilir; bu nedenle, yalnızca, gerekli değerin ilk değerden farklı olduğu alanları ayarlamanız gerekir.

System/390 çevirici dili makroları

z/OS

IBM MQ for z/OS , adlandırılmış değişmezleri içeren iki çevirici dili makrosu ve her bir yapıyı oluşturmak için bir makro sağlar.

Bunlar z/OS Assembler COPY dosyalarında listelenir ve **thlqual**.SCSQMACS içine kurulur.

Bu makrolara şu kod kullanılarak çağrılır:

```
MY_MQMD CMQMDA EXPIRY=0,MSGTYPE=MQMT_DATAGRAM
```

PL/I içerme dosyaları

z/OS

IBM MQ for z/OS , PL/I 'de IBM MQ uygulamaları yazarken gereksinim duyduğunuz tüm tanımlamaları içeren içerme dosyalarını sağlar.

Dosyalar PL/I include files içinde listelenir ve **thlqual**.SCSQPLIC dizinine kurulur:

IBM MQ sınırlı kod öbeğini programınıza bağlamak istiyorsanız, bu dosyaları programınıza ekleyin (bkz. “Programınızın çalıştırılmak üzere hazırlanması” sayfa 979). IBM MQ aramalarını dinamik olarak bağlamak istiyorsanız yalnızca CMQP ' yi ekleyin (bkz. “IBM MQ sınırlı kod öbeğini dinamik olarak çağırma” sayfa 986). Dinamik bağlantı oluşturma yalnızca toplu iş ve IMS programları için gerçekleştirilebilir.

Kuyruğa alma için yordam uygulaması yazılması

Kuyruğa alma uygulamaları yazma, bir kuyruk yöneticisine bağlanma ve bağlantı kesme, yayınlama/abone olma ve nesneleri açma ve kapatma hakkında bilgi edinmek için bu bilgileri kullanın.

Uygulama yazmakla ilgili daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692
- “Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705
- “Nesnelerin açılması ve kapatılması” sayfa 712
- “Kuyruktaki iletilerin konması” sayfa 722
- “Kuyruktan ileti alma” sayfa 736
- “Yayınlama/abone olma uygulamaları yazılıyor” sayfa 776
- “Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
- “İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818
- “Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
- “MQI ve kümelerle çalışma” sayfa 847
- z/OS “IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852
- z/OS “IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları” sayfa 65

İlgili kavramlar

“Uygulama geliştirme kavramları” sayfa 6
IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

“IBM MQ için uygulama geliştirilmesi” sayfa 5
İleti göndermek ve almak için uygulamalar geliştirebilir ve kuyruk yöneticilerinizi ve ilgili kaynakları yönetebilirsiniz. IBM MQ , birçok farklı dilde ve çerçevede yazılmış uygulamaları destekler.

“IBM MQ uygulamaları için tasarımla ilgili önemli noktalar” sayfa 47
Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğine karar verdiğinizde, IBM MQtarafından sunulan özellikleri nasıl kullanacağınıza karar vermeniz gerekir.

“İstemci yordamsal uygulamaları yazılıyor” sayfa 875
IBM MQ üzerinde yordam dili kullanarak istemci uygulamaları yazmak için bilmeniz gerekenleri.

“Yordamsal uygulama oluşturma” sayfa 957
Birkaç yordam dilinden birinde bir IBM MQ uygulaması yazabilir ve uygulamayı birkaç farklı platformda çalıştırabilirsiniz.

“Yordamsal program hatalarının işlenmesi” sayfa 994
Bu bilgiler, bir çağrı yaptığında ya da iletisi son hedefine teslim edildiğinde, uygulamalarınızla ilişkili MQI çağrılarıyla ilgili hataları açıklar.

İlgili görevler

“IBM MQ örnek yordam programlarının kullanılması” sayfa 1013
Bu örnek programlar yordamsal dillerde yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir. Farklı platformlarda IBM MQ programları.

İleti Kuyruğu Arabirimine Genel Bakış

İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

İleti Kuyruğu Arabirimi aşağıdakilerden oluşur:

- Programların kuyruk yöneticisine ve olanaklarına erişebileceği *çağrılar*
- Programların kuyruk yöneticisine veri aktarmak ve kuyruk yöneticisinden veri almak için kullandığı *yapılar*
- Kuyruk yöneticisine veri aktarmak ve kuyruk yöneticisinden veri almak için *temel veri tipleri*

z/OS IBM MQ for z/OS ayrıca şunları sağlar:

- z/OS toplu iş programlarının değişiklikleri kesinleştirebileceği ve geri alabileceği fazladan iki çağrı.
- *Veri tanımlama dosyaları* (bazen kopya dosyaları, makrolar, içerme dosyaları ve üstbilgi dosyaları olarak da bilinir), IBM MQ for z/OSile verilen sabitlerin değerlerini tanımlar.
- Uygulamalarınızla bağlantı düzenlemek için *sınırlı kod öbeği programları* .
- z/OS platformunda MQI 'ın nasıl kullanılacağını gösteren örnek programlar grubu. Bu örneklerle ilgili daha fazla bilgi için bkz. “z/OS için örnek programların kullanılması” sayfa 1111.

IBM i IBM MQ for IBM i ayrıca şunları sağlar:

- *Veri tanımlama dosyaları* (bazen kopya dosyaları, makrolar, içerme dosyaları ve üstbilgi dosyaları olarak da bilinir), IBM MQ for IBM iile verilen sabitlerin değerlerini tanımlar.
- ILE C, ILE COBOL ve ILE RPG uygulamalarınıza bağlantı düzenlemek için üç sınırlı kod öbeği programı.
- IBM i platformunda MQI 'ın nasıl kullanılacağını gösteren örnek programlar grubu.

AIX, Linux, and Windows sistemleri de aşağıdakileri sağlar:

- IBM MQ for AIX, Linux, and Windows sistem programlarının değişiklikleri kesinleştirebileceği ve geri alabileceği çağrılar.
- Bu altyapılarda sağlanan değişmezlerin değerlerini tanımlayan *dosyaları içer* .
- Uygulamalarınızı bağlamak için *Kitaplık dosyaları* .
- Bu altyapılarda MQI ' ın nasıl kullanılacağını gösteren örnek programlar grubu. Bu örneklerle ilgili daha fazla bilgi için bkz. “Multiplatforms üzerinde örnek programların kullanılması” sayfa 1014.
- Dış hareket yöneticilerine yönelik bağ tanımları için örnek kaynak ve yürütülebilir kod.

MQI hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “MQI çağrıları” sayfa 694
- “Nokta çağrılarını eşitle” sayfa 695
- “Veri dönüştürme, veri tipleri, veri tanımları ve yapılar” sayfa 695
- “IBM MQ sınırlı kod öbeği programları ve kitaplık dosyaları” sayfa 696
- “Tüm çağrılar için ortak olan parametreler” sayfa 701
- “Arabelleklerin belirtilmesi” sayfa 702
- z/OS “z/OS toplu işte dikkat edilmesi gereken noktalar” sayfa 702
- “AIX and Linux sinyal işleme” sayfa 703

İlgili kavramlar

“Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

“Kuyruktaki iletilerin konması” sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

“Kuyruktan ileti alma” sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

“Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

“Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

“MQI ve kümelerle çalışma” sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

“IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

“IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları” sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

MQI çağrıları

İleti Kuyruğu Arabirimi 'ndeki (MQI) çağrılar hakkında bilgi edinmek için bu bilgileri kullanın.

MQI 'daki çağrılar aşağıdaki gibi gruplanabilir:

MQCONN, MQCONNX ve MQDISC
Bir programı, bir kuyruk yöneticisine (seçeneklerle ya da seçeneklerle) bağlamak ve bir programın bağlantısını kesmek için bu çağrıları kullanın. z/OSiçin CICS programları yazarsanız, bu çağrıları kullanmanız gerekmez. Ancak, uygulamanızı diğer platformlara bağlamayı istiyorsanız bunları kullanmanız önerilir.

MQOPEN ve MQCLOSE
Kuyruk gibi bir nesneyi açmak ve kapatmak için bu çağrıları kullanın.

MQPUT ve MQPUT1
Kuyruğa bir ileti koymak için bu çağrıları kullanın.

MQGet
Kuyruktaki iletilere göz atmak ya da kuyruktaki iletileri kaldırmak için bu çağrıyı kullanın.

MQSUB, MQSUBRQ
Bir konuya abonelik kaydetmek ve abonelikle eşleşen yayınları istemek için bu çağrıları kullanın.

MQINQ
Bir nesnenin özniteliklerini sorgulamak için bu çağrıyı kullanın.

MQSET
Bir kuyruğun bazı özniteliklerini ayarlamak için bu çağrıyı kullanın. Diğer nesne tiplerinin özniteliklerini ayarlayamazsınız.

MQBEGIN, MQCMIT ve MQBACK
IBM MQ bir iş biriminin eşgüdümcüsü olduğunda bu çağrıları kullanın. MQBEGIN, iş birimini başlatır. MQCMIT ve MQBACK, iş birimi sırasında yapılan güncellemeleri kesinleştirerek ya da geriye işleyerek iş birimini sona erdirdi. IBM i IBM i kesinleştirme denetleyicisi, IBM MQ for IBM iüzerinde genel iş birimlerini koordine etmek için kullanılır. Yerel başlatma kesinleştirme denetimi, kesinleştirme ve geriye işleme komutları kullanılır.

MQCRTMH, MQBUFMH, MQMHBUF, MQDLTMH
İleti tanıtıcısı yaratmak, ileti tanıtıcısını arabelleğe ya da arabelleği ileti tanıtıcısına dönüştürmek ve ileti tanıtıcısını silmek için bu çağrıları kullanın.

MQSETMP, MQINQMP, MQDLTMP
İleti tanıtıcısında bir ileti özelliği ayarlamak, bir ileti özelliğini sorgulamak ve ileti tanıtıcısından bir özelliği silmek için bu çağrıları kullanın.

MQCB, MQCB_FUNCTION, MQCTL
Bir geri çağırma işlevini kaydetmek ve denetlemek için bu çağrıları kullanın.

MQSTAT
Önceki zamanuyumsuz koyma işlemlerine ilişkin durum bilgilerini almak için bu çağrıyı kullanın.

MQI çağrılarına ilişkin açıklamalar için Çağrı açıklamaları konusuna bakın.

Nokta çağrılarını eşitle
Farklı platformlardaki eşitleme noktası çağrıları hakkında bilgi edinmek için bu bilgileri kullanın.

Eşitleme noktası çağrıları aşağıdaki gibi kullanılabilir:

IBM MQ for z/OS çağrılar
z/OS

IBM MQ for z/OS , MQCMIT ve MQBACK çağrılarını sağlar.

z/OS toplu iş programlarındaki bu çağrıları, kuyruk yöneticisine son eşitleme noktasından bu yana tüm MQGET ve MQPUT işlemlerinin kalıcı (kesinleştirilmiş) kılınması ya da geriletilmesi gerektiğini söylemek için kullanın. Diğer ortamlardaki değişiklikleri kesinleştirmek ve geri almak için:

CICS
EXEC CICS SYNCPOINT ve EXEC CICS SYNCPOINT ROLLBACK gibi komutları kullanın.

IMS
IOPCB, CHKP (denetim noktası) ve ROLB (geriye işleme) çağrıları için GU (benzersiz alma) gibi IMS eşitleme noktası olanaklarını kullanın.

RRS
Uygun şekilde MQCMIT ve MQBACK ya da SRRCMIT ve SRRBACK kullanın. (Bkz. “Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri” sayfa 822.)

Not: SRRCMIT ve SRRBACK yerel RRS komutlarıdır, MQI çağrıları değildir.

IBM i çağrılar
IBM i

IBM MQ for IBM i , MQCMIT ve MQBACK komutlarını sağlar. IBM i COMMIT ve ROLLBACK komutlarını ya da IBM i kesinleştirme denetimi olanaklarını başlatan diğer komutları ya da çağrıları da kullanabilirsiniz (örneğin, EXEC CICS SYNCPOINT).

AIX, Linux, and Windows platformlarında IBM MQ çağrıları
ALW

IBM MQ for AIX, Linux, and Windows MQCMIT ve MQBACK çağrılarını sağlar.

Kuyruk yöneticisine, son eşitleme noktasından bu yana tüm MQGET ve MQPUT işlemlerinin kalıcı (kesinleştirilmiş) kılınmasını ya da geriletilmesini söylemek için programlardaki eşitleme noktası çağrılarını kullanın. CICS ortamındaki değişiklikleri kesinleştirmek ve geri yüklemek için EXEC CICS SYNCPOINT ve EXEC CICS SYNCPOINT ROLLBACK gibi komutları kullanın.

Veri dönüştürme, veri tipleri, veri tanımları ve yapılar
İleti Kuyruğu Arabirimi 'ni kullanırken veri dönüştürmeleri, temel veri tipleri, IBM MQ veri tanımlamaları ve yapıları hakkında bilgi edinmek için bu bilgileri kullanın.

Veri dönüştürme

MQXCNVC (karakterleri dönüştür) çağrısı, ileti karakteri verilerini bir karakter kümesinden diğerine dönüştürür. IBM MQ for z/OSdışında, bu çağrı yalnızca bir veri dönüştürme çıkışından kullanılır.

MQXCNVC çağrısıyla kullanılan sözdizimi için MQXCNVC-karakter dönüştürme ve veri dönüştürme çıkışlarının yazılması ve çağrılmasıyla ilgili yönergeler için “Veri dönüştürme çıkışları yazılıyor” sayfa 941 başlıklı konuya bakın.

Temel veri tipleri

MQI, desteklenen programlama dilleri için temel veri tipleri ya da yapılandırılmamış alanlar sağlar.

Bu veri tipleri, Temel veri tiplerindetam olarak açıklanmıştır.

IBM MQ veri tanımlamaları

z/OS IBM MQ for z/OS , COBOL kopya dosyaları, derleme dili makroları, tek bir PL/I dosyası, tek bir C dili içeren dosya ve C++ dili içeren dosyalar biçiminde veri tanımları sağlar.

IBM i IBM MQ for IBM i , COBOL kopyalama dosyaları, RPG kopyalama dosyaları, C dili dosyalar ve C++ dili içerme dosyaları biçiminde veri tanımları sağlar.

IBM MQ ile sağlanan veri tanımlama dosyaları aşağıdakileri içerir:

- Tüm IBM MQ değişmezlerinin ve dönüş kodlarının tanımları
- IBM MQ yapılarının ve veri tiplerinin tanımlamaları
- Yapıların kullanıma hazırlanmasına ilişkin değişmez tanımlar
- Çağrıların her biri için işlev prototipleri (yalnızca PL/I ve C dili için)

IBM MQ veri tanımlama dosyalarının tam açıklaması için bkz. “IBM MQ veri tanımlama dosyaları” sayfa 688.

Yapılar

“MQI çağrıları” sayfa 694içinde listelenen MQI çağrılarıyla birlikte kullanılan yapılar, desteklenen programlama dillerinin her biri için veri tanımlama dosyalarında sağlanır. IBM i z/OS IBM MQ for z/OS ve IBM MQ for IBM i , bu yapıların bazı alanlarını doldururken kullanabileceğiniz sabitleri içeren dosyaları sağlar. Bunlar hakkında daha fazla bilgi için bkz. IBM MQ veri tanımlamaları.

Yapıların bir özeti için bkz. Yapı veri tipleri .

IBM MQ sınırlı kod öbeği programları ve kitaplık dosyaları

Sağlanan sınırlı kod öbeği programları ve kitaplık dosyaları, her platform için burada listelenir.

Yürütülebilir bir uygulama oluştururken sınırlı kod öbeği programlarının ve kitaplık dosyalarının nasıl kullanılacağına ilişkin ek bilgi için bkz. “Yordamsal uygulama oluşturma” sayfa 957. C++ kitaplık dosyalarına bağlanma hakkında bilgi için bkz. C++ kullanarak *IBM MQ C++ kullanma*.

AIX *IBM MQ for AIX kitaplık dosyaları*

IBM MQ for AIXişletim sistemi tarafından sağlananlara ek olarak, programınızı, uygulamanızı çalıştırdığınız ortam için sağlanan MQI kitaplık dosyalarına bağlamanız gerekir.

İş parçacığı kullanmayan bir uygulamada, aşağıdaki kitaplıklardan birine bağlantı sağlayın:

Çizelge 106. İş parçacığı kullanmayan AIX uygulamaları için kitaplık dosyaları

Kitaplık dosyası	**Çevre**
libmqm.a	C İçin Sunucu
libmqic.a ve libmqm.a	C İçin İstemci
libmqmzf.a	C için kurulabilir hizmet çıkışları
libmqmxa.a	Sunucu XA arabirimi
libmqmxa64.a	Sunucu diğer XA arabirimi
libmqcxa.a	İstemci XA arabirimi
libmqcxa64.a	İstemci alternatif XA arabirimi

Çizelge 106. İş parçacığı kullanmayan AIX uygulamaları için kitaplık dosyaları (devamı var)	
Kitaplık dosyası	**Çevre**
libmqmcbrt.o	Micro Focus COBOL desteği için IBM MQ çalıştırma zamanı kitaplığı
libmqmcb.a	COBOL Sunucusu
libmqicb.a	COBOL İçin İstemci
libimqc23ia.a	Client for C++ (XLC 16)
libimqs23ia.a	Sunucu-C++ (XLC 16)
V 9.3.5 libimqc23ca.a	Client for C++ (XLC 17)
V 9.3.5 libimqs23ca.a	Sunucu-C++ (XLC 17)

V 9.3.5 "ia" içeren kitaplıklar XLC 16 derleyicisiyle oluşturulmuşken, adında "ca" olan kitaplıklar XLC 17 derleyicisiyle oluşturulmuştur.

İş parçacıklı bir uygulamada, aşağıdaki kitaplıklardan birine bağlantı sağlayın:

Çizelge 107. İş parçacıklı AIX uygulamaları için kitaplık dosyaları. Kitaplık dosyalarını ve her kitaplık dosyasına ilişkin ortamı listeleyen iki sütunlu bir çizelge.	
Kitaplık dosyası	**Çevre**
libmqm_r.a	C İçin Sunucu
libmqic_r.a ve libmqm_r.a	C İçin İstemci
libmqmzf_r.a	C için kurulabilir hizmet çıkışları
libmqmxa_r.a	Sunucu XA arabirimi
libmqmxa64_r.a	Sunucu diğer XA arabirimi
libmqcxa_r.a	İstemci XA arabirimi
libmqcxa64_r.a	İstemci alternatif XA arabirimi
libimqc23ia_r.a	Client for C++ (XLC 16)
libimqs23ia_r.a	Sunucu-C++ (XLC 16)
V 9.3.5 libimqc23ca_r.a	Client for C++ (XLC 17)
V 9.3.5 libimqs23ca_r.a	Sunucu-C++ (XLC 17)

V 9.3.5 "ia" içeren kitaplıklar XLC 16 derleyicisiyle oluşturulmuşken, adında "ca" olan kitaplıklar XLC 17 derleyicisiyle oluşturulmuştur.

Not: Birden çok kitaplığa bağlantı veremezsiniz. Yani, aynı anda hem iş parçacıklı hem de iş parçacıklı olmayan bir kitaplığa bağlantı veremezsiniz.

IBM i *IBM MQ for IBM i kitaplık dosyaları*

IBM MQ for IBM iiçinde, programınızı işletim sistemi tarafından sağlananlara ek olarak, uygulamanızı çalıştırdığınız ortam için sağlanan MQI kitaplık dosyalarına bağlayın.

İş parçacıklı olmayan uygulamalar için:

Çizelge 108. İş parçacığı kullanmayan IBM i uygulamaları için kitaplık dosyaları	
Kitaplık dosyası	**Çevre**
LIBMQM	Sunucu ve İstemci hizmet programı
LIBMQIC (IBM QIC)	İstemci hizmet programı
IMQB23I4	C++ temel hizmet programı
IMQS23I4	C++ sunucu hizmeti programı
LIBMQMZF	C için kurulabilir çıkışlar

İş parçacıklı bir uygulamada:

Çizelge 109. İş parçacıklı IBM i uygulamaları için kitaplık dosyaları	
Kitaplık dosyası	**Çevre**
LIBMQM_R	Sunucu & istemcisi hizmet programı
IMQB23I4_R	C++ temel hizmet programı
IMQS23I4_R	C++ sunucu hizmeti programı
LIBMQMZF_R	C için kurulabilir çıkışlar
LIBMQIC_R	İstemci hizmet programı

IBM MQ for IBM i'da C + +' da uygulamalarınızı yazabilirsiniz. C++ uygulamalarınızı nasıl bağlayacağınızı ve C + + kullanmanın tüm yönleriyle ilgili tüm ayrıntıları görmek için bkz. C++ kullanarak.

Linux *IBM MQ for Linux kitaplık dosyaları*

IBM MQ for Linuxüzerinde, işletim sistemi tarafından sağlananlara ek olarak, uygulamanızı çalıştırdığınız ortam için sağlanan MQI kitaplık dosyalarına programınızı bağlamanız gerekir.

İş parçacığı kullanmayan bir uygulamada, aşağıdaki kitaplıklardan birine bağlantı sağlayın:

Çizelge 110. İş parçacığı kullanmayan Linux uygulamaları için kitaplık dosyaları	
Kitaplık dosyası	**Çevre**
libmqm.so	C İçin Sunucu
libmqic.so ve libmqm.so	C İçin İstemci
libmqmzf.so	C için kurulabilir hizmet çıkışları
libmqmxa.so	Sunucu XA arabirimi
libmqmxa64.so	Sunucu diğer XA arabirimi
libmqcxa.so	İstemci XA arabirimi
libmqcxa64.so	İstemci alternatif XA arabirimi
libimqc23gl.so	C++ için İstemci
libimqs23gl.so	C++ için sunucu

İş parçacıklı bir uygulamada, aşağıdaki kitaplıklardan birine bağlantı sağlayın:

Çizelge 111. İş parçacıklı Linux uygulamaları için kitaplık dosyaları	
Kitaplık dosyası	**Çevre**
libmqm_r.so	C İçin Sunucu
libmqic_r.so ve libmqm_r.so	C İçin İstemci
libmqmzf_r.so	C için kurulabilir hizmet çıkışları
libmqmxa_r.so	Sunucu XA arabirimi
libmqmxa64_r.so	Sunucu diğer XA arabirimi
libmqcxa_r.so	İstemci XA arabirimi
libmqcxa64_r.so	İstemci alternatif XA arabirimi
libimqc23gl_r.so	C++ için İstemci
libimqs23gl_r.so	C++ için sunucu

Not: Birden çok kitaplığa bağlantı veremezsiniz. Yani, aynı anda hem iş parçacıklı hem de iş parçacıklı olmayan bir kitaplığa bağlantı veremezsiniz.

Windows *IBM MQ for Windows kitaplık dosyaları*

IBM MQ for Windowsişletim sistemi tarafından sağlananlara ek olarak, programınızı, uygulamanızı çalıştırdığınız ortam için sağlanan MQI kitaplık dosyalarına bağlamanız gerekir:

Çizelge 112. Windows uygulamaları için kitaplık dosyaları	
Kitaplık Dosyası	**Çevre**
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	Sunucu-C (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	Client for C (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqmxa.lib	C için Sunucu XA arabirimi (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqcxa.lib	C için İstemci XA arabirimi (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	Client MTS for C (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcics4.lib32	C için sunucu TXSeries CICS desteği (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqccics4.lib32	C (32 bit) için İstemci TXSeries CICS desteği
MQ_INSTALLATION_PATH\Tools\Lib\mqmzf.lib	C (32 bit) için kurulabilir hizmet çıkışları
MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb.lib	IBM COBOL için sunucu (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqmcb.lib	Micro Focus COBOL Sunucusu (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccbb.lib	Client for IBM COBOL (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\mqiccb.lib	Client for Micro Focus COBOL (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqs23vn.lib	C++ için sunucu (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqc23vn.lib	Client for C++ (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqb23vn.lib	C++ için temel (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib\imqx23vn.lib	Client MTS for C++ (32 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	C için sunucu (64 bit)

Çizelge 112. Windows uygulamaları için kitaplık dosyaları (devamı var)	
Kitaplık Dosyası	**Çevre**
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	Client for C (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmxa.lib	C (64 bit) için Sunucu XA arabirimi
MQ_INSTALLATION_PATH\Tools\Lib64\mqcxa.lib	C için İstemci XA arabirimi (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	Client MTS for C (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb.lib	IBM COBOL sunucusu (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb.lib	Micro Focus COBOL Sunucusu (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccbb.lib	Client for IBM COBOL (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb.lib	Client for Micro Focus COBOL (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\imqs23vn.lib	C++ için sunucu (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\imqc23vn.lib	Client for C++ (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\imqb23vn.lib	C++ için temel (64 bit)
MQ_INSTALLATION_PATH\Tools\Lib64\imqx23vn.lib	Client MTS for C++ (64 bit)

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

.NET programlarını derlemek için `amqmdnet.dll` komutunu kullanın. Daha fazla bilgi için “IBM MQ .NET programlarının derlenmesi” sayfa 588 bölümdeki “.NET uygulamalarının geliştirilmesi” sayfa 533 bölümüne bakın.

Bu dosyalar, önceki yayınlarla uyumluluk için gönderilir:

```
mqic32.lib
mqic32xa.lib
```

z/OS *IBM MQ for z/OS sınırlı kod öbeği programları*

IBM MQ for z/OSile yazılmış bir programı çalıştırmadan önce, uygulamayı çalıştırdığınız ortama ilişkin olarak IBM MQ for z/OS ile birlikte sağlanan sınırlı kod öbeği programına bu programı bağlamanız gerekir.

Sınırlı kod öbeği programı, IBM MQ for z/OS ' in işleyebileceği isteklere yönelik çağrılarınızın işlenmesinin ilk aşamasını sağlar.

IBM MQ for z/OS aşağıdaki kod parçası programlarını sağlar:

CSQBSTUB
z/OS toplu iş programları için sınırlı kod öbeği programı

CSQBRRSI
MQI yoluyla RRS kullanan z/OS toplu iş programları için sınırlı kod öbeği programı

CSQBRSTB
Doğrudan RRS kullanan z/OS toplu iş programları için sınırlı kod öbeği programı

CSQCSTUB
CICS programları için sınırlı kod öbeği programı

CSQQSTUB
IMS programları için sınırlı kod öbeği programı

CSQXSTUB
CICS dışı dağıtılmış kuyruğa alma çıkışları için sınırlı kod öbeği programı

CSQASTUB
Veri dönüştürme çıkışları için sınırlı kod öbeği programı

Uyarı: Belirli bir ortam için listelenenden başka bir sınırlı kod öbeği programı kullanırsanız, bu programın öngörülemeyen sonuçları olabilir.

Not: CSQBRSTB sınırlı kod öbeği programını kullanıyorsanız, SYS1.CSSLIB. (SYS1.CSSLIB , *Çağrılabilir Hizmetler Kitaplığı* olarak da bilinir). RRS hakkında daha fazla bilgi için bkz. "Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri" sayfa 822.

Diğer bir seçenek olarak, sınırlı kod öbeğini programınızdan dinamik olarak çağırabilirsiniz. Bu teknik "IBM MQ sınırlı kod öbeğini dinamik olarak çağırma" sayfa 986içinde açıklanmıştır.

IMSiçinde, IBM MQtarafından sağlanan özel bir dil arabirim modülü de kullanmanız gerekebilir.

Aynı IMS MPP bölgesinde CSQBSTUB ve CSQQSTUB ile bağlantılı olarak düzenlenen uygulamaları çalıştırmayın. Bu, DFS3607I ya da CSQQ005E iletileri gibi sorunlara neden olabilir. Adres alanındaki ilk MQCONN çağrısı hangi arabirimin kullanıldığını belirler; bu nedenle, CSQQSTUB ve CSQBSTUB hareketlerinin farklı IMS ileti bölgelerinde çalışması gerekir.

Tüm çağrılar için ortak olan parametreler
Tüm çağrılar için ortak olan iki tip parametre vardır: tanıtıcıları ve dönüş kodları.

Çekme noktalarını kullanma

Tüm MQI çağrıları bir ya da daha çok *tanıtıcı*kullanır. Bunlar, kuyruk yöneticisini, kuyruğu ya da başka bir nesneyi, iletiyi ya da aboneliği çağrıya uygun olarak tanımlar.

Bir programın kuyruk yöneticisiyle iletişim kurabilmesi için, programın kuyruk yöneticisini tanıdığı benzersiz bir tanıtıcısı olmalıdır. Bu tanıtıcı, bazen *Hconn*olarak adlandırılan *bağlantı tanıtıcısı*olarak adlandırılır. CICS programları için bağlantı tanıtıcısı her zaman sıfırdır. Diğer tüm altyapılar ya da program biçemleri için, bağlantı tanıtıcısı MQCONN ya da MQCONNX çağrısı tarafından, program kuyruk yöneticisine bağlandığında döndürülür. Programlar, diğer çağrıları kullandıklarında bağlantı tanıtıcısını giriş parametresi olarak geçirirler.

Bir programın IBM MQ nesnesiyle çalışabilmesi için, programın nesneyi tanıdığı benzersiz bir tanıtıcısı olmalıdır. Bu tanıtıcı, bazen *Hobj*olarak adlandırılan *nesne tanıtıcısı*olarak adlandırılır. Tanıtıcı, program, üzerinde çalışmak üzere nesneyi açtığında MQOPEN çağrısı tarafından döndürülür. Programlar, sonraki MQPUT, MQGET, MQINQ, MQSET ya da MQCLOSE çağrılarını kullandıklarında nesne tanıtıcısını giriş değiştirgesi olarak geçirirler.

Benzer şekilde, MQSUB çağrısı, sonraki MQGET, MQCB ya da MQSUBRQ çağrılarında aboneliği tanımlamak için kullanılan bir *abonelik tanıtıcısı* ya da *Hsub*döndürür ve ileti özelliklerini işleyen belirli çağrılar bir *ileti tanıtıcısı* ya da *Hmsg*kullanır.

Dönüş kodlarının anlaşılması

Her çağrı tarafından çıkış değiştirgeleri olarak bir tamamlanma kodu ve neden kodu döndürülür. Bunlar toplu olarak *dönüş kodları*olarak bilinir.

Bir aramanın başarılı olup olmadığını göstermek için, arama tamamlandığında her arama bir *tamamlanma kodu* döndürür. Tamamlama kodu tipik olarak, başarıyı gösteren MQCC_OK ya da başarısızlığı gösteren MQCC_FAILED ' dir. Bazı çağrılar, kısmen başarılı olduğunu gösteren bir ara durum (MQCC_WARNING) döndürebilir.

Her arama, aramanın başarısız olma nedenini ya da kısmen başarılı olma nedenini gösteren bir *neden kodu* da döndürür. Kuyruğun dolu olması, bir kuyruk için izin verilmeyen alma işlemleri ve kuyruk yöneticisi için tanımlanmayan belirli bir kuyruk gibi durumları kapsayan birçok neden kodu vardır. Programlar, nasıl devam edileceğine karar vermek için neden kodunu kullanabilir. Örneğin, kullanıcılardan giriş verilerini değiştirmelerini isteyebilir, daha sonra yeniden arama yapabilir ya da kullanıcıya bir hata iletisi döndürebilirler.

Tamamlanma kodu MQCC_OK olduğunda, neden kodu her zaman MQRC_NONE olur.

Her aramaya ilişkin tamamlanma ve neden kodları, bu aramanın açıklamasıyla birlikte listelenir. Arama tanımlamaları konusuna bakın ve listeden uygun çağrıyı seçin.

Düzeltici eylem fikirleri de içinde olmak üzere daha ayrıntılı bilgi için bkz:

- z/OS IBM MQ for z/OS iletileri, tamamlama ve neden kodları - IBM MQ for z/OS
- Diğer tüm IBM MQ platformları için İletiler ve neden kodları

Arabelleklerin belirtilmesi

Kuyruk yöneticisi arabelleklere yalnızca gerekliyse başvuruda bulunur. Bir çağrıda arabelleğe gerek duymuyorsanız ya da arabellek sıfır uzunluğuysa, arabellek için boş değerli bir gösterge kullanabilirsiniz.

Gerek duyduğunuz arabelleğin boyutunu belirtirken her zaman veri kalınlığı kullanın.

Bir çağrıdaki çıkışı tutmak için bir arabellek kullandığınızda (örneğin, bir MQGET çağrısına ilişkin ileti verilerini tutmak için ya da MQINQ çağrısı tarafından sorgulanan özniteliklerin değerlerini), belirttiğiniz arabellek geçerli değilse ya da salt okunur saklama alandaysa, kuyruk yöneticisi bir neden kodu döndürmeyi dener. Ancak, her zaman bir neden kodu döndüremeyebilir.

z/OS *z/OS toplu işte dikkat edilmesi gereken noktalar*

MQI 'ı çağıyan z/OS toplu iş programları, gözetmen ya da sorun durumunda olabilir.

Ancak, aşağıdaki koşulları karşılamaları gerekir:

- Hizmet isteği bloğu (SRB) kipinde değil, görev kipinde olmalıdır.
- Bunlar ASC (Primary address space control; Birincil adres alanı denetimi) kipinde olmalıdır (Access Register ASC kipinde değil).
- Bellek çapraz modunda olmamalılar. Birincil adres alan numarası (ASN), ikincil ASN ve ana ASN değerine eşit olmalıdır.
- Bunlar MPF çıkış programları olarak kullanılmamalıdır.
- Hiçbir z/OS kilidi tutulamaz.
- FRU yığınında hiçbir işlev kurtarma yordamı (FRR) olamaz.
- Herhangi bir program durumu sözcüğü (PSW) anahtarı MQCONN ya da MQCONNX çağrısı için yürürlükte olabilir (anahtar TCB anahtarındaki saklama yeri kullanımıyla uyumluysa), ancak MQCONN ya da MQCONNX tarafından döndürülen bağlantı tanıtıcısını kullanan sonraki çağrılar:
  - MQCONN ya da MQCONNX çağrısında kullanılan PSW anahtarına sahip olmalıdır
  - Aynı PSW anahtarı altında erişilebilir (uygunsa yazma için) parametrelere sahip olmalıdır
  - Görevin herhangi bir alt görevinde değil, aynı görev (TCB) altında verilmelidir
- Bunlar 24 bit ya da 31 bit adresleme kipinde olabilir. Ancak, 24 bitlik adresleme kipi geçerliyse, parametre adresleri geçerli 31 bitlik adresler olarak yorumlanmalıdır.

Bu koşullardan herhangi biri karşılanmazsa, bir program denetimi gerçekleşebilir. Bazı durumlarda çağrı başarısız olur ve bir neden kodu döndürülür.

Linux AIX *AIX and Linux önemli noktalar*

AIX and Linux uygulamalarını geliştirirken bilmeniz gereken noktalar.

Linux AIX *AIX and Linux sistemlerinde çatal sistemi çağrısı*

IBM MQ uygulamalarında bir çatal sistem çağrısı kullanırken bu noktalara dikkat edin.

Uygulamanız `fork`' ı kullanmak isterse, bu uygulamanın üst süreci, `MQCONN`gibi IBM MQ çağrıları yapmadan ya da **`ImqQueueManager`**kullanarak bir IBM MQ nesnesi oluşturmadan önce `fork` öğesini aramalıdır.

Uygulamanız herhangi bir IBM MQ çağrısı yaptıktan sonra bir alt süreç oluşturmak isterse, alt öğenin tam bir kopyası değil, yeni bir örnek olduğundan emin olmak için uygulama kodu `fork()` with `exec()` kullanmalıdır.

Uygulamanız exec ( ) kullanmıyorsa, alt süreç içinde yapılan IBM MQ API çağrısı MQRC_ENVIRONMENT_ERROR değerini döndürür.

Linux AIX *AIX and Linux sinyal işleme*

Genel olarak, AIX and Linux sistemleri, iş parçacıklı olmayan (süreç) bir ortamdan çok iş parçacıklı bir ortama taşınmıştır. Birçok durumda, sinyaller ve sinyal işleme, desteklenmesine rağmen, çok iş parçacıklı ortama iyi uymaz ve çeşitli kısıtlamalar vardır.

Genel olarak, AIX and Linux sistemleri, iş parçacıklı olmayan (süreç) bir ortamdan çok iş parçacıklı bir ortama taşınmıştır. İş parçacıklı olmayan ortamda, bazı işlevler yalnızca sinyaller kullanılarak uygulanabilir, ancak çoğu uygulamanın sinyalleri ve sinyal işlemeyi bilmesine gerek yoktu. Çok iş parçacıklı ortamda, iş parçacığı tabanlı temel öğeler, işaretler kullanılarak iş parçacıklı olmayan ortamlarda uygulanmak üzere kullanılan bazı işlevleri destekler.

Birçok durumda, sinyaller ve sinyal işleme, desteklenmesine rağmen, çok iş parçacıklı ortama iyi uymaz ve çeşitli kısıtlamalar vardır. Her birinin sinyalleri işlemeye çalıştığı çok iş parçacıklı bir ortamda uygulama kodunu farklı ara katman yazılımı kitaplıklarıyla (uygulamanın bir parçası olarak çalışan) bütünleştirirken bu sorun ortaya çıkabilir. Bir süreç içinde yalnızca bir yürütme iş parçacığı olduğunda çalışan sinyal işleyicilerinin (işlem başına tanımlanan) kaydedilmesi ve geri yüklenmesinin geleneksel yaklaşımı, çok iş parçacıklı bir ortamda çalışmaz. Bunun nedeni, birçok yürütme iş parçacığının öngörülemeyen sonuçlarla işlem çapındaki bir kaynağı saklamaya ve geri yüklemeye çalışıyor olmasıdır.

Linux AIX *İş parçacıklı olmayan uygulamalar*

Her MQI işlevi, sinyaller için kendi sinyal işleyicisini ayarlar. Bunlar için kullanıcı işleyicileri, MQI işlev çağrısı süresince değiştirilir. Diğer sinyaller, kullanıcı tarafından yazılan işleyiciler tarafından normal şekilde yakalanabilir.

Her MQI işlevi, sinyaller için kendi sinyal işleyicisini ayarlar:

SIGALRM
SIGBUS
SIGFPE
SIGSEGV
SIGILL

Bunlar için kullanıcı işleyicileri, MQI işlev çağrısı süresince değiştirilir. Diğer sinyaller, kullanıcı tarafından yazılan işleyiciler tarafından normal şekilde yakalanabilir. Bir işleyici kurmazsanız, varsayılan işlemler (örneğin, yoksay, çekirdek dökümü ya da çık) yerinde bırakılır.

IBM MQ zamanuyumlu bir sinyal (SIGSEGV, SIGBUS, SIGFPE, SIGILL) işledikten sonra, MQI işlev çağrısından önce sinyali herhangi bir kayıtlı sinyal işleyicisine aktarmayı dener.

Linux AIX *İş parçacıklı uygulamalar*

Bir iş parçacığının MQDISC 'ye kadar MQCONN (ya da MQCONNX) içinden IBM MQ ' e bağlı olduğu varsayılır.

Zamanuyumlu sinyaller

Zamanuyumlu sinyaller belirli bir iş parçacığında ortaya çıkar.

AIX and Linux sistemleri, işlemin tamamı için bu tür sinyaller için bir sinyal işleyicisinin ayarlanmasına güvenle izin verir. Ancak IBM MQ , herhangi bir iş parçacığı IBM MQ' a bağlıyken uygulama işleminde aşağıdaki sinyaller için kendi işleyicisini ayarlar:

SIGBUS
SIGFPE
SIGSEGV
SIGILL

Çok iş parçacıklı uygulamalar yazıyorsanız, her sinyal için tek bir işlem çapında sinyal işleyici vardır. IBM MQ kendi zamanuyumlu sinyal işleyicilerini ayarladığında, her sinyal için önceden kayıtlı işleyicileri kaydeder. IBM MQ listelenen sinyallerden birini işledikten sonra IBM MQ , işlem içindeki ilk IBM MQ bağlantısı sırasında yürürlükte olan sinyal işleyiciyi çağırmayı dener. Önceden kaydedilen işleyiciler, tüm uygulama iş parçacıklarının IBM MQile bağlantısı kesildiğinde geri yüklenir.

Sinyal işleyicileri IBM MQtarafından kaydedilip geri yüklendiğinden, uygulama iş parçacıkları bu sinyaller için sinyal işleyicileri oluşturmamalıdır; aynı işlemin başka bir iş parçacığının da IBM MQ' e bağlanması olasılığı vardır.

Not: Bir uygulama ya da bir ara katman yazılımı kitaplığı (uygulamanın bir parçası olarak çalışan), bir iş parçacığı IBM MQ' e bağlıyken bir sinyal işleyicisi oluşturduğunda, uygulamanın sinyal işleyicisi o sinyalin işlenmesi sırasında ilgili IBM MQ işleyicisini çağırmalıdır.

Sinyal işleyicilerini kurarken ve geri yüklerken, genel ilke, kaydedilecek son sinyal işleyicinin geri yüklenecek ilk kişi olması gerekimidir:

- Bir uygulama IBM MQile bağlantı kurduktan sonra bir sinyal işleyicisi oluşturduğunda, uygulamanın IBM MQile bağlantısını kesmeden önce önceki sinyal işleyicisi geri yüklenmelidir.
- Bir uygulama IBM MQ' a bağlanmadan önce bir sinyal işleyicisi oluşturduğunda, uygulamanın sinyal işleyicisini geri yüklemeden önce IBM MQ ile bağlantısını kesmesi gerekir.

Not: Kaydedilecek son sinyal işleyicinin geri yüklenecek ilk kişi olması gerektiği genel ilkesine uyulmaması, uygulamada beklenmeyen sinyal işlenmesine ve olası olarak uygulama tarafından sinyal kaybına neden olabilir.

Zamanuyumsuz sinyaller

IBM MQ , istemci uygulamaları olmadıkça, iş parçacıklı uygulamalarda zamanuyumsuz sinyaller kullanmaz.

İş parçacıklı istemci uygulamaları için dikkat edilmesi gereken ek noktalar

IBM MQ , bir sunucuya G/Ç sırasında aşağıdaki sinyalleri işler. Bu sinyaller iletişim yığını tarafından tanımlanır. Bir iş parçacığı bir kuyruk yöneticisine bağlıyken uygulama bu sinyaller için bir sinyal işleyici oluşturmamalıdır:

SIGPIPE (TCP/IP için)

Linux AIX *MQI 'da AIX and Linux sinyal işleme kullanılırken dikkat edilmesi gereken diğer noktalar*

AIX and Linuxüzerinde sinyal işlemek için MQI kullanılırken, hızlı yol uygulamaları, sinyal işleyicileri içindeki MQI işlev çağrıları, MQI çağrıları sırasında sinyaller, kullanıcı çıkışları ve kurulabilir hizmetler ve VMS çıkış işleyicileri için dikkat edilmesi gereken başka noktalar vardır.

Fastpath (güvenilir) uygulamaları

Fastpath uygulamaları IBM MQ ile aynı süreçte çalışır ve çok iş parçacıklı ortamda çalışır.

Bu ortamda IBM MQ , SIGSEGV, SIGBUS, SIGFPE ve SIGILL zamanuyumlu sinyallerini işler. Diğer tüm sinyaller, IBM MQ' e bağlıyken Fastpath uygulamasına teslim edilmemelidir. Bunun yerine, uygulama tarafından engellenmeli ya da işlenmelidir. Bir Fastpath uygulaması böyle bir olayı duyarsa, kuyruk yöneticisinin durdurulması ve yeniden başlatılması gerekir ya da tanımlanmamış bir durumda bırakılabilir. MQCONNX altındaki Fastpath uygulamalarına ilişkin kısıtlamaların tam listesi için bkz. “MQCONNX çağrısı kullanılarak bir kuyruk yöneticisiyle bağlantı kurulması” sayfa 708.

İşaret işleyicileri içindeki MQI işlev çağrıları

İşleyici işleyicisindeyken MQI işlevini çağırmayın.

Başka bir MQI işlevi etkinken bir sinyal işleyicisinden MQI işlevini çağırmaya çalışırsanız, MQRC_CALL_IN_PROGRESS döndürülür. Başka bir MQI işlevi etkin olmadığı halde bir sinyal işleyicisinden MQI işlevini çağırmaya çalışırsanız, yalnızca seçici çağrıların bir işleyiciden ya da işleyicinin içinden yayınlanabileceği işletim sistemi kısıtlamaları nedeniyle bu, işlem sırasında bir zaman başarısız olabilir.

Program çıkışı sırasında otomatik olarak çağrılabilecek C++ yıkıcı yöntemleri için, MQI işlevlerinin çağrılmasını durduramayabilirsiniz. MQRC_CALL_IN_PROGRESS ile ilgili hataları yoksayın. Bir sinyal işleyici exit () çağrısında bulunursa, IBM MQ eşitleme noktasında kesinleştirilmemiş iletileri her zamanki gibi geri çevirip açık kuyrukları kapatır.

MQI çağrıları sırasında sinyaller

MQI işlevleri, EINTR kodunu ya da uygulama programlarının eşdeğerini döndürmez.

MQI çağrısı sırasında bir sinyal oluşursa ve işleyici *return*çağrısında bulunursa, çağrı, sinyal gerçekleşmemiş gibi çalışmaya devam eder. Özellikle, MQGET, denetimi uygulamaya hemen döndürmek için bir sinyal tarafından kesilemez. Bir MQGET ' den çıkmak istiyorsanız, kuyruğu GET_DISABLED olarak ayarlayın; diğer bir seçenek olarak, sınırlı bir süre sonu olan MQGET çağrısına ilişkin bir döngü kullanın (gmo.WaitInterval ayarlanmış MQGMO_WAIT) ve döngüyü kesen bir işareti ayarlamak için sinyal işleyicinizi (iş parçacıklı olmayan bir ortamda) ya da eşdeğeri bir ortamda kullanın.

AIX AIX ortamında IBM MQ , sinyallerin kesintiye uğradığı sistem çağrılarını yeniden başlatmanızı gerektirir. Kendi işaret işleyicinizi sigaction (2) ile kurarken, yeni eylem yapısının sa_flags alanında SA_RESTART işaretini ayarlayın; tersi durumda IBM MQ bir sinyal tarafından kesilen çağrıyı tamamlayamayabilir.

Kullanıcı çıkışları ve kurulabilir hizmetler

Çok iş parçacıklı bir ortamda IBM MQ sürecinin bir parçası olarak çalışan kullanıcı çıkışları ve kurulabilir hizmetler, fastpath uygulamalarıyla aynı kısıtlamalara sahiptir. Bunların IBM MQ ' e kalıcı olarak bağlı olduğunu ve bu nedenle sinyalleri ya da iş parçacığı korumalı olmayan işletim sistemi çağrılarını kullanmadığını göz önünde bulundurun.

Kuyruk yöneticisine bağlanma ve bağlantı kesme

IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

Bu bağlantının nasıl kurulacağı, programın çalıştığı platforma ve ortama bağlıdır:

Multi **IBM MQ for Multiplatforms**

Bu ortamlarda çalışan programlar, bağlanmak için MQCONN MQI çağrıyı ve bağlantı kesilmek için MQDISC çağrıyı kullanabilir. Diğer bir seçenek olarak, programlar MQCONNX çağrılarını kullanabilir.

z/OS **IBM MQ for z/OS toplu**

Bu ortamda çalışan programlar, bağlanmak için MQCONN MQI çağrıyı ve bağlantı kesilmek için MQDISC çağrıyı kullanabilir. Diğer bir seçenek olarak, programlar MQCONNX çağrılarını kullanabilir.

z/OS toplu iş programları, aynı TCB ' de birden çok kuyruk yöneticisine ardışık ya da eşzamanlı olarak bağlanabilir.

z/OS **IMS**

IMS denetim bölgesi başlatıldığında bir ya da daha çok kuyruk yöneticilerine bağlanır. Bu bağlantı IMS komutlarıyla denetlenir. z/OSüzerinde IMS bağdaştırıcısının nasıl denetleneceğine ilişkin bilgi için bkz. IBM MQ for z/OS Yönetimi. Ancak, IMS programlarını kuyruğa alan ileti yazıcıları, bağlanmak istedikleri kuyruk yöneticisini belirtmek için MQCONN MQI çağrıyı kullanmalıdır. MQDISC çağrıyı kullanarak o kuyruk yöneticisiyle bağlantıyı kesebilirler.

Bir eşitleme noktası oluşturan bir IMS çağrısının ardından ve başka bir kullanıcı için bir ileti işlemeden önce IMS bağdaştırıcısı, uygulamanın tanıtıcıları kapatmasını ve kuyruk yöneticisiyle bağlantısını kesmesini sağlar. Bkz. “IMS uygulamalarında eşitleme noktaları” sayfa 821.

IMS programları, aynı TCB ' de birden çok kuyruk yöneticisine ardışık ya da eşzamanlı olarak bağlanabilir.

z/OS CICS Hareket Sunucusu- z/OS

CICS sisteminin kendisi bağlı olduğundan, CICS programlarının bir kuyruk yöneticisine bağlanmak için herhangi bir iş yapması gerekmez. Bu bağlantı genellikle başlatma sırasında otomatik olarak yapılır, ancak IBM MQ for z/OSile verilen CKQC hareketini de kullanabilirsiniz. CKQC hakkında daha fazla bilgi için bkz. IBM MQ for z/OS Yönetimi.

CICS görevleri yalnızca CICS bölgesinin bağlı olduğu kuyruk yöneticisine bağlanabilir.

CICS programları MQI bağlanma ve bağlantı kesme çağrılarını da kullanabilir (MQCONN ve MQDISC). Bunu, bu uygulamaları en az kurtarma ileCICS dışı ortamlara bağlamanız için yapmak isteyebilirsiniz. Ancak bu çağrılar, CICS ortamında *her zaman* başarıyla tamamlanır. Bu, dönüş kodunun kuyruk yöneticisine yönelik bağlantının gerçek durumunu yansıtmayabileceği anlamına gelir.

TXSeries for Windows and Open Systems (ve Açık Sistemler için)

CICS sisteminin kendisi bağlı olduğundan, bu programların bir kuyruk yöneticisine bağlanmak için herhangi bir iş yapması gerekmez. Bu nedenle, aynı anda yalnızca bir bağlantı desteklenir. CICS uygulamaları, bağlantı tanıtıcısı elde etmek için bir MQCONN çağrısı ve çıkmadan önce bir MQDISC çağrısı yayınlamalıdır.

Bir kuyruk yöneticisine bağlanma ve bağlantı kesme hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “MQCONN çağrısı kullanılarak bir kuyruk yöneticisiyle bağlantı kurulması” sayfa 707
- “MQCONNX çağrısı kullanılarak bir kuyruk yöneticisiyle bağlantı kurulması” sayfa 708
- “MQDISC kullanarak programların kuyruk yöneticisiyle bağlantısını kesme” sayfa 711

İlgili kavramlar

“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

“Nesnelerin açılması ve kapatılması” sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

“Kuyruktaki iletilerin konması” sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

“Kuyruktan ileti alma” sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

“Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

“Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

“MQI ve kümelerle çalışma” sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

“IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

“IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları” sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

MQCONN çağrısı kullanılarak bir kuyruk yöneticisiyle bağlantı kurulması

MQCONN çağrısı kullanılarak bir kuyruk yöneticisine nasıl bağlanılacağını öğrenmek için bu bilgileri kullanın.

Genel olarak, belirli bir kuyruk yöneticisine ya da varsayılan kuyruk yöneticisine bağlanabilirsiniz:

- IBM MQ for z/OSiçin, toplu iş ortamında varsayılan kuyruk yöneticisi CSQBDEFV modülünde belirtilir.
- IBM MQ for Multiplatformsiçin, mqs.ini dosyasında varsayılan kuyruk yöneticisi belirtilir.

Diğer bir seçenek olarak, z/OS MVS toplu iş, TSO ve RRS ortamlarında, bir kuyruk paylaşım grubu içindeki herhangi bir kuyruk yöneticisine bağlanabilirsiniz. MQCONN ya da MQCONNX isteği, grubun etkin üyelerinden birini seçer.

Bir kuyruk yöneticisine bağlandığınızda, görevin yerel olması gerekir. IBM MQ uygulamasıyla aynı sisteme ait olmalıdır.

IMS ortamında, kuyruk yöneticisinin IMS denetim bölgesine ve programın kullandığı bağımlı bölgeye bağlanması gerekir. IBM MQ for z/OS kurulduğunda CSQQDEFV modülünde varsayılan kuyruk yöneticisi belirtilir.

TXSeries CICS ortamı ve Windows ve AIXiçin TXSeries ile, kuyruk yöneticisi CICSiçin bir XA kaynağı olarak tanımlanmalıdır.

Varsayılan kuyruk yöneticisine bağlanmak için, tamamen boşluklardan oluşan ya da boş (X'00 ') bir karakterle başlayan bir ad belirterek MQCONN' u çağırın.

Bir uygulamanın bir kuyruk yöneticisine başarıyla bağlanması için yetkisi olmalıdır. Daha fazla bilgi için bkz. Güvenlik.

MQCONN çıkışı:

- Bağlantı tanıtıcısı (**Hconn**)
- Tamamlanma kodu
- Neden kodu

Sonraki MQI çağrılarında bağlantı tanıtıcısını kullanın.

Neden kodu, uygulamanın o kuyruk yöneticisine zaten bağlı olduğunu gösteriyorsa, döndürülen bağlantı tanıtıcısı, uygulama ilk bağlandığında döndürülen bağlantı tanıtıcısıyla aynıdır. Çağıran uygulama bağlı kalmayı beklediğinden, uygulama bu durumda MQDISC çağrısına izin vermemelidir.

Bağlantı tanıtıcısının kapsamı, nesne tanıtıcısının kapsamıyla aynıdır (bkz. “MQOPEN çağrıyı kullanarak nesneleri açma” sayfa 713).

Değiştirgelerin tanımları, MQCONNiçindeki MQCONN çağrısının tanımında verilir.

Çağrıyı yayınladığınızda kuyruk yöneticisi susturma durumundaysa ya da kuyruk yöneticisi sona eriyorsa MQCONN çağrısı başarısız olur.

MQCONN ya da MQCONNX Kapsamı

Bir MQCONN ya da MQCONNX çağrısının kapsamı, genellikle bunu yayınlayan iş parçacığıdır. Yani, çağrıdan döndürülen bağlantı tanıtıcısı yalnızca, çağrıyı yayınlayan iş parçacığı içinde geçerlidir. Aynı anda, tanıtıcı kullanılarak yalnızca bir arama yapılabilir. Farklı bir iş parçacığından kullanılırsa, geçersiz olduğu için reddedilir. Uygulamanızda birden çok iş parçacığı varsa ve her biri IBM MQ çağrılarını kullanmak istiyorsa, her biri MQCONN ya da MQCONNX yayınlamalıdır.

Bir süreç birden çok MQCONN çağrısı yaptığında, her bir çağrı aynı kuyruk yöneticisine yapılmasına gerek yoktur. Ancak, bir iş parçacığından aynı anda yalnızca bir IBM MQ bağlantısı kurulabilir. Diğer bir seçenek olarak, “MQCONNX ile paylaşılan (iş parçacığından bağımsız) bağlantılar” sayfa 709 ' i tek bir iş parçacığından birden çok IBM MQ bağlantısına ve herhangi bir iş parçacığından IBM MQ bağlantısına izin vermek için kullanabilirsiniz.[7]

Uygulamanız bir istemci olarak çalışıyorsa, bir iş parçacığı içinde birden çok kuyruk yöneticisine bağlanabilir.

MQCONNX çağrısı kullanılarak bir kuyruk yöneticisiyle bağlantı kurulması

MQCONNX çağrısı MQCONN çağrısına benzer, ancak aramanın çalışma şeklini denetleme seçeneklerini içerir.

MQCONNX 'e giriş olarak z/OS ya da z/OS paylaşılan kuyruk sistemlerinde bir kuyruk paylaşım grubu adıkuyruk yöneticisi adını girebilirsiniz. Kuyruk yöneticisiyle bağlantının nasıl sağlandığını denetleme seçenekleri, MQCNOadlı bir yapıda sağlanır.

MQCONNX çıkışı:

- Bağlantı tanıtıcısı (Hconn)
- Tamamlanma kodu
- Neden kodu

Sonraki MQI çağrılarında bağlantı tanıtıcısını kullanırsınız.

MQCNO yapısının *Options* alanında ayarlanan bağlanma seçenekleri, bağlantının birkaç özniteliğinin denetlenmesine olanak sağlar. Belirli bir not aşağıdaki seçenek gruplarıdır:

- Bağlama seçenekleri, *güvenilir uygulamalar* yaratılmasına olanak sağlar. Güvenilir uygulamalar, IBM MQ uygulamasının ve yerel kuyruk yöneticisi aracısının aynı işlem olduğunu belirtir. Aracı işleminin artık kuyruk yöneticisine erişmek için bir arabirim kullanması gerekmediğinden, bu uygulamalar kuyruk yöneticisinin bir uzantısı haline gelir. Bu davranış, MQCNO_FASTPATH_BINDING seçeneği belirtilerek istenmiştir. Güvenilir uygulamalar için geçerli olan kısıtlamalarla ilgili daha fazla bilgi için bkz. “Güvenilir uygulamalar için kısıtlamalar” sayfa 708.
- Tanıtıcı paylaşım seçenekleri, paylaşılan bağlantıların yaratılmasına izin verir. Paylaşılan bağlantılar, aynı işlem içindeki farklı iş parçacıkları arasında tanıtıcıları paylaşabilir. Paylaşılan bağlantılar hakkında daha fazla bilgi için bkz. “MQCONNX ile paylaşılan (iş parçacığından bağımsız) bağlantılar” sayfa 709.

MQCNO, uygulamanın kuyruk yöneticisine yönelik bağlantının nasıl doğrulanacağını denetlemesini de sağlar. Kimlik doğrulama kimlik bilgileri, MQCNO yapısından gönderme yapılan bir MQCSP yapısında belirtilebilir.

MQCONNX çağrısına ilişkin değiştirgelerin ve denetlenebilen bağlantı özniteliklerinin tam açıklaması için MQCONNX-Connect kuyruk yöneticisi (genişletilmiş)başlıklı konuya bakın.

Güvenilir uygulamalar için kısıtlamalar

Güvenilir uygulamalar için geçerli olan kısıtlamalar. Bazı kısıtlamalar tüm platformlar için geçerlidir ve diğerleri platforma özgüdür.

T

- Güvenilir uygulamaların kuyruk yöneticisiyle bağlantısını belirtik olarak kesmeniz gerekir.
- Kuyruk yöneticisini **endmqm** komutuyla sona erdirmeden önce güvenilir uygulamaları durdurmalısınız.
- MQCNO_FASTPATH_BINDING ile zamanuyumsuz sinyaller ve süreölçer kesintileri (`sigkill`gibi) kullanmamalısınız.
- Tüm altyapılarda, güvenilen bir uygulamadaki bir iş parçacığı bir kuyruk yöneticisine bağlanamazken, aynı işlemdeki başka bir iş parçacığı farklı bir kuyruk yöneticisine bağlı.
- Linux AIX AIX and Linux sistemlerinde, tüm MQI çağrıları için etkin userID ve groupID olarak mqm kullanmalısınız. Bu kimlikleri, kimlik doğrulaması gerektiren MQI dışı bir çağrı (örneğin, bir dosya açmak) yapmadan önce değiştirebilirsiniz, ancak sonraki MQI çağrısından önce bu kimlikleri mqm olarak değiştirmeniz gerekir.
- IBM i IBM i'ta:

[7] IBM MQ for AIX or Linux sistemleriyle çok iş parçacıklı uygulamalar kullanırken, uygulamaların iş parçacıkları için yeterli yığın boyutuna sahip olduğundan emin olmanız gerekir. Çok iş parçacıklı uygulamalar kendi başlarına ya da diğer sinyal işleyicileriyle (örneğin, CICS) MQI çağrıları yaparken 256 KB ya da daha büyük bir yığın boyutu kullanmayı düşünün.

1. Güvenilir uygulamalar, QMQM kullanıcı profili altında çalıştırılmalıdır. Kullanıcı tanıtımının QMQM grubunun üyesi olması ya da programın QMQM yetkisini benimsemesi yeterli değildir. QMQM kullanıcı tanıtımının etkileşimli işlerde oturum açmak için kullanılması ya da güvenilir uygulamalar çalıştıran işler için iş tanımlamasında belirtilmesi mümkün olmayabilir. Bu durumda, IBM MQ programları çalışırken işin yürürlükteki kullanıcısını geçici olarak QMQM olarak değiştirmek için IBM i tanıtım değiş tokuş API işlevlerini, QSYGETPH, QWTSETP ve QSYRLSPH ' yi kullanmak bir yaklaşımdır. Bu işlevlerin ayrıntıları, kullanımlarına ilişkin bir örnekle birlikte, IBM i*Uygulama programlama arabirimleri* belgelerinin Güvenlik API ' leri bölümünde sağlanır.
2. Güvenilen uygulamaları System-Request Option 2 seçeneğini kullanarak ya da ENDJOB komutunu kullanarak çalıştırdıkları işleri sona erdirerek iptal etmeyin.

- ALW AIX, Linux, and Windows sistemlerinde güvenilir 32 bit uygulamalar desteklenmez. Güvenilir bir 32 bit uygulamayı çalıştırmayı denerseniz, bu uygulama standart bağlı bir bağlantıya indirgenir.

MQCONNX ile paylaşılan (iş parçacığından bağımsız) bağlantılar
MQCONNX ile Paylaşılan bağlantılar ve dikkate alınacak bazı kullanım notları hakkında bilgi edinmek için bu bilgileri kullanın.

Not: IBM MQ for z/OSüzerinde desteklenmez.

IBM MQ for z/OSdışındaki IBM MQ platformlarında, MQCONN ile yapılan bir bağlantı yalnızca bağlantıyı yapan iş parçacığı için kullanılabilir. MQCONNX çağrısındaki seçenekler, bir işlemdeki tüm iş parçacıkları tarafından paylaşılabilecek bir bağlantı yaratmanızı sağlar. Uygulamanız aynı iş parçacığında MQI çağrıları verilmesini gerektiren bir hareket ortamında çalışıyorsa, aşağıdaki varsayılan seçeneği kullanmalısınız:

MQCNO_HANDLE_SHARE_NONE
Paylaşılmayan bir bağlantı yaratır.

Diğer ortamların çoğunda, aşağıdaki iş parçacığından bağımsız, paylaşılan bağlantı seçeneklerinden birini kullanabilirsiniz:

MQCNO_HANDLE_SHARE_BLOCK
Paylaşılan bir bağlantı yaratır. Bir MQCNO_HANDLE_SHARE_BLOCK bağlantısında, bağlantı başka bir iş parçacığında MQI çağrısı tarafından kullanıyorsa, MQI çağrısı yürürlükteki MQI çağrısı tamamlanıncaya kadar bekler.

MQCNO_HANDLE_SHARE_NO_BLOCK
Paylaşılan bir bağlantı yaratır. Bir MQCNO_HANDLE_SHARE_NO_BLOCK bağlantısında, bağlantı başka bir iş parçacığında MQI çağrısı tarafından kullanıyorsa, MQI çağrısı MQRC_CALL_IN_PROGRESSnedeniyle hemen başarısız olur.

MTS (Microsoft Transaction Server) ortamı dışında varsayılan değer MQCNO_HANDLE_SHARE_NONE' dir. MTS ortamında varsayılan değer MQCNO_HANDLE_SHARE_BLOCK' dir.

MQCONNX çağrısından bir bağlantı tanıtıcısı döndürülür. Tanıtıcı, işlemdeki herhangi bir iş parçacığından gelen sonraki MQI çağrıları tarafından kullanılabilir ve bu çağrılar MQCONNXtarafından döndürülen tanıtıcı ile ilişkilendirilir. Tek bir paylaşılan tanıtıcı kullanan MQI çağrıları, iş parçacıkları arasında diziselleştirilir.

Örneğin, paylaşılan bir tanıtıcı ile aşağıdaki etkinlik sırası gerçekleştirilebilir:

1. İş parçacığı 1 sorunları MQCONNX ve paylaşılan bir tanıtıcı *h1* alır
2. İş parçacığı 1 bir kuyruk açar ve *h1* kullanarak bir alma isteği yayınlar
3. İş parçacığı 2, *h1* kullanarak bir koyma isteği yayınlar
4. İş parçacığı 3, *h1* kullanarak bir koyma isteği yayınlar
5. MQDISC *h1* kullanılarak iş parçacığı 2 sorunları

Tanıtıcı herhangi bir iş parçacığı tarafından kullanılırken, bağlantıya erişim diğer iş parçacıkları tarafından kullanılamaz. Bir iş parçacığının başka bir iş parçacığından gelen önceki çağrıların tamamlanmasını beklediği durumlarda, MQCONNX seçeneğini MQCNO_HANDLE_SHARE_BLOCKile kullanın.

Ancak engelleme zorluklara neden olabilir. “2” sayfa 709. adımda 1. iş parçacığının, henüz ulaşmamış olabilecek iletileri bekleyen bir alma isteği yayınladığını varsayın (bekleme ile alma). Bu durumda, iş parçacığı 1 'deki alma isteği sürdüğü sürece, iş parçacığı 2 ve 3 de beklemeye (engellendi) bırakılır. Tanıtıcıda başka bir MQI çağrısı çalışıyorsa, MQI çağrısını hatayla döndürmeyi tercih ediyorsanız, MQCONNX seçeneğini MQCNO_HANDLE_SHARE_NO_BLOCKile kullanın.

Paylaşılan bağlantı kullanım notları

1. Bir nesneyi açarak oluşturulan herhangi bir nesne tutamacı (Hobj) bir Hconn ile ilişkilendirilir; bu nedenle paylaşılan bir Hconn için Hobjs, Hconn 'u kullanan herhangi bir iş parçacığı tarafından paylaşılır ve kullanılabilir. Benzer şekilde, bir Hconn altında başlatılan herhangi bir iş birimi o Hconn ile ilişkilendirilir; bu nedenle bu da paylaşılan Hconn ile iş parçacıkları arasında paylaşılır.
2. *Herhangi bir* iş parçacığı, yalnızca ilgili MQCONNX 'i çağıRAN iş parçacığını değil, paylaşılan bir Hconn bağlantısını kesmek için MQDISC ' yi çağırabilir. MQDISC, Hconn 'u tüm iş parçacıkları için kullanılamaz kılarak sonlandırır.
3. Tek bir iş parçacığı dizisel olarak birden çok paylaşılan Hconn kullanabilir; örneğin, paylaşılan bir Hconn 'un altına bir ileti koymak için MQPUT kullanın ve her işlem farklı bir yerel iş birimi altında olmak üzere başka bir paylaşılan Hconn kullanarak başka bir ileti yerleştirin.
4. Paylaşılan Hconn 'lar genel bir iş biriminde kullanılamaz.

Multi *MQ_CONNECT_TYPE ile MQCONNX çağrı seçeneklerinin kullanılması*

Farklı MQCONNX çağrı seçeneklerini ve bunların **MQ_CONNECT_TYPE** ortam değişkeniyle nasıl kullanıldığını anlamak için bu bilgileri kullanın.

Not: MQ_CONNECT_TYPE ' in yalnızca STANDARD bağ tanımları için etkisi vardır. Diğer bağlamalar için **MQ_CONNECT_TYPE** yoksayılır.

IBM MQ for Multiplatformsüzerinde, bir MQCONNX çağrısında kullanılan MQCNO yapısının Options alanında belirtilen bağ tanımı tipiyle birlikte **MQ_CONNECT_TYPE** ortam değişkenini kullanabilirsiniz.

*Çizelge 113. MQCONNX çağrı seçeneklerinin **MQ_CONNECT_TYPE** ortam değişkeniyle nasıl kullanıldığı*

MQCONNX çağrı seçeneği	MQ_CONNECT_TYPE ortam değişkeni	Sonuç
Standart	Tanımlı değil	Standart
Standart	Standart	Standart
Standart	FastPath	Standart
Standart	CLIENT	CLIENT
Standart	LOCAL	Standart

MQCNO_STANDARD_BINDING belirtilmediyse, varsayılan olarak MQCNO_STANDARD_BINDING değerini kullanan MQCNO_NONE değerini kullanabilirsiniz.

MQCONN ve MQCONNX için Kimlik Doğrulama ve Kimlik

Uygulamaların IBM MQ' e bağlandıklarında kimlik doğrulaması için kullanılan kimlik bilgilerini nasıl sağlayabileceğini öğrenmek için bu bilgileri kullanın.

Varsayılan kullanıcı kimliği

Bir uygulama IBM MQ ' a MQCONN ya da MQCONNX ile bağlanmak için ileti kuyruğu arabirimini (MQI) kullandığında, her zaman bir kullanıcı kimliği oluşturulur ve bağlantıyla ilişkilendirilir.

Varsayılan olarak, ilk kullanıcı kimliği her zaman uygulamanın altında çalıştığı işletim sistemi tarafından kullanılır. Bu ilk kimlik, yerel olarak bağlı ya da güvenilir uygulama bağlantıları için yeterli olabilir.

Bir uygulama MQCONN çağrısıyla bir kuyruk yöneticisine bağlandığında, uygulama varsayılan kullanıcı kimliğini değiştiremez. Ancak, aşağıdaki mekanizmalar bağlantıyla ilişkili kullanıcı kimliğini değiştirebilir:

- İstemci tarafı ya da sunucu tarafı güvenlik çıkışı.
- Kuyruk yöneticisinde kanal kimlik doğrulama kuralları.
- TLS karşılıklı kimlik doğrulaması sırasında oluşturulan bir istemci kullanıcı kimliği.

Kimlik bilgilerini sağlamak için MQCONNX kullanılıyor

MQCONNX, bir uygulamaya bağlantıyla ilişkili kimlik üzerinde daha fazla denetim sağlar. Bir uygulama, MQCONNX 'e **ConnectOpts** değiştirgesinde belirtilen bağlanma seçeneklerinin bir parçası olarak bir MQCSP yapısı sağlayabilir. MQCSP yapısı, bir kullanıcı kimliği oluşturmak için kullanılan kimlik bilgilerini içerebilir. IBM MQ , MQCSP yapısında aşağıdaki kimlik bilgilerini destekler:

- Bir kullanıcı kimliği ve parola.
- V 9.3.4 Uygulama AIX ya da Linux sistemlerinde çalışan bir kuyruk yöneticisine bağlanıyorsa, IBM MQ 9.3.4' den bir kimlik doğrulama simgesi.

Kuyruk yöneticisinin bağlantı kimlik doğrulaması ve kanal kimlik doğrulaması yapılandırması, bir uygulama tarafından sağlanan kimlik bilgilerinin nasıl işleneceğini denetler. Örneğin, bu yapılandırma aşağıdaki yönleri etkiler:

- MQCSP yapısındaki kimlik bilgilerinin doğrulanıp doğrulanmadığını ve bunların nasıl doğrulandığını belirtir.
- MQCSP yapısındaki kimlik bilgilerindeki kullanıcı kimliğinin başka bir kullanıcı kimliğiyle eşlenip eşlenmediğini belirler.
- Daha sonra, kimliği doğrulanmış kullanıcının uygulamanın bağlamı olarak kullanılıp kullanılmayacağını belirler.

Bağlantı kimlik doğrulamasıyla ilgili daha fazla bilgi için bkz. Bağlantı kimlik doğrulaması. Kanal kimlik doğrulamasıyla ilgili daha fazla bilgi için bkz. Kanal kimlik doğrulama kayıtları.

MQI kullanan C dilinde yazılmış örnek programlardan bazıları, kimlik doğrulama kimlik bilgilerini sağlamak için MQCSP yapısının nasıl kullanıldığını gösterir. Ek bilgi için aşağıdaki örnek programlara bakın:

- “"Get" örnek programları” sayfa 1045
- “Put örnek programları” sayfa 1057
- “Tarayıcı örnek programı” sayfa 1033
- “TLS örnek programı” sayfa 1072

İlgili bilgiler

MQCSP yapısını kullanarak kullanıcıların belirlenmesi ve doğrulanmaları

MQCSP-Güvenlik değiştirgeleri

Kullanıcıların tanımlanması ve kimliğinin doğrulanmesi

MQDISC kullanarak programların kuyruk yöneticisiyle bağlantısını kesme

MQDISC kullanarak bir kuyruk yöneticisiyle programların bağlantısını kesmeye ilişkin bilgi edinmek için bu bilgileri kullanın.

MQCONN ya da MQCONNX çağrısıyla bir kuyruk yöneticisine bağlanan bir program kuyruk yöneticisiyle tüm etkileşimi tamamladığında, MQDISC çağrısıyla bağlantıyı keser:

- CICS Transaction Server for z/OS uygulamalarında, MQCONNX kullanılmadığı sürece çağrı isteğe bağlıdır ve uygulama sona ermeden önce bağlantı etiketini atmak istersiniz.
- IBM MQ for IBM i sistemlerinde, işletim sisteminden oturum kapattığınızda örtük bir MQDISC çağrısı yapılır.

MQDISC çağrısına giriş olarak, kuyruk yöneticisine bağlandığınızda MQCONN ya da MQCONNX tarafından döndürülen bağlantı tanıtıcısını (Hconn) sağlamanız gerekir.

CICS on z/OSdışında, MQDISC çağrıldıktan sonra bağlantı tanıtıcısı (Hconn) artık geçerli değildir ve MQCONN ya da MQCONNX 'i yeniden çağırıncaya kadar başka MQI çağrıları yayınlayamazsınız. MQDISC, bu tanıtıcıyı kullanarak hala açık olan nesneler için örtük bir MQCLOSE yapar.

z/OS z/OS' e bağlı bir istemci için, bir MQDISC çağrısı yayınlandığında örtük bir kesinleştirme gerçekleşir, ancak açık olan kuyruk tanıtıcıları kanal gerçekten sona erinceye kadar kapatılmaz.

IBM MQ for z/OSüzerinde bağlanmak için MQCONNX kullanırsanız, MQDISC MQCONNX tarafından oluşturulan bağlantı etiketinin kapsamını da sonlandırır. Ancak, bir CICS, IMSya da RRS uygulamasında, bir bağlantı etiketiyle ilişkili etkin bir kurtarma birimi varsa, MQDISC, MQRC_CONN_TAG_NOT_ALLOWED neden koduyla reddedilir.

Değiştirgelerin tanımları, MQDISC' deki MQDISC çağrısının tanımında verilir.

MQDISC yayınlanmadığı zaman

Yaratma iş parçacığı sona erdiğinde standart, paylaşılmayan bağlantı (Hconn) temizlenir. Paylaşılan bağlantı yalnızca, tüm işlem sona erdiğinde örtük olarak geriletir ve bağlantısı kesilir. Paylaşılan Hconn 'u yaratan iş parçacığı, Hconn var olmaya devam ederken sona ererse, Hconn hala kullanılabilir.

Yetki denetimi

MQCLOSE ve MQDISC çağrıları genellikle yetki denetimi gerçekleştirmez.

Olağan durumlarda, IBM MQ nesnesini açma ya da bu nesneye bağlanma yetkisi olan bir iş o nesneyi kapatır ya da o nesneyle bağlantısını keser. Bir IBM MQ nesnesine bağlanan ya da nesneyi açan bir işin yetkisi iptal edilse bile, MQCLOSE ve MQDISC çağrıları kabul edilir.

Nesnelerin açılması ve kapatılması

Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

Aşağıdaki işlemlerden herhangi birini gerçekleştirmek için önce *aç* ilgili IBM MQ nesneyi gerçekleştirmeniz gerekir:

- İletilerin kuyruğa konması
- Kuyruktan ileti alma (göz atma ya da alma)
- Bir nesnenin özniteliklerini ayarlama
- Herhangi bir nesnenin özniteliklerini sorma

Nesne ile ne yapmak istediğinizi belirtmek için çağrı seçeneklerini kullanarak nesneyi açmak için MQOPEN çağrısına kullanın. Tek kural dışı durum, kuyruğa tek bir ileti koymak istiyorsanız, kuyruğu hemen kapatın. Bu durumda, MQPUT1 çağrısıyla *açılış* aşamasını atlayabilirsiniz (bkz. “MQPUT1 çağrısı kullanılarak kuyruğa bir ileti konması” sayfa 730).

MQOPEN çağrısıyla bir nesneyi açmadan önce programınızı bir kuyruk yöneticisine bağlamanız gerekir. Bu, “Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705içinde tüm ortamlar için ayrıntılı olarak açıklanır.

Açabileceğiniz dört tip IBM MQ nesnesi vardır:

- Kuyruk
- Ad listesi
- Süreç tanımlaması
- Kuyruk yöneticisi

MQOPEN çağrıyı kullanarak tüm bu nesneleri benzer bir şekilde açarsınız. IBM MQ nesneleriyle ilgili ek bilgi için Nesne tipleribaşlıklı konuya bakın.

Aynı nesneyi bir kereden fazla açabilir ve her yeni bir nesne tanıtıcısı aldığınızda açabilirsiniz. Bir tanıtıcı kullanarak kuyruktaki iletilere göz atmak ve başka bir tanıtıcı kullanarak aynı kuyruktaki iletileri kaldırmak

isteyebilirsiniz. Bu, aynı nesneyi kapatmak ve yeniden açmak için kaynakları kullanarak tasarruf sağlar. Aynı zamanda *ve* iletilerine göz atmak üzere bir kuyruk açabilirsiniz.

Ayrıca, tek bir MQOPEN ile birden çok nesne açabilir ve bunları MQCLOSE kullanarak kapatabilirsiniz. Bunun nasıl yapılacağını öğrenmek için bkz. "Dağıtım listeleri" sayfa 732 .

Bir nesneyi açma girişiminde bulunduğunuzda, kuyruk yöneticisi, MQOPEN çağrısında belirttiğiniz seçenekler için o nesneyi açma yetkiniz olup olmadığını denetler.

Bir program kuyruk yöneticisiyle bağlantısını kestiğinde nesneler otomatik olarak kapatılır. IMS ortamında, bir program GU (get unique) IMS çağrısını izleyen yeni bir kullanıcı için işlem başlattığında bağlantıyı kesmeye zorlanır. IBM i platformunda, bir iş sona erdiğinde nesneler otomatik olarak kapatılır.

Açtığınız nesneleri kapatmak iyi bir programlama uygulamasıdır. Bunu yapmak için MQCLOSE çağrıyı kullanın.

Nesneleri açma ve kapatma hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- "MQOPEN çağrıyı kullanarak nesneleri açma" sayfa 713
- "Dinamik kuyruklar yaratılması" sayfa 720
- "Uzak kuyrukların açılması" sayfa 721
- "MQCLOSE çağrısı kullanılarak nesnelerin kapatılması" sayfa 721

İlgili kavramlar

"İleti Kuyruğu Arabirimine Genel Bakış" sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

"Kuyruk yöneticisine bağlanma ve bağlantı kesme" sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

"Kuyruktaki iletilerin konması" sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

"Kuyruktan ileti alma" sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

"Nesne öznitelikleri hakkında sorma ve bunları ayarlama" sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

"İş birimlerinin kesinleştirilmesi ve geri çekilmesi" sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

"Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması" sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

"MQI ve kümelerle çalışma" sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

"IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma" sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

"IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları" sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

MQOPEN çağrıyı kullanarak nesneleri açma

MQOPEN çağrısı kullanılarak nesne açılmasıyla ilgili bilgi edinmek için bu bilgileri kullanın.

MQOPEN çağrısına giriş olarak şunları sağlamanız gerekir:

- Bağlantı tanıtıcısı. z/OSüzerindeki CICS uygulamaları için, değişmez MQHC_DEF_HCONN değerini (sıfır değerine sahip) belirtebilir ya da MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını

kullanabilirsiniz. Diğer programlar için her zaman MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını kullanın.
- Açmak istediğiniz nesnenin tanımı, nesne tanımlayıcı yapısını (MQOD) kullanarak.
- Çağrı işlemini denetleyen bir ya da daha fazla seçenek.

MQOPEN çıkışı:

- Nesneye erişiminizi gösteren bir nesne tanıtıcısı. Sonraki MQI çağrılarına girişte bunu kullanın.
- Dinamik bir kuyruk yaratıyorsanız (ve altyapınızda destekleniyorsa), değiştirilmiş bir nesne tanımlayıcı yapısı.
- Bir tamamlanma kodu.
- Bir neden kodu.

Nesne tanıtıcısının kapsamı

Nesne tanıtıcısının (Hobj) kapsamı, bağlantı tanıtıcısının (Hconn) kapsamıyla aynıdır.

Bu, "MQCONN ya da MQCONNX Kapsamı" sayfa 707 ve "MQCONNX ile paylaşılan (iş parçacığından bağımsız) bağlantılar" sayfa 709ile kaplıdır. Ancak, bazı ortamlarda göz önünde bulundurulması gereken ek noktalar vardır:

CICS
Bir CICS programında, tanıtıcıyı yalnızca MQOPEN çağrısını yaptığınız aynı CICS görevi içinde kullanabilirsiniz.

IMS ve z/OS toplu iş
IMS ve toplu iş ortamlarında tutamacı aynı görev içinde kullanabilirsiniz, ancak alt görevler içinde kullanamazsınız.

MQOPEN çağrısının değiştirgelerine ilişkin açıklamalar MQOPENiçinde verilmiştir.

Aşağıdaki kısımlarda, MQOPEN ' e giriş olarak sağlamanız gereken bilgiler açıklanmaktadır.

Nesnelerin tanımlanması (MQOD yapısı)

Açmak istediğiniz nesneyi tanımlamak için MQOD yapısını kullanın. Bu yapı, MQOPEN çağrısına ilişkin bir giriş değiştirgesidir. (Dinamik bir kuyruk yaratmak için MQOPEN çağrısının kullanılmasıyla yapı kuyruk yöneticisi tarafından değiştirilir.)

MQOD yapısının tüm ayrıntıları için bkz. MQOD.

Dağıtım listeleri için MQOD yapısını kullanma hakkında bilgi için bkz. "Dağıtım listeleri" sayfa 732altındaki "MQOD yapısının kullanılması" sayfa 733 .

Ad çözümlemesi
MQOPEN çağrılarının kuyruk ve kuyruk yöneticisi adlarını çözme şekli.

Not: Kuyruk yöneticisi diğer adı, RNAME alanı olmayan bir uzak kuyruk tanımlamasıdır.

Bir IBM MQ kuyruğunu açtığınızda, MQOPEN çağrısı belirttiğiniz kuyruk adında bir ad çözme işlevi gerçekleştirir. Bu, kuyruk yöneticisinin sonraki işlemleri hangi kuyrukta gerçekleştireceğini belirler. Başka bir deyişle, nesne tanımlayıcınızda (MQOD) bir diğer ad kuyruğunun ya da uzak bir kuyruğun adını belirlediğinizde, çağrı adı yerel bir kuyruğa ya da bir iletim kuyruğuna çözülür. Bir kuyruk herhangi bir giriş, göz atma ya da küme tipi için açılırsa, varsa yerel bir kuyruğa çözülür ve yoksa başarısız olur. Yalnızca çıkış, yalnızca sorma ya da çıkış ve sorma için açılırsa, yerel olmayan bir kuyruğa çözülür. Ad çözme işlemine genel bakış için bkz. Çizelge 114 sayfa 715 . *ObjectQMgrName* içinde sağladığınız ad, *ObjectName*içindeki addan *önce* çözümlenir.

Çizelge 114 sayfa 715 içinde ayrıca, bir kuyruk yöneticisinin adı için bir diğer ad tanımlamak üzere uzak bir kuyruğun yerel tanımlamasını nasıl kullanabileceğiniz de gösterilir. Bu, uzak kuyruğa ileti yerleştirirken hangi iletim kuyruğunun kullanılacağını seçmenizi sağlar; böylece, örneğin, birçok uzak kuyruk yöneticilerine gönderilen iletiler için tek bir iletim kuyruğu kullanabilirsiniz.

Aşağıdaki çizelgeyi kullanmak için, önce **Input to MQOD**(MQOD ' ye giriş) başlığı altındaki iki sol kolonu okuyun ve uygun vakayı seçin. Daha sonra, yönergeleri izleyerek ilgili satırı okuyun. **Çözümlenmiş adlar** kolonlarındaki yönergeleri izleyerek, **MQOD ' ye giriş** kolonlarına dönebilir ve değerleri yönlendirildiği gibi ekleyebilirsiniz ya da verilen sonuçlarla çizelgeden çıkabilirsiniz. Örneğin, *ObjectName*girmeniz gerekebilir.

Çizelge 114. MQOPEN kullanılırken kuyruk adlarının çözülmesi

MQOD ' ye Giriş	**MQOD ' ye Giriş**	**Çözümlenen adlar**	**Çözümlenen adlar**	**Çözümlenen adlar**
ObjectQMgrName	*ObjectName*	*ObjectQMgrName*	*ObjectName*	Transmission queue
Boş ya da yerel kuyruk yöneticisi	CLUSTER özniteliği olmayan yerel kuyruk	Yerel kuyruk yöneticisi	Giriş *ObjectName*	Uygulanamaz (yerel kuyruk kullanıldı)
Boş kuyruk yöneticisi	CLUSTER öznitelikli yerel kuyruk	İş yükü yönetimi seçilen küme kuyruğu yöneticisi ya da PUT üzerinde seçilen belirli küme kuyruğu yöneticisi	Giriş *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE ve kullanılan yerel kuyruk SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)
Yerel kuyruk yöneticisi	CLUSTER öznitelikli yerel kuyruk	Yerel kuyruk yöneticisi	Giriş *ObjectName*	Uygulanamaz (yerel kuyruk kullanıldı)
Boş ya da yerel kuyruk yöneticisi	Model kuyruğu	Yerel kuyruk yöneticisi	Oluşturulan ad	Uygulanamaz (yerel kuyruk kullanıldı)
Boş ya da yerel kuyruk yöneticisi	CLUSTER öznitelikli ya da CLUSTER özniteliği olmayan diğer ad kuyruğu	*ObjectQMgrAd* değiştirilmeden ve *ObjectName* diğer ad kuyruğu tanımlama nesnesinde *BaseQName* olarak ayarlanarak ad çözümlemesini yeniden gerçekleştirin. *ObjectQMgrAdı* belirtildiğinde yerel olarak tanımlanan bir diğer ada çözülmemelidir, ancak *ObjectQMgrAdı* ' ın boş olduğu kümelenmiş bir diğer ada (diğer kuyruk yöneticilerinin bulundurduğu) çözülebilir.		
Yerel kuyruk yöneticisi	CLUSTER öznitelikli diğer ad kuyruğu	Diğer ad, yerel olarak tanımlanmamış bir küme kuyruğuna ya da diğer adla aynı *ObjectName* olan bir küme kuyruğuna çözümlenmemelidir.		

Çizelge 114. MQOPEN kullanılırken kuyruk adlarının çözülmesi (devamı var)

MQOD ' ye Giriş	**MQOD ' ye Giriş**	**Çözümlenen adlar**	**Çözümlenen adlar**	**Çözümlenen adlar**
Boş kuyruk yöneticisi	CLUSTER öznitelikli diğer ad kuyruğu	Diğer ad, diğer adla aynı *ObjectName* olan bir küme kuyruğuna çözülebilir.		
Boş ya da yerel kuyruk yöneticisi	Uzak kuyruğun yerel tanımlaması	*ObjectQMgrName* ayarı *RemoteQMgrName*ve *ObjectName* ayarı *RemoteQName*ile yeniden ad çözümlemesi gerçekleştirin. Uzak kuyruklar çözülmemelidir		Boş değilse *XmitQName* özniteliğinin adı; tersi durumda, uzak kuyruk tanımlaması nesnesinde *RemoteQMgrName* . SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)
Boş kuyruk yöneticisi	Eşleşen yerel nesne yok; küme kuyruğu bulundu	İş yükü yönetimi seçilen küme kuyruğu yöneticisi ya da PUT üzerinde seçilen belirli küme kuyruğu yöneticisi	Giriş *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)
Boş ya da yerel kuyruk yöneticisi	Eşleşen yerel nesne yok; küme kuyruğu bulunamadı		Hata, kuyruk bulunamadı	Geçerli değildir
Yerel kuyruk yöneticisiyle aynı kuyruk paylaşım grubundaki kuyruk yöneticisinin adı	Yerel paylaşılan kuyruk	Yerel kuyruk yöneticisi	Giriş *ObjectName*	Geçerli değildir
Yerel iletim kuyruğunun adı	(Çözümlenmedi)	Giriş *ObjectQMgrAd*	Giriş *ObjectName*	Giriş *ObjectQMgrAd* SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)
Kuyruk yöneticisi diğer adı tanımlaması (*RemoteQMgrAd* yerel kuyruk yöneticisi olabilir)	(Çözülmedi, uzak kuyruk)	*RemoteQMgrAd*olarak ayarlanmış *ObjectQMgrAdı* ile ad çözümlemesini yeniden gerçekleştirin. Uzak kuyruklara çözülmemelidir	Giriş *ObjectName*	Boş değilse *XmitQName* özniteliğinin adı; tersi durumda, uzak kuyruk tanımlaması nesnesinde *RemoteQMgrName* . SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)
Kuyruk yöneticisi herhangi bir yerel nesnenin adı değil; küme kuyruğu yöneticileri ya da kuyruk yöneticisi diğer adı bulundu	(Çözümlenmedi)	*ObjectQMgrAd* ya da PUT üzerinde seçilen belirli küme kuyruğu yöneticisi	Giriş *ObjectName*	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)

Çizelge 114. MQOPEN kullanılırken kuyruk adlarının çözülmesi (devamı var)				
MQOD ' ye Giriş	**MQOD ' ye Giriş**	**Çözümlenen adlar**	**Çözümlenen adlar**	**Çözümlenen adlar**
Kuyruk yöneticisi herhangi bir yerel nesnenin adı değil; küme nesnesi bulunamadı	(Çözümlenmedi)	Giriş *ObjectQMgrAd*	Giriş *ObjectName*	Kuyruk yöneticisinin *DefXmitQName* özniteliği; burada *DefXmitQName* desteklenir. SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)

Notlar:

1. *BaseQName* , diğer ad kuyruğu tanımındaki temel kuyruğun adıdır.
2. *RemoteQName* , uzak kuyruğun yerel tanımındaki uzak kuyruğun adıdır.
3. *RemoteQMgrName* , uzak kuyruğun yerel tanımından uzak kuyruk yöneticisinin adıdır.
4. *XmitQName* , uzak kuyruğun yerel tanımındaki iletim kuyruğunun adıdır.
5. Bir kuyruk paylaşım grubunun (QSG) parçası olan IBM MQ for z/OS kuyruk yöneticilerini kullanırken, Çizelge 114 sayfa 715içindeki yerel kuyruk yöneticisi adı yerine kuyruk paylaşım grubunun adı kullanılabilir.

   Yerel kuyruk yöneticisi hedef kuyruğu açamazsa ya da kuyruğa bir ileti gönderemezse, ileti belirtilen ObjectQMgrAdına (grup içi kuyruğa alma ya da bir IBM MQ kanalı aracılığıyla) aktarılır.
6. CLUSTER, çizelgenin *ObjectName* kolonunda, kuyruğun hem CLUSTER, hem de CLUSNL özniteliklerine gönderme yapar.
7. SYSTEM.QSG.TRANSMIT.QUEUE , yerel ve uzak kuyruk yöneticileri aynı kuyruk paylaşım grubundaysa kullanılır; grup içi kuyruğa alma etkinleştirilir.
8. Her küme gönderen kanalına farklı bir küme iletim kuyruğu atadıysanız, SYSTEM.CLUSTER.TRANSMIT.QUEUE , küme iletim kuyruğunun adı olmayabilir. Birden çok küme iletim kuyruğuna ilişkin ek bilgi için Clustering: Planning nasıl küme iletim kuyrukları yapılandırılacağınıbaşlıklı konuya bakın.
9. Kuyruk yöneticisinin herhangi bir yerel nesnenin adı olmadığı bir durumda; küme kuyruğu yöneticileri ya da kuyruk yöneticisi diğer adı bulundu.

   ObjectQMgrNamekomutunu kullanarak bir kuyruk yöneticisi adı sağladığınızda ve yerel kuyruk yöneticisi tarafından bu hedefe ulaşacak farklı küme adları bilinen birden çok küme kanalı varsa, hedef kuyruğun küme adından bağımsız olarak, bu kanallardan herhangi biri iletiyi taşımak için kullanılabilir.

   Bu kuyrukla ilgili iletilerin yalnızca, kuyrukla aynı küme adına sahip bir kanal üzerinden gönderilmesini bekliyorsanız, bu beklenmeyen bir durum olabilir.

   Ancak **ObjectQMgrName** bu durumda önceliklidir ve küme iş yükü dengelemesi, içinde bulundukları küme adından bağımsız olarak, o kuyruk yöneticisine ulaşabilecek tüm kanalları dikkate alır.

Bir diğer ad kuyruğunun açılması, diğer adın çözüldüğü temel kuyruğu da açar ve uzak bir kuyruk açıldığında iletim kuyruğu da açılır. Bu nedenle, belirttiğiniz kuyruğu ya da diğeri açıkken çözüldüğü kuyruğu silemezsiniz.

Bir diğer ad kuyruğu yerel olarak tanımlanmış başka bir diğer ad kuyruğuna (bir kümede paylaşılan ya da paylaşılmayan) çözülemese de, uzaktan tanımlanan bir küme diğer adı kuyruğuna çözülmesine izin verilir ve bu nedenle temel kuyruk olarak belirtilebilir.

Çözülen kuyruk adı ve çözülen kuyruk yöneticisi adı, MQOD ' daki *ResolvedQName* ve *ResolvedQMgrName* alanlarında saklanır.

Dağıtılmış bir kuyruğa alma ortamında ad çözümlemesi hakkında daha fazla bilgi için bkz. Kuyruk adı çözümlemesi nedir?.

MQOPEN çağrısının seçeneklerinin kullanılması

MQOPEN çağrısının **Options** değiştirgesinde, açmakta olduğunuz nesne için size verilen erişimi denetlemek üzere bir ya da daha çok seçenek belirlemelisiniz. Bu seçeneklerle şunları yapabilirsiniz:

- Bir kuyruğu açın ve o kuyruğa konan tüm iletilerin, kuyruğun aynı eşgörünümüne yönlendirilmeleri gerektiğini belirtin
- Üzerine ileti koymanıza izin vermek için bir kuyruk açın
- Bir kuyruktaki iletilere göz atmanızı sağlamak için kuyruğu açın
- Bir kuyruktan ileti kaldırmanızı sağlamak için kuyruğun açılması
- Bir nesneyi sorgulamanızı ve özniteliklerini ayarlamanızı sağlamak için nesneyi açın (ancak yalnızca kuyrukların özniteliklerini ayarlayabilirsiniz)
- İletileri yayınlamak için bir konuyu ya da konu dizesini açma
- Bağlam bilgilerini bir iletiyle ilişkilendir
- Güvenlik denetimleri için kullanılacak alternatif bir kullanıcı kimliğini aday gösterin
- Kuyruk yöneticisi susturma durumundaysa çağrıyı denetle

Küme kuyruğu için MQOPEN seçeneği
Kuyruk tanıtıcısı için kullanılan bağ tanımı, MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDya da MQBND_BIND_ON_GROUPdeğerini alabilen **DefBind** kuyruk özniteliğinden alınır.

MQPUT komutunu kullanarak kuyruğa konan tüm iletileri aynı rotayla aynı kuyruk yöneticisine yöneltmek için MQOPEN çağrısında MQOO_BIND_ON_OPEN seçeneğini kullanın.

Bir hedefin MQPUT zamanında seçileceğini belirtmek için, yani ileti temelinde MQOPEN çağrısında MQOO_BIND_NOT_FIXED seçeneğini kullanın.

MQPUT kullanılarak kuyruğa konan ileti gruplarındaki tüm iletilerin aynı hedef örneğe ayrıldığını belirtmek için, MQOPEN çağrısında MQOO_BIND_ON_GROUP seçeneğini kullanın.

Gruptaki tüm iletilerin aynı hedefte işlendiğinden emin olmak için kümelerle ileti grupları kullanılırken MQOO_BIND_ON_OPEN ya da MQOO_BIND_ON_GROUP belirtilmelidir.

Bu seçeneklerden herhangi birini belirtmezseniz, MQOO_BIND_AS_Q_DEFvarsayılan değeri kullanılır.

MQODiçinde bir kuyruk yöneticisinin adını belirtirseniz, o kuyruk yöneticisindeki kuyruk seçilir. Kuyruk yöneticisi adı boşsa, herhangi bir eşgörünüm seçilebilir. Ek bilgi için bkz. “MQOPEN ve kümeler” sayfa 848 .

Bir küme kuyruğunu QALIAS tanımı kullanarak açarsanız, bazı kuyruk öznitelikleri temel kuyruk değil, diğer ad kuyruğu tarafından tanımlanır. Küme öznitelikleri, diğer ad kuyruğu tarafından geçersiz kılınan temel kuyruk tanımlamasının öznitelikleri arasındadır. Örneğin, aşağıdaki parçacıkta, küme kuyruğu MQOO_BIND_ON_OPENile değil, MQOO_BIND_NOT FIXED ile açılır. Küme kuyruğu tanımlaması kümede duyurulmuştur; diğer ad kuyruğu tanımı kuyruk yöneticisi için yereldir.

```
DEFINE QLOCAL(CLQ1) CLUSTER(MYCLUSTER) DEFBIND(OPEN) REPLACE
DEFINE QALIAS(ACLQ1) TARGET(CLQ1) DEFBIND(NOTFIXED) REPLACE
```

İletileri koymak için MQOPEN seçeneği
İleti koymak üzere bir kuyruğu ya da konuyu açmak için MQOO_OUTPUT seçeneğini kullanın.

İletilere göz atmak için MQOPEN seçeneği
Üzerindeki iletilere *göz atabilmek* üzere bir kuyruk açmak için, MQOPEN çağrısıyla MQOO_BROWSE seçeneğini kullanın.

Bu, kuyruk yöneticisinin kuyruktaki sonraki iletiyi tanımlamak için kullandığı bir *göz atma imleci* yaratır. Daha fazla bilgi için “Kuyruktaki iletilere göz atma” sayfa 770 başlıklı konuya bakın.

Not:

1. Uzak kuyruktaki iletilere göz atamaz; MQOO_BROWSE seçeneğini kullanarak uzak bir kuyruğu açmayın.

2. Bir dağıtım listesi açarken bu seçeneği belirleyemezsiniz. Dağıtım listeleriyle ilgili daha fazla bilgi için bkz. “Dağıtım listeleri” sayfa 732.
3. İşbirliği taramasını kullanıyorsanız MQOO_CO_OP ' yi MQOO_BROWSE ile birlikte kullanın; bkz. Seçenekler

İletileri kaldırmak için MQOPEN seçenekleri
Üç seçenek, bir kuyruktan ileti kaldırmak için kuyruğun açılmasını denetler.

MQOPEN çağrısında bunlardan yalnızca birini kullanabilirsiniz. Bu seçenekler, programınızın kuyruğa özel mi, yoksa paylaşılan erişime mi sahip olduğunu tanımlar. *Dışlayıcı erişim* , kuyruğu kapatıncaya kadar, iletileri yalnızca siz kaldırabilirsiniz. Başka bir program iletileri kaldırmak için kuyruğu açma girişiminde bulunursa, MQOPEN çağrısı başarısız olur. *Paylaşılan erişim* , birden çok programın kuyruktan gelen iletiler.

En uygun yaklaşım, kuyruk tanımlandığında kuyruk için amaçlanan erişim tipini kabul etmektir. Kuyruk tanımlaması, **Shareability** ve **DefInputOpenOption** öznitelikleri. Bu erişimi kabul etmek için MQOO_INPUT_AS_Q_DEF seçeneğini kullanın. Bu özniteliklerin ayarının, bu seçeneği kullandığınızda size verilecek erişim tipini nasıl etkilediğini görmek için bkz. Çizelge 115 sayfa 719 .

Çizelge 115. MQOPEN çağrısının kuyruk özniteliklerinin ve seçeneklerinin kuyruklara erişimi nasıl etkilediği

Kuyruk öznitelikleri		**MQOPEN seçenekleriyle erişim tipi**		
Shareability	**DefInputOpenOption**	**AS_Q_DEF**	**PAYLAŞILAN**	**dışlayıcı**
Paylaşılabilir	PAYLAŞILAN	paylaşılan	paylaşılan	dışlayıcı
Paylaşılabilir	dışlayıcı	dışlayıcı	paylaşılan	dışlayıcı
Paylaşılmadı*	SHARED*	dışlayıcı	dışlayıcı	dışlayıcı
PAYLAŞILAMAZ	dışlayıcı	dışlayıcı	dışlayıcı	dışlayıcı

Not: * Bu öznitelik birleşimine sahip olacak bir kuyruk tanımlayabilseniz de, varsayılan giriş açma seçeneği paylaşılabilirlik özniteliği tarafından geçersiz kılınır.

Alternatif olarak:

- Diğer programlar aynı anda kuyruktan ileti kaldırabilse bile uygulamanızın başarılı bir şekilde çalışabileceğini biliyorsanız, MQOO_INPUT_SHARED seçeneğini kullanın. Çizelge 115 sayfa 719 içinde, bazı durumlarda, bu seçenekle bile, size kuyruk için dışlayıcı erişimin nasıl verileceği gösterilir.
- Uygulamanızın başarılı bir şekilde çalışabileceğini ancak diğer programların aynı anda kuyruktan ileti kaldırmasını engellemesi durumunda biliyorsanız, MQOO_INPUT_EXCLUSIVE seçeneğini kullanın.

Not:

1. Uzak kuyruktan ileti kaldıramazsınız. Bu nedenle, MQOO_INPUT_ * seçeneklerinden herhangi birini kullanarak uzak bir kuyruk açamazsınız.
2. Bir dağıtım listesi açarken bu seçeneği belirleyemezsiniz. Daha fazla bilgi için bkz. “Dağıtım listeleri” sayfa 732.

Öznitelikleri ayarlamak ve sorgulamak için MQOPEN seçenekleri
Özniteliklerini ayarlayabileceğiniz bir kuyruğu açmak için MQOO_SET seçeneğini kullanın.

Başka bir nesne tipinin özniteliklerini ayarlayamazsınız (bkz. “Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815).

Özniteliklerini sorgulamak üzere bir nesneyi açmak için MQOO_INQUIRE seçeneğini kullanın.

Not: Bir dağıtım listesi açarken bu seçeneği belirleyemezsiniz.

İleti bağlamıyla ilgili MQOPEN seçenekleri
Bağlam bilgilerini bir kuyruğa yerleştirirken bir iletiyle ilişkilendirmek istiyorsanız, kuyruğu açarken ileti bağlamı seçeneklerinden birini kullanmanız gerekir.

Seçenekler, iletiyi oluşturan *kullanıcı* ile iletiyi oluşturan *uygulamayla* ilgili bağlam bilgileri arasında fark yaratmanızı sağlar. Ayrıca, iletiyi kuyruğa koyduğunuzda bağlam bilgilerini ayarlayabilir ya da bağlamın başka bir kuyruk tanıtıcısından otomatik olarak alınmasını seçebilirsiniz.

İlgili kavramlar
"İleti bağlamı" sayfa 45
İleti bağlamı bilgileri, iletiyi alan uygulamanın iletiyi başlatan hakkında bilgi edinmesini sağlar.

"İleti bağlamı bilgilerinin denetlenmesi" sayfa 728
Kuyruğa ileti koymak için MQPUT ya da MQPUT1 çağrısını kullandığınızda, kuyruk yöneticisinin ileti tanımlayıcısına varsayılan bağlam bilgileri ekleyeceğini belirtebilirsiniz. Uygun yetki düzeyine sahip uygulamalar ek bağlam bilgileri ekleyebilir. Bağlam bilgilerini denetlemek için MQPMO yapısındaki seçenekler alanını kullanabilirsiniz.

Diğer kullanıcı yetkisi için MQOPEN seçeneği
MQOPEN çağrısıyla bir nesneyi açma girişiminde bulunduğunuzda, kuyruk yöneticisi o nesneyi açma yetkiniz olup olmadığını denetler. Yetkiniz yoksa, arama başarısız olur.

Ancak, sunucu programları kuyruk yöneticisinin sunucunun kendi yetkisi yerine, üzerinde çalıştığı kullanıcının yetkisini denetlemesini isteyebilir. Bunu yapmak için, MQOPEN çağrısının MQOO_ALTERNATE_USER_AUTHORITY seçeneğini kullanmalı ve MQOD yapısının *AlternateUserId* alanında diğer kullanıcı kimliğini belirtmelidir. Genellikle sunucu, işlediği iletideki bağlam bilgilerinden kullanıcı kimliğini alır.

z/OS *Kuyruk yöneticisi susturması için MQOPEN seçeneği*
Kuyruk yöneticisi susturma durumundayken MQOPEN çağrıyı kullanırsanız, kullandığınız ortama bağlı olarak çağrı başarısız olabilir.

z/OSüzerindeki CICS ortamında, kuyruk yöneticisi susturma durumundayken MQOPEN çağrıyı kullanırsanız, çağrı her zaman başarısız olur.

Diğer z/OS ve Multiplatforms ortamlarında, kuyruk yöneticisi susturulurken, MQOPEN çağrısının MQOO_FAIL_IF_QUIESCING seçeneğini kullandığınızda çağrı başarısız olur.

Yerel kuyruk adlarını çözmek için MQOPEN seçeneği
Yerel, diğer ad ya da model kuyruğu açtığınızda, yerel kuyruk döndürülür.

Ancak, bir uzak kuyruğu ya da küme kuyruğunu açtığınızda, MQOD yapısının *ResolvedQName* ve *ResolvedQMgrName* alanları, uzak kuyruk tanımlamasında bulunan uzak kuyruk ve uzak kuyruk yöneticisi adlarıyla ya da seçilen uzak küme kuyruğuyla doldurulur.

MQOD yapısındaki *ResolvedQName* öğesini, açılan yerel kuyruğun adıyla doldurmak için MQOPEN çağrısının MQOO_RESOLVE_LOCAL_Q seçeneğini kullanın. *ResolvedQMgrName* , yerel kuyruğu barındıran yerel kuyruk yöneticisinin adıyla benzer şekilde doldurulur. Bu alan yalnızca MQOD yapısının Sürüm 3 ile kullanılabilir; yapı Sürüm 3 'ten küçükse, MQOO_RESOLVE_LOCAL_Q hata döndürülmeden yoksayılır.

Açılırken MQOO_RESOLVE_LOCAL_Q belirtirseniz (örneğin, uzak kuyruk), *ResolvedQName* iletilerin konacağı iletim kuyruğunun adıdır. *ResolvedQMgrName* , iletim kuyruğunu barındıran yerel kuyruk yöneticisinin adıdır.

Dinamik kuyruklar yaratılması
Uygulamanız sona erdikten sonra kuyruğa gerek duymadığınızda dinamik bir kuyruk kullanın.

Örneğin, yanıt kuyruğunuz için dinamik bir kuyruk kullanabilirsiniz. Kuyruğa bir ileti yerleştirdiğinizde, MQMD yapısının *ReplyToQ* alanında yanıt kuyruğunun adını belirtirsiniz (bkz. "MQMD yapısını kullanarak ileti tanımlanması" sayfa 724).

Dinamik kuyruk yaratmak için, MQOPEN çağrısıyla birlikte model kuyruğu olarak bilinen bir şablon kullanırsınız. IBM MQ komutlarını ya da işlem ve denetim panolarını kullanarak bir model kuyruğu yaratırsınız. Yarattığınız dinamik kuyruk, model kuyruğunun özniteliklerini alır.

MQOPEN ' i çağırdığınızda, MQOD yapısının *ObjectName* alanında model kuyruğunun adını belirtin. Çağrı tamamlandığında, *ObjectName* alanı, oluşturulan dinamik kuyruğun adına ayarlanır. Ayrıca, *ObjectQMgrName* alanı yerel kuyruk yöneticisinin adına ayarlanır.

Yarattığınız dinamik kuyruğun adını üç şekilde belirtebilirsiniz:

- MQOD yapısının *DynamicQName* alanında olmasını istediğiniz tam adı belirtin.
- Ad için bir önek (33 karakterden az) belirtin ve kuyruk yöneticisinin adın geri kalanını oluşturmasına izin verin. Bu, kuyruk yöneticisinin benzersiz bir ad ürettiği, ancak yine de bir denetiminiz olduğu anlamına gelir (örneğin, her kullanıcının belirli bir öneki kullanmasını isteyebilir ya da adlarında belirli bir öneki olan kuyruklara özel bir güvenlik sınıflandırması vermek isteyebilirsiniz). Bu yöntemi kullanmak için, *DynamicQName* alanının son boşluk olmayan karakteri olarak bir yıldız işareti (*) belirleyin. Dinamik kuyruk adı için tek bir yıldız işareti (*) belirlemeyin.
- Kuyruk yöneticisinin tam adı oluşturmasına izin verin. Bu yöntemi kullanmak için, *DynamicQName* alanının ilk karakter konumunda bir yıldız işareti (*) belirtin.

Bu yöntemlerle ilgili daha fazla bilgi için DynamicQName alanının açıklamasına bakın.

Dinamik ve Model kuyruklarındadinamik kuyruklar hakkında daha fazla bilgi vardır.

Uzak kuyrukların açılması

Uzak kuyruk, uygulamanın bağlı olduğu kuyruk yöneticisinden başka bir kuyruk yöneticisinin sahip olduğu bir kuyruktur.

Uzak bir kuyruğu açmak için, yerel bir kuyruk için MQOPEN çağrıyı kullanın. Kuyruğun adını aşağıdaki gibi belirtebilirsiniz:

1. MQOD yapısının *ObjectName* alanında, *yerel* kuyruk yöneticisinin tanıdığı şekilde uzak kuyruğun adını belirtin.

   Not: Bu durumda *ObjectQMgrName* alanını boş bırakın.
2. MQOD yapısının *ObjectName* alanında, *uzak* kuyruk yöneticisi tarafından bilinen uzak kuyruğun adını belirtin. *ObjectQMgrName* alanında aşağıdakilerden birini belirtin:

   - Uzak kuyruk yöneticisiyle aynı adı taşıyan iletim kuyruğunun adı. Ad ve büyük harf (büyük harf, küçük harf ya da karışım) *tam olarak*eşleşmelidir.
   - Hedef kuyruk yöneticisine ya da iletim kuyruğuna çözülen kuyruk yöneticisi diğer adı.

   Bu, kuyruk yöneticisine iletinin hedefinin yanı sıra iletim kuyruğuna ulaşmak için yerleştirilmesi gerektiğini bildirir.
3. *DefXmitQname* destekleniyorsa, MQOD yapısının *ObjectName* alanında, *uzak* kuyruk yöneticisi tarafından bilinen uzak kuyruğun adını belirtin.

   Not: *ObjectQMgrName* alanını uzak kuyruk yöneticisinin adına ayarlayın (bu durumda boş bırakılamaz).

MQOPEN çağrıldığında yalnızca yerel adlar doğrulanır; son denetim, kullanılacak iletim kuyruğunun varlığıdır.

Bu yöntemlerÇizelge 114 sayfa 715içinde özetlenmiştir.

MQCLOSE çağrısı kullanılarak nesnelerin kapatılması

Bir nesneyi kapatmak için MQCLOSE çağrıyı kullanın.

Nesne bir kuyruksa, aşağıdakilere dikkat edin:

- Kapatmadan önce geçici bir dinamik kuyruğu boşaltmanız gerekmez.

Geçici bir dinamik kuyruğu kapattığınızda, kuyruk, üzerinde bulunmaya devam eden iletilerle birlikte silinir. Bu, kuyruk için bekleyen kesinleştirilmemiş MQGET, MQPUT ya da MQPUT1 çağrıları olsa da doğrudur.

- IBM MQ for z/OSüzerinde, o kuyruk için bekleyen bir MQGMO_SET_SIGNAL seçeneğine sahip MQGET istekleriniz varsa, bunlar iptal edilir.
- MQOO_BROWSE seçeneğini kullanarak kuyruğu açtıysanız, göz atma imleciniz yok edilir.

Kapanış, eşitleme noktasıyla ilişkili değildir, bu nedenle eşitleme noktasından önce ya da sonra kuyrukları kapatabilirsiniz.

MQCLOSE çağrısına giriş olarak şunları sağlamanız gerekir:

- Bağlantı tanıtıcısı. Bunu açmak için kullanılan bağlantı tanıtıcısını kullanın ya da alternatif olarak z/OSüzerindeki CICS uygulamaları için sabit MQHC_DEF_HCONN (sıfır değerine sahip) belirtebilirsiniz.
- Kapatmak istediğiniz nesnenin tanıtıcısı. Bunu MQOPEN çağrısının çıkışından alın.
- *Options* alanında MQCO_NONE (kalıcı bir dinamik kuyruğu kapatmıyorsan).
- (Kalıcı bir dinamik kuyruk kapatılırken), kuyruk yöneticisinin kuyrukta hala ileti olsa da kuyruğu silip silmeyeceğini belirleyen denetim seçeneği.

MQCLOSE çıkışı:

- Tamamlanma kodu
- Neden kodu
- Nesne tanıtıcısı, MQHO_UNUSABLE_HOBJ değerine geri çevir

MQCLOSE çağrısının değiştirgelerine ilişkin açıklamalar MQCLOSEiçinde verilmiştir.

Kuyruktaki iletilerin konması

İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

Kuyruğa ileti koymak için MQPUT çağrıyı kullanın. MQPUT, ilk MQOPEN çağrısından sonra aynı kuyruğa birçok ileti koymak için arka arkaya MQPUT işlevini kullanabilirsiniz. Tüm iletilerinizi kuyruğa koymayı bitirdiğinizde MQCLOSE ' yi çağırın.

Kuyruğa tek bir ileti koymak ve kuyruğu hemen kapatmak istiyorsanız, MQPUT1 çağrısını kullanabilirsiniz. MQPUT1 , aşağıdaki çağrı dizisiyle aynı işlevleri gerçekleştirir:

- MQOPEN
- MQPUT
- MQCLOSE

Ancak genellikle, kuyruğa konacak birden çok iletiniz varsa, MQPUT çağrıyı kullanmak daha verimlidir. Bu, üzerinde çalıştığınız iletinin ve altyapının boyutuna bağlıdır.

Bir kuyruğa ileti koymaya ilişkin daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “İletilerin MQPUT çağrıyla yerel kuyruğa konması” sayfa 723
- “Uzak kuyruğa ileti konması” sayfa 728
- “İletinin özelliklerini ayarlama” sayfa 728
- “İleti bağlamı bilgilerinin denetlenmesi” sayfa 728
- “MQPUT1 çağrısı kullanılarak kuyruğa bir ileti konması” sayfa 730
- “Dağıtım listeleri” sayfa 732
- “Bazı durumlarda, yanıtların başarısız olduğu durumlar” sayfa 736

İlgili kavramlar

“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

“Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

“Kuyruktan ileti alma” sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

“Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

“Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

“MQI ve kümelerle çalışma” sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

“IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

“IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları” sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

İletilerin MQPUT çağrıyla yerel kuyruğa konması

MQPUT çağrıyı kullanarak iletileri yerel bir kuyruğa koymaya ilişkin bilgi edinmek için bu bilgileri kullanın.

MQPUT çağrısına giriş olarak şunları sağlamanız gerekir:

- Bağlantı tanıtıcısı (Hconn).
- Kuyruk tanıtıcısı (Hobj).
- Kuyruğa koymak istediğiniz iletinin tanımlaması. Bu, ileti tanımlayıcı yapısı (MQMD) biçimindedir.
- Bir koyma iletisi seçenekleri yapısı (MQPMO) biçiminde denetim bilgileri.
- İletide bulunan verilerin uzunluğu (MQLONG).
- İleti verilerinin kendisi.

MQPUT çağrısının çıkışı aşağıdaki gibidir:

- Neden kodu (MQLONG)
- Tamamlanma kodu (MQLONG)

Çağrı başarıyla tamamlanırsa, seçenek yapınızı ve ileti tanımlayıcı yapınızı da döndürür. Çağrı, kuyruğun adını ve iletinin gönderildiği kuyruk yöneticisini göstermek için seçenek yapınızı değiştirir. Kuyruk yöneticisinin, koymakta olduğunuz iletinin tanıtıcısı için benzersiz bir değer oluşturmasını isterseniz (MQMD yapısının *MsgId* alanında ikili sıfır belirterek), çağrı bu yapıyı size döndürmeden önce *MsgId* alanına değeri ekler. Başka bir MQPUT yayınlamadan önce bu değeri ilk durumuna getirin.

MQPUTiçinde MQPUT çağrısının bir açıklaması vardır.

MQPUT çağrısına giriş olarak gereken bilgilere ilişkin ek bilgi için aşağıdaki bağlantılara bakın:

- “Çekme noktalarının belirtilmesi” sayfa 724
- “MQMD yapısını kullanarak ileti tanımlanması” sayfa 724
- “MQPMO yapısını kullanarak seçeneklerin belirtilmesi” sayfa 724
- “İletinizdeki veriler” sayfa 727
- “İleti koyma: İleti tanıtıcılarını kullanma” sayfa 728

Çekme noktalarının belirtilmesi

z/OS uygulamalarında CICS içindeki bağlantı tanıtıcısı (*Hconn*) için, değişmez MQHC_DEF_HCONN değerini (sıfır değerine sahip) belirtebilir ya da MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını kullanabilirsiniz. Diğer uygulamalar için her zaman MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını kullanın.

Çalıştığınız ortam her neyse, MQOPEN çağrısıyla döndürülen aynı kuyruk tanıtıcısını (*Hobj*) kullanın.

MQMD yapısını kullanarak ileti tanımlanması

İleti tanımlayıcı yapısı (MQMD), MQPUT ve MQPUT1 çağrılarına ilişkin bir giriş/çıkış parametresidir. Kuyruğa koymakta olduğunuz iletiyi tanımlamak için bu iletiyi kullanın.

İleti için MQPRI_PRIORITY_AS_Q_DEF ya da MQPER_PERSISTENCE_AS_Q_DEF belirtildiyse ve kuyruk bir küme kuyruğuysa, kullanılan değerler, MQPUT ' nin çözüldüğü kuyruğun değerleridir. MQPUT için bu kuyruk geçersiz kılındıysa, çağrı başarısız olur. Ek bilgi için Kuyruk yöneticisi kümesinin yapılandırılması başlıklı konuya bakın.

Not: *MsgId* ve *CorrelId* öğesinin benzersiz olduğundan emin olmak için yeni bir ileti koymadan önce MQPMO_NEW_MSG_ID ve MQPMO_NEW_CORREL_ID öğelerini kullanın. Bu alanlardaki değerler, başarılı bir MQPUT için döndürülür.

MQMD 'nin “IBM MQ ileti” sayfa 17içinde tanımladığı ileti özelliklerine bir giriş vardır ve MQMD' de yapının kendisinin bir açıklaması vardır.

MQPMO yapısını kullanarak seçeneklerin belirtilmesi

MQPUT ve MQPUT1 çağrılarına seçenek geçirmek için MQPMO (Put Message Option) yapısını kullanın.

Aşağıdaki bölümler, bu yapının alanlarını doldurmanıza yardımcı olur. MQPMOiçinde yapının bir açıklaması vardır.

Yapı aşağıdaki alanları içerir:

- *StrucId*
- *Version*
- *Options*
- *Context*
- *ResolvedQName*
- *ResolvedQMgrName*
- *RecsPresent*
- *PutMsgRecsFields*
- *ResponseRecOffset and ResponseRecPtr*
- *OriginalMsgHandle*
- *NewMsgHandle*
- *Action*
- *PubLevel*

Bu alanların içeriği aşağıdaki gibidir:

StrucId
Bu, yapıyı bir koyma iletisi seçenekleri yapısı olarak tanımlar. Bu, 4 karakterlik bir alandır. MQPMO_STRUC_ID değerini her zaman belirtin.

Sürüm
Bu, yapının sürüm numarasını açıklar. Varsayılan değer MQPMO_VERSION_1' dir. MQPMO_VERSION_2girerseniz, dağıtım listelerini kullanabilirsiniz (bkz. “Dağıtım listeleri” sayfa 732). MQPMO_VERSION_3girerseniz, ileti tanıtıcılarını ve ileti özelliklerini kullanabilirsiniz.

MQPMO_CURRENT_VERSION girerseniz, uygulamanız her zaman en son düzeyi kullanacak şekilde ayarlanır.

Seçenekler
Bu, aşağıdakileri denetler:

- Koyma işleminin bir iş birimine dahil edilip edilmediğini belirler
- Bir iletiyle ilişkili bağlam bilgisi miktarı
- Bağlam bilgilerinin alındığı yer
- Kuyruk yöneticisi susturma durumundaysa çağrıların başarısız olup olmayacağını belirler
- Gruplamaya ya da bölümlemeye izin verilip verilmediğini belirler
- Yeni ileti tanıtıcısı ve ilinti tanıtıcısı oluşturulması
- İletilerin ve bölümlerin kuyruğa konma sırası
- Yerel kuyruk adlarının çözümlenip çözümlenmeyeceğini belirler

Options alanını varsayılan değere (MQPMO_NONE) ayarlı olarak bırakırsanız, yerleştirdiğiniz iletiyle ilişkilendirilmiş varsayılan bağlam bilgileri vardır.

Ayrıca, aramanın eşitleme noktalarıyla çalışma şekli platform tarafından belirlenir. Eşitleme noktası denetimi varsayılan değeri z/OS ; diğer platformlar için hayır.

Bağlam
Bu, bağlam bilgilerinin kopyalanmasını istediğiniz kuyruk tanıtıcısının adını belirtir (*Options* alanında istenirse).

İleti bağlamına giriş için bkz. “İleti bağlamı” sayfa 45. Bir iletideki bağlam bilgilerini denetlemek için MQPMO yapısını kullanma hakkında bilgi için bkz. “İleti bağlamı bilgilerinin denetlenmesi” sayfa 728.

ResolvedQName
Bu, iletiyi almak için açılan kuyruğun adını (herhangi bir diğer ad çözüldükten sonra) içerir. Bu bir çıkış alanıdır.

ResolvedQMgrAdı
Bu, *ResolvedQName*içinde kuyruğun sahibi olan kuyruk yöneticisinin adını (herhangi bir diğer ad çözüldükten sonra) içerir. Bu bir çıkış alanıdır.

MQPMO, dağıtım listeleri için gerekli alanları da barındırabilir (bkz. “Dağıtım listeleri” sayfa 732). Bu olanağı kullanmak istiyorsanız, MQPMO yapısının Sürüm 2 kullanılır. Bu, aşağıdaki alanları içerir:

RecsPresent
Bu alan, dağıtım listesindeki kuyrukların sayısını içerir; yani, Put Message Records (MQPMR) ve ilgili Response Records (MQRR).

Girdiğiniz değer, MQOPEN ' de sağlanan Nesne Kayıtları sayısıyla aynı olabilir. Ancak, değer MQOPEN çağrısında sağlanan Nesne Kaydı sayısından azsa ya da Put İleti Kaydı belirtmezseniz, tanımlı olmayan kuyrukların değerleri, ileti tanımlayıcısı tarafından sağlanan varsayılan değerlerden alınır. Ayrıca, değer sağlanan Nesne Kayıtları sayısından büyükse, fazla Put İleti Kayıtları yoksayılır.

Aşağıdakilerden birini yapmanız önerilir:

- Her hedeften bir rapor ya da yanıt almak istiyorsanız, MQOR yapısında görünenle aynı değeri girin ve *MsgId* alanlarını içeren MQPMR ' ları kullanın. Bu *MsgId* alanlarını sıfırlamak için kullanıma hazırlayın ya da MQPMO_NEW_MSG_ID değerini belirleyin.

  İletiyi kuyruğa koyduğunuzda, kuyruk yöneticisinin yarattığı *MsgId* değerleri MQPMR ' larda kullanılabilir olur; her bir rapor ya da yanıtla hangi hedefin ilişkilendirildiğini tanımlamak için bunları kullanabilirsiniz.
- Raporları ya da yanıtları almak istemiyorsanız, aşağıdakilerden birini seçin:

  1. Hemen başarısız olan hedefleri belirlemek istiyorsanız, *RecsPresent* alanında MQOR yapısında göründüğü gibi aynı değeri girmek ve bu hedefleri tanımlamak için MQRR ' leri sağlamak isteyebilirsiniz. MQPMR belirtmeyin.

2. Başarısız olan hedefleri tanımlamak istemiyorsanız, *RecsPresent* alanına sıfır girin ve MQPMR 'ler ya da MQRR' ler sağlamayın.

Not: MQPUT1kullanıyorsanız, Yanıt Kaydı Göstergelerinin ve Yanıt Kaydı Göreli Göreli Konumlarının sayısı sıfır olmalıdır.

Put Message Records (MQPMR) ve Response Records (MQRR) ile ilgili tam açıklama için MQPMR ve MQRRbaşlıklı konuya bakın.

PutMsgRecFields
Bu, her bir Put Message Record (MQPMR) içinde hangi alanların bulunduğunu gösterir. Bu alanların listesi için bkz. "MQPMR yapısının kullanılması" sayfa 735.

PutMsgRecOffset* ve *PutMsgRecPtr
İşaretçiler (genellikle C dilinde) ve göreli konumlar (genellikle COBOL dilinde) Put Message Records ile ilgili olarak kullanılır (MQPMR yapısına genel bakış için bkz. "MQPMR yapısının kullanılması" sayfa 735).

İlk koyma iletisi kaydına ilişkin bir gösterge belirtmek için *PutMsgRecPtr* alanını ya da ilk koyma iletisi kaydına ilişkin göreli konumu belirtmek için *PutMsgRecOffset* alanını kullanın. Bu, MQPMO ' nun başlangıcındaki göreli konumdur. *PutMsgRecFields* alanına bağlı olarak, *PutMsgRecOffset* ya da *PutMsgRecPtr*için boş olmayan bir değer girin.

ResponseRecGöreli Konumu ve ResponseRecPtr
Ayrıca, Yanıt Kayıtlarını adreslemek için işaretçiler ve göreli konumlar da kullanabilirsiniz (Yanıt Kayıtlarıyla ilgili daha fazla bilgi için bkz. "MQRR yapısının kullanılması" sayfa 734).

İlk yanıt kaydına ilişkin bir gösterge belirtmek için *ResponseRecPtr* alanını ya da ilk yanıt kaydının göreli konumunu belirtmek için *ResponseRecOffset* alanını kullanın. Bu, MQPMO yapısının başlangıcındaki göreli konumdur. *ResponseRecOffset* ya da *ResponseRecPtr*için boş olmayan bir değer girin.

Not: İletileri bir dağıtım listesine koymak için MQPUT1 kullanıyorsanız, *ResponseRecPtr* boş değerli ya da sıfır olmalı ve *ResponseRecOffset* sıfır olmalıdır.

MQPMO yapısının 3. sürümü ayrıca aşağıdaki alanları da içerir:

OriginalMsgTanıtıcısı
Bu alanda yapabileceğiniz kullanım, *İşlem* alanının değerine bağlıdır. İlişkili ileti özellikleriyle yeni bir ileti ekliyorsanız, bu alanı önceden yarattığınız ileti tanıtıcısı olarak ayarlayın ve özellikleri açık olarak ayarlayın. Daha önce alınan bir iletiye yanıt olarak bir rapor iletiyor, yanıtlıyor ya da üretiyorsanız, bu alan o iletinin ileti tanıtıcısını içerir.

NewMsgTanıtıcısı
Bir *NewMsgHandle*belirtirseniz, *OriginalMsgHandle*ile ilişkili tanıtıcı geçersiz kılma özellikleriyle ilişkilendirilmiş özellikler görüntülenir. Daha fazla bilgi için bakınız: Action (MQLONG).

Eylem
Gerçekleştirilmekte olan koymanın tipini belirtmek için bu alanı kullanın. Olası değerler ve anlamları aşağıdaki gibidir:

MQACTP_YENI
Bu, başka bir iletiyle ilgili olmayan yeni bir iletidir.

MQACTP_FORWARD
Bu ileti daha önce alındı ve şimdi iletiliyor.

MQACTP_REPLY
Bu ileti, önceden alınan bir iletiye verilen yanıttır.

MQACTP_REPORT
Bu ileti, önceden alınan bir iletinin sonucu olarak oluşturulan bir rapordur.

Daha fazla bilgi için bakınız: Action (MQLONG).

PubLevel
Bu ileti bir yayınsa, bu alanı hangi aboneliklerin alacağını belirlemek için ayarlayabilirsiniz. Yalnızca bu değerden küçük ya da bu değere eşit *SubLevel* olan abonelikler bu yayını alır. Varsayılan değer, en yüksek düzey olan 9 'dur ve herhangi bir *SubLevel* aboneliklerinin bu yayını alabileceği anlamına gelir.

İletinizdeki veriler

MQPUT çağrısının **Buffer** değiştirgesinde verilerinizi içeren arabelleğin adresini verin. İletilerinizdeki verilere herhangi bir şey ekleyebilirsiniz. Ancak iletilerdeki veri miktarı, bunları işleyen uygulamanın performansını etkiler.

Verilerin büyüklük üst sınırı aşağıdaki tarafından belirlenir:

- Kuyruk yöneticisinin **MaxMsgLength** özniteliği
- İletiyi koymakta olduğunuz kuyruğun **MaxMsgLength** özniteliği
- IBM MQ tarafından eklenen herhangi bir ileti üstbilgisinin boyutu (alınmeyen ileti üstbilgisi, MQDLH ve dağıtım listesi üstbilgisi, MQDH dahil)

Kuyruk yöneticisinin **MaxMsgLength** özniteliği, kuyruk yöneticisinin işleyebileceği ileti boyutunu tutar. Bu, V6 ya da sonraki sürümdeki tüm IBM MQ ürünleri için varsayılan değer olan 100 MB ' ye sahiptir.

Bu özniteliğin değerini saptamak için, kuyruk yöneticisi nesnesinde MQINQ çağrısına bakın. Büyük iletiler için bu değeri değiştirebilirsiniz.

Bir kuyruğun **MaxMsgLength** özniteliği, kuyruğa koyabileceğiniz ileti büyüklüğü üst sınırını belirler. Büyüklüğü bu özniteliğin değerinden büyük olan bir iletiyi koyma girişiminde bulunursanız, MQPUT çağrıınız başarısız olur. Bir iletiyi uzak kuyruğa koyuyorsanız, başarıyla yerleştirebileceğiniz ileti büyüklüğü üst sınırı, uzak kuyruğun **MaxMsgLength** özniteliği, iletinin hedef rotası boyunca konduğu ara iletim kuyrukları ve kullanılan kanallar tarafından belirlenir.

Bir MQPUT işlemi için, iletinin boyutu, hem kuyruğun hem de kuyruk yöneticisinin **MaxMsgLength** özniteliğinden küçük ya da bu özniteliğe eşit olmalıdır. Bu özniteliklerin değerleri bağımsızdır, ancak kuyruğun *MaxMsgLength* değerini kuyruk yöneticisinin değerinden küçük ya da ona eşit bir değere ayarlamanız önerilir.

IBM MQ , aşağıdaki durumlarda iletilere üstbilgi ekler:

- Uzak kuyruğa bir ileti koyduğunuzda, IBM MQ iletiye bir iletim üstbilgisi yapısı (MQXQH) ekler. Bu yapı, hedef kuyruğun adını ve hedef kuyruk yöneticisini içerir.
- IBM MQ bir iletiyi uzak kuyruğa teslim edemezse, iletiyi teslim edilmeyen (teslim edilmeyen) kuyruğa yerleştirmeyi dener. İletiye bir MQDLH yapısı ekler. Bu yapı, hedef kuyruğun adını ve iletinin gönderilmeyen iletiler kuyruğuna konma nedenini içerir.
- Bir iletiyi birden çok hedef kuyruğa göndermek istiyorsanız, IBM MQ iletiye bir MQDH üstbilgisi ekler. Bu, iletim kuyruğundaki bir dağıtım listesine ait iletide bulunan verileri açıklar. İleti uzunluğu üst sınırı için en uygun değeri seçerken bunu göz önünde bulundurun.
- İleti bir gruptaki bir bölüm ya da iletiyse, IBM MQ bir MQMDE ekleyebilir.

Bu yapılar MQDH ve MQMDE' de açıklanmıştır.

İletileriniz bu kuyruklar için izin verilen boyut üst sınırıysa, bu üstbilgilerin eklenmesi, iletiler çok büyük olduğu için koyma işlemlerinin başarısız olduğu anlamına gelir. Koyma işlemlerinin başarısız olma olasılığını azaltmak için:

- İletilerinizin boyutunu, iletimin ve gönderilmeyen iletiler kuyruklarının **MaxMsgLength** özniteliğinden daha küçük yapın. En azından MQ_MSG_HEADER_LENGTH değişmezinin değerine izin ver (büyük dağıtım listeleri için daha fazla).
- Gönderilmeyen iletiler kuyruğunun **MaxMsgLength** özniteliğinin, ileti kuyruğunun sahibi olan kuyruk yöneticisinin *MaxMsgLength* özniteliğiyle aynı değere ayarlandığından emin olun.

Kuyruk yöneticisine ve ileti kuyruklama değişmezlerine ilişkin öznitelikler Kuyruk yöneticisine ilişkin özniteliklerkısmında açıklanmıştır.

z/OS Dağıtılmış bir kuyruğa alma ortamında teslim edilmemiş iletilerin nasıl işlendiğine ilişkin bilgi için bkz. Teslim edilmemiş/işlenmemiş iletiler.

İleti koyma: İleti tanıtıcılarını kullanma

MQPMO yapısında iki ileti tanıtıcısı vardır: *OriginalMsgHandle* ve *NewMsgHandle*. Bu ileti tanıtıcıları arasındaki ilişki, MQPMO *İşlem* alanının değeriyle tanımlanır.

Tüm ayrıntılar için bkz. Action (MQLONG). Bir ileti koymak için ileti tanıtıcısı gerekli değildir. Bunun amacı, özellikleri bir iletiyle ilişkilendirmektir; bu nedenle, yalnızca ileti özelliklerini kullanıyorsanız gereklidir.

Uzak kuyruğa ileti konması

Bir iletiyi yerel bir kuyruk yerine uzak bir kuyruğa (yani, uygulamanızın bağlı olduğu kuyruk yöneticisinden başka bir kuyruk yöneticisine) koymak istediğinizde, yalnızca, kuyruğu açtığınızda kuyruğun adını nasıl belirteceğiniz dikkate alınır.Bu, “Uzak kuyrukların açılması” sayfa 721içinde açıklanmıştır. Yerel bir kuyruk için MQPUT ya da MQPUT1 çağrısına ilişkin herhangi bir değişiklik yoktur.

Uzak ve iletim kuyruklarının kullanılmasıyla ilgili ek bilgi için IBM MQ dağıtılmış kuyruğa alma teknikleribaşlıklı konuya bakın.

İletinin özelliklerini ayarlama

Ayarlamak istediğiniz her özellik için MQSETMP ' yi çağırın. İleti kümesini yerleştirdiğinizde, MQPMO yapısının ileti tanıtıcısı ve işlem alanları.

Özellikleri bir iletiyle ilişkilendirmek için iletinin bir ileti tanıtıcısı olmalıdır. MQCRTMH işlev çağrısını kullanarak bir ileti tanıtıcısı yaratın. Ayarlamak istediğiniz her özellik için bu ileti tanıtıcısını belirterek MQSETMP ' yi çağırın. MQSETMP kullanımını göstermek için amqsstma.cadlı örnek bir program sağlanır.

Bu yeni bir iletiyse, MQPUT ya da MQPUT1kullanarak bir kuyruğa yerleştirdiğinizde, MQPMO ' daki OriginalMsgtanıtıcı alanını bu ileti tanıtıcısının değerine ayarlayın ve MQPMO Action alanını MQACTP_NEW (varsayılan değer budur) olarak ayarlayın.

Bu, daha önce aldığınız bir iletiyse ve şimdi iletiyor, yanıtlıyor ya da yanıt olarak bir rapor gönderiyorsanız, özgün ileti tanıtıcısını MQPMO ' nun OriginalMsgtanıtıcı alanına ve NewMsgtanıtıcı alanındaki yeni ileti tanıtıcısı alanına koyun. İşlem alanını uygun şekilde MQACTP_FORWARD, MQACTP_REPLY ya da MQACTP_REPORT olarak ayarlayın.

Önceden aldığınız bir iletideki MQRFH2 üstbilgisinde özellikler varsa, bunları MQBUFMH çağrısını kullanarak ileti tanıtıcısı özelliklerine dönüştürebilirsiniz.

İletinizi, ileti özelliklerini işleyemeyen IBM WebSphere MQ 7.0düzeyinden önceki bir düzeydeki bir kuyruk yöneticisine yerleştiriyorsanız, kanal tanımlamasında PropertyControl değiştirgesini özelliklerin nasıl işleneceğini belirtecek şekilde ayarlayabilirsiniz.

İleti bağlamı bilgilerinin denetlenmesi

Kuyruğa ileti koymak için MQPUT ya da MQPUT1 çağrısını kullandığınızda, kuyruk yöneticisinin ileti tanımlayıcısına varsayılan bağlam bilgileri ekleyeceğini belirtebilirsiniz. Uygun yetki düzeyine sahip uygulamalar ek bağlam bilgileri ekleyebilir. Bağlam bilgilerini denetlemek için MQPMO yapısındaki seçenekler alanını kullanabilirsiniz.

İleti bağlamı bilgileri, iletiyi alan uygulamanın iletiyi başlatan hakkında bilgi edinmesini sağlar. Tüm bağlam bilgileri, ileti tanımlayıcısının bağlam alanlarında saklanır. Bilgi tipi kimlik, kaynak ve kullanıcı bağlamı bilgilerine dayanır.

Bağlam bilgilerini denetlemek için MQPMO yapısındaki *Options* alanını kullanın.

Bağlam bilgisi için herhangi bir seçenek belirtmezseniz, kuyruk yöneticisi, ileti tanımlayıcısında var olabilecek bağlam bilgilerinin üzerine, iletiniz için oluşturduğu tanıtıcı ve bağlam bilgilerini yazar. Bu, MQPMO_DEFAULT_CONTEXT seçeneğinin belirtilmesiyle aynıdır.Yeni bir ileti yarattığınızda (örneğin, sorgu ekranından kullanıcı girişi işlenirken) bu varsayılan bağlam bilgilerini isteyebilirsiniz.

İletinizle ilişkili bağlam bilgisi istemiyorsanız, MQPMO_NO_CONTEXT seçeneğini kullanın. Bir iletiyi bağlam olmadan koyarken, IBM MQ tarafından yapılan tüm yetki denetimleri boş bir kullanıcı kimliği

kullanılarak yapılır. Boş bir kullanıcı kimliğine IBM MQ kaynakları için belirtik yetki atanamaz, ancak 'kimse' özel grubunun bir üyesi olarak kabul edilir. Özel grup nobodyhakkında daha fazla ayrıntı için bkz. Kurulabilir hizmetler arabirimi başvuru bilgileri.

Aşağıdaki kısımlarda gösterilen MQOO_ seçeneğini ve MQPMO_ seçeneğini kullanarak MQOPEN ve MQPUT seçeneklerini kullanarak bağlam ayarını yapabilirsiniz. Bağlam ayarını yalnızca bir MQPUT1kullanarak da yapabilirsiniz; bu durumda, aşağıdaki kısımlarda gösterilen MQPMO_ seçeneğini belirlemeniz gerekir.

Bu konunun aşağıdaki bölümlerinde kimlik bağlamının, kullanıcı bağlamının ve tüm bağlamın kullanımı açıklanmıştır.

- "Kimlik bağlamı geçirilerek" sayfa 729
- "Kullanıcı bağlamı geçirilir" sayfa 729
- "Tüm bağlam geçirilerek" sayfa 730
- "Kimlik bağlamını ayarlama" sayfa 730
- "Kullanıcı bağlamını ayarlama" sayfa 730
- "Tüm bağlam ayarlanıyor" sayfa 730

Kimlik bağlamı geçirilerek

Genel olarak, programlar, veriler nihai hedefine ulaşıncaya kadar, kimlik bağlamı bilgilerini iletiden uygulamaya aktarmalıdır.

Programlar, verileri her değiştirdiğinde kaynak bağlam bilgilerini değiştirmelidir. Ancak, herhangi bir bağlam bilgisini değiştirmek ya da ayarlamak isteyen uygulamaların uygun yetki düzeyine sahip olmaları gerekir. Kuyruk yöneticisi, uygulamalar kuyrukları açtığında bu yetkiyi denetler; MQOPEN çağrısı için uygun bağlam seçeneklerini kullanma yetkisine sahip olmaları gerekir.

Uygulamanız bir ileti alırsa, iletideki verileri işler ve değiştirilen verileri başka bir iletiye (başka bir uygulama tarafından işlenmek üzere) yerleştirirse, uygulamanın kimlik bağlamı bilgilerini özgün iletiden yeni iletiye aktarması gerekir. Kuyruk yöneticisinin kaynak bağlam bilgilerini yaratmasına izin verebilirsiniz.

Özgün iletideki bağlam bilgilerini saklamak için, iletiyi almak üzere kuyruğu açtığınızda MQOO_SAVE_ALL_CONTEXT seçeneğini kullanın. Bu, MQOPEN çağrısıyla kullandığınız diğer seçeneklere ek olarak kullanılır. Ancak, yalnızca iletiye göz atmanız durumunda bağlam bilgilerini kaydedemeyeceğinizi unutmayın.

İkinci iletiyi oluşturduğunuzda:

- MQOO_PASS_IDENTITY_CONTEXT seçeneğini kullanarak kuyruğu açın (MQOO_OUTPUT seçeneğine ek olarak).
- İleti koyma seçenekleri yapısının *Context* alanında, bağlam bilgilerini sakladığınız kuyruğun tanıtıcısını belirtin.
- Koyma iletisi seçenekleri yapısının *Options* alanında, MQPMO_PASS_IDENTITY_CONTEXT seçeneğini belirtin.

Kullanıcı bağlamı geçirilir

Yalnızca kullanıcı bağlamını geçirmeyi seçemezsiniz. Bir ileti koyarken kullanıcı bağlamını iletmek için MQPMO_PASS_ALL_CONTEXT belirtin. Kullanıcı bağlamındaki özellikler, kaynak bağlamla aynı şekilde iletilir.

Bir MQPUT ya da MQPUT1 oluştuğunda ve bağlam geçirildiğinde, kullanıcı bağlamındaki tüm özellikler alınan iletiden koyma iletisine geçirilir. Koyma uygulamasının değiştirdiği kullanıcı bağlamı özellikleri özgün değerleriyle birlikte konur. Koyma uygulamasının sildiği kullanıcı bağlamı özellikleri, koyma iletisinde geri yüklenir. Koyma uygulamasının iletiye eklediği kullanıcı bağlamı özellikleri korunur.

Tüm bağlam geçirilerek

Uygulamanız bir ileti alır ve ileti verilerini (değiştirilmeden) başka bir iletiye yerleştirirse, uygulamanın özgün iletiden yeni iletiye tüm (kimlik, kaynak ve kullanıcı) bağlam bilgilerini aktarması gerekir. Bunu yapabilen bir uygulama örneği, iletileri bir kuyruktan diğerine taşıyan bir ileti taşıyıcısıdır.

MQOO_PASS_ALL_CONTEXT MQOPEN seçeneğini ve MQPMO_PASS_ALL_CONTEXT koyma iletisi seçeneğini kullanmanız dışında, kimlik bağlamının geçirilmesiyle aynı yordamı izleyin.

Kimlik bağlamını ayarlama

Bir iletiye ilişkin kimlik bağlamı bilgilerini ayarlamak istiyorsanız:

- MQOO_SET_IDENTITY_CONTEXT seçeneğini kullanarak kuyruğu açın.
- MQPMO_SET_IDENTITY_CONTEXT seçeneğini belirterek iletiyi kuyruğa koyun. İleti tanımlayıcıda, gerek duyduğunuz kimlik bağlamı bilgilerini belirtin.

Not: MQOO_SET_IDENTITY_CONTEXT ve MQPMO_SET_IDENTITY_CONTEXT seçeneklerini kullanarak kimlik bağlamı alanlarının bazılarını (ancak tümünü değil) ayarladığınızda, kuyruk yöneticisinin diğer alanların hiçbirini ayarlamadığını fark etmek önemlidir.

İleti bağlamı seçeneklerinden herhangi birini değiştirmek için, çağrıyı yayınlamak için gereken yetkilere sahip olmanız gerekir. Örneğin, MQOO_SET_IDENTITY_CONTEXT ya da MQPMO_SET_IDENTITY_CONTEXT kullanabilmek için `+setid` iznine sahip olmanız gerekir.

Kullanıcı bağlamını ayarlama

Kullanıcı bağlamında bir özellik ayarlamak için, MQSETMP çağrısını yaparken ileti özelliği tanımlayıcısının (MQPD) Bağlam alanını MQPD_USER_CONTEXT olarak ayarlayın.

Kullanıcı bağlamında bir özellik ayarlamak için özel bir yetkiye gerek yoktur. Kullanıcı bağlamında MQOO_SET_ * ya da MQPMO_SET_ * bağlam seçeneği yok.

Tüm bağlam ayarlanıyor

Bir ileti için hem kimlik hem de kaynak bağlam bilgilerini ayarlamak istiyorsanız:

1. MQOO_SET_ALL_CONTEXT seçeneğini kullanarak kuyruğu açın.
2. MQPMO_SET_ALL_CONTEXT seçeneğini belirterek iletiyi kuyruğa koyun. İleti tanımlayıcısında, istediğiniz kimlik ve kaynak bağlamı bilgilerini belirtin.

Her bağlam ayarı tipi için uygun yetki gereklidir.

İlgili kavramlar

"İleti bağlamı" sayfa 45

İleti bağlamı bilgileri, iletiyi alan uygulamanın iletiyi başlatan hakkında bilgi edinmesini sağlar.

İlgili başvurular

"İleti bağlamıyla ilgili MQOPEN seçenekleri" sayfa 720

Bağlam bilgilerini bir kuyruğa yerleştirirken bir iletiyle ilişkilendirmek istiyorsanız, kuyruğu açarken ileti bağlamı seçeneklerinden birini kullanmanız gerekir.

MQPUT1 çağrısı kullanılarak kuyruğa bir ileti konması

Üzerine tek bir ileti koyduktan hemen sonra kuyruğu kapatmak istediğinizde MQPUT1 çağrıyı kullanın. Örneğin, bir sunucu uygulaması farklı kuyrukların her birine yanıt gönderirken MQPUT1 çağrısının kullanılması olasıdır.

MQPUT1 , işlevsel olarak MQOPEN çağrısıyla MQPUT ve ardından MQCLOSE çağrısıyla eşdeğerdir. MQPUT ve MQPUT1 çağrılarının sözdizimindeki tek fark, MQPUT için bir nesne tanıtıcısı belirtirken, MQPUT1 için MQOPEN ' de tanımlandığı şekilde bir nesne tanımlayıcı yapısı (MQOD) belirtirsiniz (bkz. "Nesnelerin tanımlanması (MQOD yapısı)" sayfa 714). Bunun nedeni, MQPUT1 çağrısına, açılması gereken kuyruk hakkında bilgi vermeniz gerekirken, MQPUT çağrıldığında kuyruğun açık olması gerekir.

MQPUT1 çağrısına giriş olarak şunları sağlamanız gerekir:

- Bağlantı tanıtıcısı.
- Açmak istediğiniz nesnenin açıklaması. Bu, bir nesne tanımlayıcı yapısı (MQOD) biçimindedir.
- Kuyruğa koymak istediğiniz iletinin tanımlaması. Bu, ileti tanımlayıcı yapısı (MQMD) biçimindedir.
- Bir koyma iletisi seçenekleri yapısı (MQPMO) biçiminde denetim bilgileri.
- İletide bulunan verilerin uzunluğu (MQLONG).
- İleti verilerinin adresi.

MQPUT1 ' in çıkışı:

- Tamamlanma kodu
- Neden kodu

Çağrı başarıyla tamamlanırsa, seçenek yapınızı ve ileti tanımlayıcı yapınızı da döndürür. Çağrı, kuyruğun adını ve iletinin gönderildiği kuyruk yöneticisini göstermek için seçenek yapınızı değiştirir. Kuyruk yöneticisinin koyduğunuz iletinin tanıtıcısı için benzersiz bir değer oluşturmasını isterseniz (MQMD yapısının *MsgId* alanında ikili sıfır belirterek), çağrı bu yapıyı size döndürmeden önce *MsgId* alanına değeri ekler.

Not: Model kuyruğu adıyla MQPUT1 ' i kullanamazsınız; ancak, bir model kuyruğu açıldıktan sonra, dinamik kuyruğa bir MQPUT1 yayınlayabilirsiniz.

MQPUT1 için altı giriş değiştirgesi şunlardır:

Hconn
Bu bir bağlantı tanıtıcısı. CICS uygulamaları için, değişmez MQHC_DEF_HCONN değerini (sıfır değerine sahip) belirtebilir ya da MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını kullanabilirsiniz. Diğer programlar için her zaman MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını kullanın.

ObjDesc
Bu bir nesne tanımlayıcı yapısıdır (MQOD).

ObjectName ve *ObjectQMgrName* alanlarında, ileti koymak istediğiniz kuyruğun adını ve bu kuyruğun sahibi olan kuyruk yöneticisinin adını belirtin.

Model kuyruklarını kullanamadığı için, MQPUT1 çağrısı için *DynamicQName* alanı yoksayıldı.

Kuyruğu açma yetkisini sınamak için kullanılacak diğer bir kullanıcı kimliğini aday göstermek istiyorsanız *AlternateUserId* alanını kullanın.

MsgDesc
Bu bir ileti tanımlayıcı yapısıdır (MQMD). MQPUT çağrısında olduğu gibi, kuyruğa koymakta olduğunuz iletiyi tanımlamak için bu yapıyı kullanın.

PutMsgOpts
Bu bir koyma iletisi seçenekleri yapısıdır (MQPMO). MQPUT çağrısı için kullandığınız gibi kullanın (bkz. “MQPMO yapısını kullanarak seçeneklerin belirtilmesi” sayfa 724).

Options alanı sıfır olarak ayarlandığında, kuyruk yöneticisi kuyruğa erişim yetkisi için sınama gerçekleştirirken kendi kullanıcı kimliğinizi kullanır. Kuyruk yöneticisi, MQOD yapısının *AlternateUserId* alanında verilen diğer kullanıcı kimliğini de yoksayar.

BufferLength
Bu, iletinizin uzunluğudur.

Buffer
Bu, iletinizin metnini içeren arabellek alanıdır.

Kümeleri kullandığınızda, MQPUT1 MQOO_BIND_NOT_FIXED etkinmiş gibi çalışır. Uygulamaların, iletinin nereye gönderildiğini saptamak için MQOD yapısı yerine MQPMO yapısındaki çözülen alanları kullanması gerekir. Ek bilgi için Kuyruk yöneticisi kümesinin yapılandırılması başlıklı konuya bakın.

MQPUT1içinde MQPUT1 çağrısının bir açıklaması vardır.

Dağıtım listeleri

IBM MQ for z/OSüzerinde desteklenmez. Dağıtım listeleri, bir iletiyi tek bir MQPUT ya da MQPUT1 çağrısında birden çok hedefe yerleştirmenizi sağlar.Tek bir MQOPEN çağrısı birden çok kuyruğu açabilir ve tek bir MQPUT çağrısı bu kuyrukların her birine bir ileti yerleştirebilir. Bu işlem için kullanılan MQI yapılarındaki bazı soysal bilgilerin yerini, dağıtım listesinde yer alan tek tek hedeflerle ilgili belirli bilgiler alabilir.

Uyarı: Dağıtım listeleri, konu nesnelerini işaret eden diğer ad kuyruklarının kullanılmasını desteklemez. Bir diğer ad kuyruğu dağıtım listesindeki bir konu nesnesini gösteriyorsa, IBM MQ `MQRC_ALIAS_BASE_Q_TYPE_ERROR`değerini döndürür.

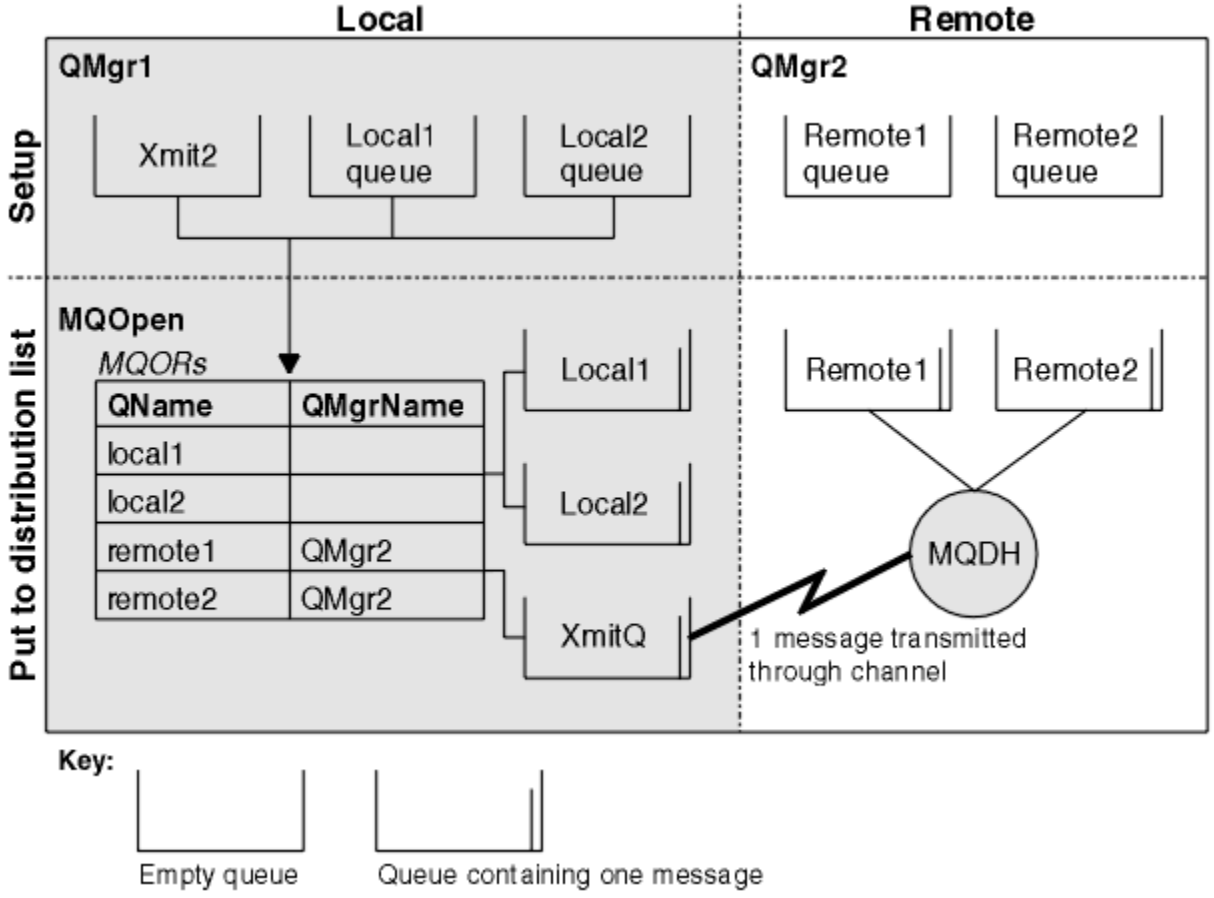
Bir MQOPEN çağrısı verildiğinde, soysal bilgiler Nesne Tanımlayıcısından (MQOD) alınır. *`Version`* alanında MQOD_VERSION_2 ve *`RecsPresent`* alanında sıfırdan büyük bir değer belirtirseniz, *`Hobj`* bir kuyruk yerine bir listenin (bir ya da daha fazla kuyruğun) tanıtıcısı olarak tanımlanabilir. Bu durumda, belirli bilgiler, hedefin ayrıntılarını (*`ObjectName`* ve *`ObjectQMgrName`*) veren nesne kayıtları (MQOR) aracılığıyla verilir.

Nesne tanıtıcısı (*`Hobj`*), MQPUT çağrısına geçirilerek tek bir kuyruk yerine bir listeye yerleştirilmenizi sağlar.

Kuyruklara bir ileti konduğunda (MQPUT), Put Message Option yapısı (MQPMO) ve Message Descriptor (MQMD) soysal bilgileri alınır. Belirli bilgiler, Put Message Records (MQPMR) biçiminde verilir.

Yanıt Kayıtları (MQRR), her hedef kuyruğa özgü bir tamamlanma kodu ve neden kodu alabilir.

Şekil 56 sayfa 732 içinde dağıtım listelerinin nasıl çalıştığı gösterilmektedir.



Şekil 56. Dağıtım listeleri nasıl çalışır?

Dağıtım listelerinin açılması

MQOPEN çağrısıyla bir dağıtım listesi açın ve listeyle ne yapmak istediğinizi belirtmek için çağrı seçeneklerini kullanın.

MQOPEN ' e giriş olarak şunları sağlamalısınız:

- Bağlantı tanıtıcısı (açıklama için bkz. “Kuyruktaki iletilerin konması” sayfa 722)
- Nesne Tanımlayıcı yapısındaki (MQOD) soysal bilgiler
- Nesne Kaydı yapısını (MQOR) kullanarak, açmak istediğiniz her kuyruğun adı

MQOPEN çıkışı:

- Dağıtım listesine erişiminizi gösteren bir nesne tanıtıcısı
- Soysal tamamlanma kodu
- Soysal neden kodu
- Her hedef için bir tamamlanma kodu ve nedeni içeren Yanıt Kayıtları (isteğe bağlı)

MQOD yapısının kullanılması

Açmak istediğiniz kuyrukları tanımlamak için MQOD yapısını kullanın.

Bir dağıtım listesi tanımlamak için *Version* alanında MQOD_VERSION_2 , *RecsPresent* alanında sıfırdan büyük bir değer ve *ObjectType* alanında MQOT_Q belirtmeniz gerekir. MQOD yapısının tüm alanlarına ilişkin açıklamalar için MQOD kısmına bakın.

MQOR yapısının kullanılması

Her hedef için bir MQOR yapısı sağlayın.

Yapı, hedef kuyruk ve kuyruk yöneticisi adlarını içerir. MQOD içindeki *ObjectName* ve *ObjectQMgrName* alanları dağıtım listeleri için kullanılmaz. Bir ya da daha çok nesne kaydı olmalıdır. *ObjectQMgrName* boş bırakılırsa, yerel kuyruk yöneticisi kullanılır. Bu alanlarla ilgili daha fazla bilgi için bkz. ObjectName ve ObjectQMgrAd .

Hedef kuyrukları iki şekilde belirtebilirsiniz:

- *ObjectRecOffset*göreli konum alanını kullanarak.

  Bu durumda, uygulamanın MQOD yapısı içeren kendi yapısını bildirmesi ve ardından MQOR kayıtları dizisi (gerektiği kadar dizi öğesi ile) ve *ObjectRecOffset* MQOD ' un başlangıcından itibaren dizideki ilk öğenin göreli konumuna ayarlanması gerekir. Bu göreli konumların doğru olduğundan emin olun.

  Uygulamanın çalıştığı tüm ortamlarda kullanılabiliyorsa, programlama dili tarafından sağlanan yerleşik olanakların kullanılması önerilir. Aşağıdaki kod, COBOL programlama dili için bu tekniği gösterir:

```
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.
```

  Diğer bir seçenek olarak, programlama dili ilgili tüm ortamlarda gerekli yerleşik olanakları desteklemiyorsa, MQOD_CURRENT_LENGTH değişmezini kullanın. Aşağıdaki kod bu tekniği gösterir:

```
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.
```

  Ancak bu yalnızca MQOD yapısı ve MQOR kayıtları dizisi bitişikse doğru çalışır; derleyici MQOD ile MQOR dizisi arasına byte 'ları atlarsa, bunlar *ObjectRecOffset*içinde saklanan değere eklenmelidir.

İşaretçi veri tipini desteklemeyen ya da gösterge veri tipini farklı ortamlara (örneğin, COBOL programlama dili) taşınmayacak şekilde uygulayan programlama dilleri için *ObjectRecOffset* kullanılması önerilir.

- *ObjectRecPtr*gösterge alanını kullanarak.

  Bu durumda, uygulama MQOR yapılarının dizisini MQOD yapısından ayrı olarak bildirebilir ve *ObjectRecPtr* değerini dizinin adresine ayarlayabilir. Aşağıdaki kod, C programlama dili için bu tekniği gösterir:

```
MQOD MyMqod;
MQOR MyMqor[100];
MyMqod.ObjectRecPtr = MyMqor;
```

  Gösterge veri tipini farklı ortamlara (örneğin, C programlama dili) taşınabilir bir şekilde destekleyen programlama dilleri için *ObjectRecPtr* kullanılması önerilir.

Hangi tekniği seçerseniz seçin, *ObjectRecOffset* ve *ObjectRecPtr* yöntemlerinden birini kullanmanız gerekir; Her ikisi de sıfırsa ya da her ikisi de sıfır değilse, çağrı MQRC_OBJECT_RECORDS_ERROR neden koduyla başarısız olur.

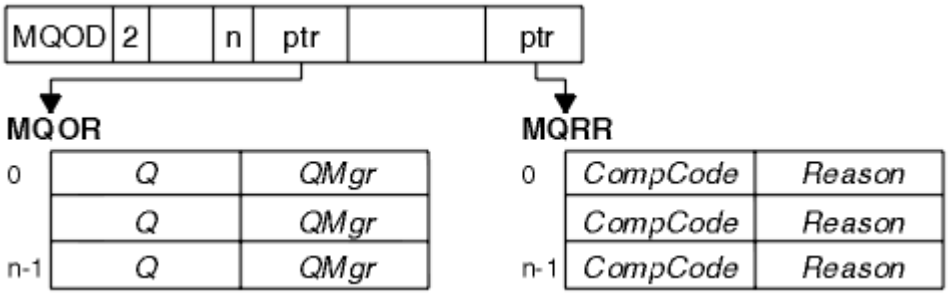
MQRR yapısının kullanılması

Bu yapılar hedefe özgüldür; her Yanıt Kaydı, dağıtım listesinin her kuyruğu için bir *CompCode* ve *Reason* alanı içerir. Herhangi bir sorunun nerede olduğunu ayırt edebilmeniz için bu yapıyı kullanmalısınız.

Örneğin, MQRC_MULTIPLE_REASON kodunu alırsanız ve dağıtım listeniz beş hedef kuyruk içeriyorsa, bu yapıyı kullanmazsanız, sorunların hangi kuyruklar için geçerli olduğunu bilmezsiniz. Ancak, her hedef için bir tamamlanma kodunuz ve neden kodunuz varsa, hataları daha kolay bulabilirsiniz.

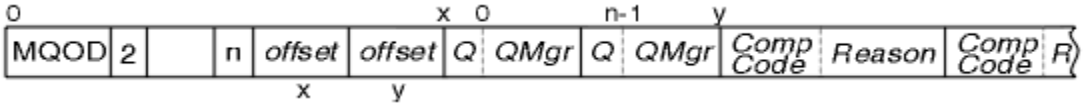
MQRR yapısına ilişkin ek bilgi için MQRR belgesine bakın.

Şekil 57 sayfa 734 içinde C ' de bir dağıtım listesini nasıl açabileceğiniz gösterilmektedir.



Şekil 57. Dağıtım listesini C ' de açma

Şekil 58 sayfa 734 , COBOL içinde bir dağıtım listesini nasıl açabileceğinizi gösterir.



Şekil 58. COBOL ' da bir dağıtım listesi açma

MQOPEN seçeneklerinin kullanılması

Dağıtım listesini açarken aşağıdaki seçenekleri belirleyebilirsiniz:

- MQOO_ÇıKışı
- MQOO_FAIL_IF_QUIESCING (isteğe bağlı)
- MQOO_ALTERNATE_USER_AUTHORITY (isteğe bağlı)
- MQOO_ * _CONTEXT (isteğe bağlı)

Bu seçeneklerin açıklaması için bkz. “Nesnelerin açılması ve kapatılması” sayfa 712 .

Dağıtım listesine ileti konması
İletileri bir dağıtım listesine koymak için MQPUT ya da MQPUT1kullanabilirsiniz.

Giriş olarak şunları sağlamanız gerekir:

- Bağlantı tanıtıcısı (açıklama için bkz. “Kuyruktaki iletilerin konması” sayfa 722).
- Bir nesne tanıtıcısı. Bir dağıtım listesi MQOPEN kullanılarak açılırsa, *Hobj* yalnızca listeye koymanıza izin verir.
- İleti tanımlayıcı yapısı (MQMD). Bu yapının açıklaması için bkz. MQMD .
- Bir koyma iletisi seçeneği yapısı (MQPMO) biçiminde denetim bilgileri. MQPMO yapısının alanlarını doldurmaya ilişkin bilgi için bkz. “MQPMO yapısını kullanarak seçeneklerin belirtilmesi” sayfa 724 .
- Koyma İletisi Kayıtları (MQPMR) biçiminde denetim bilgileri.
- İletide bulunan verilerin uzunluğu (MQLONG).
- İleti verilerinin kendisi.

Çıkış:

- Tamamlanma kodu
- Neden kodu
- Yanıt Kayıtları (isteğe bağlı)

MQPMR yapısının kullanılması

Bu yapı isteğe bağlıdır ve MQMD ' de önceden tanımlananlardan farklı olarak tanımlamak isteyebileceğiniz bazı alanlara ilişkin hedefe özgü bilgileri verir.

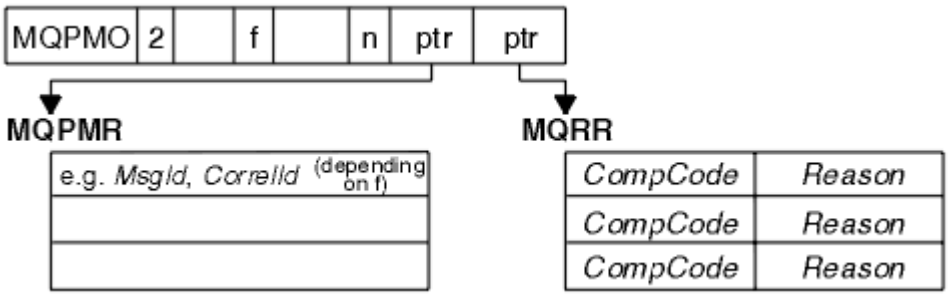
Bu alanların tanımı için bkz. MQPMR.

Her kaydın içeriği, MQPMO ' nun *PutMsgRecFields* alanında verilen bilgilere bağlıdır. Örneğin, AMQSPTL0.C (tanım için bkz. “Dağıtım Listesi örnek programı” sayfa 1043) Dağıtım listelerinin kullanımını gösteren örnek, MQPMR ' de *MsgId* ve *CorrelId* için değer sağlamayı seçer. Örnek programın bu bölümü şöyle görünür:

```
typedef struct
{
MQBYTE24 MsgId;
MQBYTE24 CorrelId;
} PutMsgRec;
...
/**********************
MQLONG PutMsgRecFields=MQPMRF_MSG_ID | MQPMRF_CORREL_ID;
```

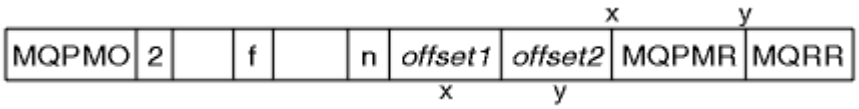
Bu, dağıtım listesinin her hedefi için *MsgId* ve *CorrelId* ' in sağlandığı anlamına gelir. Koyma İletisi Kayıtları bir dizi olarak sağlanır.

Şekil 59 sayfa 735 içinde bir iletiyi C içindeki bir dağıtım listesine nasıl yerleştirebileceğiniz gösterilmektedir.



Şekil 59. C ' deki bir dağıtım listesine ileti konması

Şekil 60 sayfa 736 , COBOL içindeki bir dağıtım listesine bir iletiyi nasıl koyabileceğinizi gösterir.



Şekil 60. COBOL ' da dağıtım listesine ileti konması

MQPUT1 ' i kullanma

MQPUT1kullanıyorsanız, aşağıdaki noktaları göz önünde bulundurun:

1. *ResponseRecOffset* ve *ResponseRecPtr* alanlarının değerleri boş değerli ya da sıfır olmalıdır.
2. Gerekiyorsa, Yanıt Kayıtları MQOD ' den adreslenmelidir.

Bazı durumlarda, yanıtların başarısız olduğu durumlar

MQOPEN ve MQPUT çağrısı verme aralığınız sırasında bir komutta FORCE seçeneği kullanılarak bir kuyruğun belirli öznitelikleri değiştirilirse, MQPUT çağrısı başarısız olur ve MQRC_OBJECT_CHANGED neden kodunu döndürür.

Kuyruk yöneticisi, nesne tanıtıcısını artık geçerli değil olarak işaretler. Bu durum, MQPUT1 çağrısı işlenirken ya da değişiklikler, kuyruk adının çözüldüğü herhangi bir kuyruğa uygulanırsa da oluşur. Tanıtıcıyı bu şekilde etkileyen öznitelikler, MQOPENiçinde MQOPEN çağrısının tanımında listelenir. Çağrınız MQRC_OBJECT_CHANGED neden kodunu döndürürse, kuyruğu kapatın, yeniden açın ve iletiyi yerleştirmeyi yeniden deneyin.

İletileri (ya da kuyruk adının çözüldüğü herhangi bir kuyruk) koymayı denediğiniz bir kuyruk için koyma işlemleri engellenirse, MQPUT ya da MQPUT1 çağrısı başarısız olur ve MQRC_PUT_INENGELLENEN neden kodunu döndürür. Daha sonra arama girişiminde bulunursanız, uygulamanın tasarımı, diğer programların kuyrukların özniteliklerini düzenli olarak değiştirmesini sağlayacaksa, iletiyi başarıyla koyabilirsiniz.

Furthemore, iletinizi yerleştirmeye çalıştığınız kuyruk doluysa, MQPUT ya da MQPUT1 çağrısı başarısız olur ve MQRC_Q_FULL değerini döndürür.

Devingen bir kuyruk (geçici ya da kalıcı) silindiyse, önceden edinilen bir nesne tanıtıcısını kullanan MQPUT çağrıları başarısız olur ve MQRC_Q_DELETED neden kodunu döndürür. Bu durumda, artık size bir yararı olmayacağı için nesne tutamacını kapatmak iyi bir uygulamadır.

Dağıtım listeleri söz konusu olduğunda, tek bir istekte birden çok tamamlanma kodu ve neden kodu oluşabilir. Bunlar yalnızca MQOPEN ve MQPUT ' daki *CompCode* ve *Reason* çıkış alanları kullanılarak işlenemez.

İletileri birden çok hedefe koymak için dağıtım listelerini kullandığınızda, Yanıt Kayıtları her hedef için belirli *CompCode* ve *Reason* öğelerini içerir. MQCC_FAILED ' in tamamlanma kodunu alırsanız, hiçbir hedef kuyruğa başarıyla ileti konmaz. Tamamlanma kodu MQCC_WARNING ise, ileti hedef kuyruklardan birine ya da daha fazlasına başarıyla konmuştur. MQRC_MULTIPLE_REASON dönüş kodunu alırsanız, neden kodlarının tümü her hedef için aynı değildir. Bu nedenle, hataya neden olan kuyruğu ya da kuyrukları saptamak için MQRR yapısını kullanmanız önerilir.

Kuyruktan ileti alma

Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

Bir kuyruktan iki şekilde ileti alabilirsiniz:

1. Diğer programların artık göremeyeceği bir iletiyi kuyruktan kaldırabilirsiniz.
2. Özgün iletiyi kuyrukta bırakarak bir iletiyi kopyalayabilirsiniz. Bu, *göz atma*olarak bilinir. Göz attıktan sonra iletiyi kaldırabilirsiniz.

Her iki durumda da MQGET çağrıyı kullanırsınız, ancak önce uygulamanızın kuyruk yöneticisine bağlanması ve kuyruğu açmak (giriş, göz atma ya da her ikisi için) için MQOPEN çağrıyı kullanmanız gerekir. Bu işlemler “Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705 ve “Nesnelerin açılması ve kapatılması” sayfa 712içinde açıklanır.

Kuyruğu açtığınızda, aynı kuyruktaki iletilere göz atmak ya da iletileri kaldırmak için MQGET çağrısını yinelemeli olarak kullanabilirsiniz. Kuyruktan istediğiniz tüm iletileri almayı bitirdiğinizde MQCLOSE ' yi çağırın.

Kuyruktan ileti almaya ilişkin ek bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “MQGET çağrıyı kullanarak kuyruktan ileti alma” sayfa 737
- “İletilerin kuyruktan alınma sırası” sayfa 741
- “Belirli bir iletiyi alma” sayfa 753
- “Kalıcı olmayan iletilerin performansını artırma” sayfa 754
- z/OS “Dizin tipi” sayfa 758
- “4 MB ' den uzun iletileri işleme” sayfa 759
- “İleti bekleniyor” sayfa 764
- z/OS “Sinyal” sayfa 765
- “Geriletme atlanıyor” sayfa 766
- “Uygulama verilerini dönüştürme” sayfa 769
- “Kuyruktaki iletilere göz atma” sayfa 770
- “MQGET çağrılarının başarısız olduğu bazı durumlar” sayfa 775

İlgili kavramlar

“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

“Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

“Kuyruktaki iletilerin konması” sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

“Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

“Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

“MQI ve kümelerle çalışma” sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

“IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

“IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları” sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

MQGET çağrıyı kullanarak kuyruktan ileti alma

MQGET çağrısı, açık bir yerel kuyruktan bir ileti alır. Başka bir sistemdeki bir kuyruktan ileti alamaz.

MQGET çağrısına giriş olarak şunları sağlamanız gerekir:

- Bağlantı tanıtıcısı.

- Bir kuyruk tanıtıcısı.
- Kuyruktan almak istediğiniz iletinin tanımlaması. Bu, ileti tanımlayıcı (MQMD) yapısı biçimindedir.
- İleti Alma Seçenekleri (MQGMO) yapısı biçiminde denetim bilgileri.
- İletiyi tutmak için atadığınız arabelleğin büyüklüğü (MQLONG).
- İletinin konacağı saklama alanının adresi.

MQGET çıkışı:

- Neden kodu
- Tamamlanma kodu
- Arama başarıyla tamamlanırsa, belirttiğiniz arabellek alanındaki ileti
- Seçenek yapınız, iletinin alındığı kuyruğun adını gösterecek şekilde değiştirildi
- Alınan iletiyi tanımlamak için değiştirilen alanların içeriğiyle birlikte ileti tanımlayıcı yapınız
- İletinin uzunluğu (MQLONG)

MQGETiçinde MQGET çağrısının bir açıklaması var.

Aşağıdaki kısımlarda, MQGET çağrısına giriş olarak sağlamanız gereken bilgiler açıklanmaktadır.

- “Bağlantı tanıtıcılarının belirtilmesi” sayfa 738
- “MQMD yapısını ve MQGET çağrısıyla iletileri tanımlama” sayfa 738
- “MQGMO yapısını kullanarak MQGET seçeneklerinin belirtilmesi” sayfa 738
- “Arabellek alanının büyüklüğünün belirtilmesi” sayfa 741

Bağlantı tanıtıcılarının belirtilmesi

z/OS For CICS on z/OS applications, you can specify the constant MQHC_DEF_HCONN (which has the value zero), or use the connection handle returned by the MQCONN or MQCONNX call. Diğer uygulamalar için her zaman MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını kullanın.

MQOPEN çağırdığınızda döndürülen kuyruk tanıtıcısını (*Hobj*) kullanın.

MQMD yapısını ve MQGET çağrısıyla iletileri tanımlama

Bir kuyruktan almak istediğiniz iletiyi tanımlamak için ileti tanımlayıcı yapısını (MQMD) kullanın.

Bu, MQGET çağrısına ilişkin bir giriş/çıkış değiştirgesidir. MQMD 'nin “IBM MQ ileti” sayfa 17içinde tanımladığı ileti özelliklerine bir giriş vardır ve MQMD' de yapının kendisinin bir açıklaması vardır.

Kuyruktan hangi iletiyi almak istediğinizi biliyorsanız, bkz. “Belirli bir iletiyi alma” sayfa 753.

Belirli bir ileti belirtmezseniz, MQGET kuyruktaki *ilk* iletiyi alır. “İletilerin kuyruktan alınma sırası” sayfa 741 içinde bir iletinin önceliğinin, kuyruğun **MsgDeliverySequence** özniteliğinin ve MQGMO_LOGICAL_ORDER seçeneğinin kuyruktaki iletilerin sırasını nasıl belirlediği açıklanmaktadır.

Not: MQGET ' i bir kereden fazla kullanmak istiyorsanız (örneğin, kuyruktaki iletilerde adım adım ilerlemek için), her çağrıdan sonra bu yapının *MsgId* ve *CorrelId* alanlarını boş değere ayarlamanız gerekir. Bu, alınan iletinin tanıtıcılarının bu alanlarını temizler.

Ancak, iletilerinizi gruplamak istiyorsanız, *GroupId* aynı gruptaki iletiler için aynı olmalıdır; böylece çağrı, tüm grubu oluşturabilmek için önceki iletiyle aynı tanıtıcılara sahip bir iletiyi arar.

MQGMO yapısını kullanarak MQGET seçeneklerinin belirtilmesi

MQGMO yapısı, seçenekleri MQGET çağrısına geçirmek için kullanılan bir giriş/çıkış değişkenidir. Aşağıdaki bölümler, bu yapının bazı alanlarını doldurmanıza yardımcı olur.

MQGMOiçinde MQGMO yapısının bir açıklaması vardır.

StrucId
StrucId , yapıyı bir alma iletisi seçenekleri yapısı olarak tanımlamak için kullanılan 4 karakterlik bir alandır. MQGMO_STRUC_ID değerini her zaman belirtin.

Version
*Version*yapının sürüm numarasını açıklar. MQGMO_VERSION_1 varsayılan değerdir. Sürüm 2 alanlarını kullanmak ya da iletileri mantıksal sırayla almak istiyorsanız, MQGMO_VERSION_2belirtin. Sürüm 3 alanlarını kullanmak ya da iletileri mantıksal sırayla almak istiyorsanız, MQGMO_VERSION_3belirtin. MQGMO_CURRENT_VERSION, uygulamanızı en son düzeyi kullanacak şekilde ayarlar.

Options
Kodunuzda, seçenekleri istediğiniz sırada belirleyebilirsiniz; her seçenek *Options* alanında bir bit ile gösterilir.

Options alanı aşağıdakileri denetler:

- MQGET çağrısı tamamlanmadan önce bir iletinin kuyruğa gelmesini bekleyip beklemeyeceğini belirler (bkz. “İleti bekleniyor” sayfa 764)
- Alma işleminin bir iş birimine dahil edilip edilmediğini belirler.
- Hızlı ileti alışverişini sağlayan, eşitleme noktası dışında kalıcı olmayan bir iletinin alınıp alınmayacağını belirler
- z/OS IBM MQ for z/OS' da, alınan iletinin geri alma atlanıyor olarak işaretlenip işaretlenmediğini belirtir (bkz. “Geriletme atlanıyor” sayfa 766)
- İletinin kuyruktan kaldırılıp kaldırılmadığını ya da yalnızca göz atılıp atılmadığını belirler
- Göz atma imlecini kullanarak mı, yoksa başka seçim ölçütlerine göre mi ileti seçileceğini belirler
- İleti arabelleğinizden uzun olsa da aramanın başarılı olup olmayacağını belirler
- z/OS IBM MQ for z/OSüzerinde, aramanın tamamlanmasına izin verilip verilmeyeceğini belirler. Bu seçenek, bir ileti geldiğinde bildirim almak istediğinizi belirten bir işaret de ayarlar.
- Kuyruk yöneticisi susturma durumundaysa çağrıların başarısız olup olmayacağını belirler
- z/OS IBM MQ for z/OS' da, bağlantı susturma durumundaysa çağrı başarısız olup olmayacağını belirler
- Uygulama iletisi veri dönüştürmesi gerekip gerekmediğini belirler (bkz. “Uygulama verilerini dönüştürme” sayfa 769)
- İletilerin ve bölümlerin bir kuyruktan alınma sırası z/OS (IBM MQ for z/OS dışında)
- Tamamlansın, mantıksal iletiler yalnızca alınabilir z/OS (IBM MQ for z/OS dışında)
- Gruptaki iletilerin yalnızca gruptaki *tüm* iletiler kullanılabilir olduğunda alınıp alınamayacağını belirler
- Mantıksal iletideki bölümlerin yalnızca mantıksal iletideki *tüm* bölümleri kullanılabilir olduğunda z/OS (IBM MQ for z/OS dışında) alınıp alınamayacağını belirler.

Options alanını varsayılan değere (MQGMO_NO_WAIT) ayarlı bırakırsanız, MQGET çağrısı şu şekilde çalışır:

- Kuyruktaki seçim ölçütlerinizle eşleşen bir ileti yoksa, çağrı bir iletinin gelmesini beklemez, ancak hemen tamamlanır. z/OS Ayrıca, IBM MQ for z/OSiçinde arama, böyle bir ileti geldiğinde bildirim isteyen bir sinyal ayarlamaz.
- Aramanın eşitleme noktalarıyla çalışma şekli platform tarafından belirlenir:

Hizmet olarak sunulan	**Eşitleme noktası denetimi altında**
IBM i	Hayır

Hizmet olarak sunulan	Eşitleme noktası denetimi altında
AIX and Linux sistemleri	Hayır
z/OS z/OS z/OS	Evet
Windows sistemleri	Hayır

- z/OS IBM MQ for z/OS' da alınan ileti, geri alma atlanıyor olarak işaretlenmez.
- Seçilen ileti kuyruktan kaldırılır (göz atılmaz).
- Uygulama iletisi verilerinin dönüştürülmesi gerekmez.
- İleti arabelleğinizden uzunsa çağrı başarısız olur.

WaitInterval
WaitInterval alanı, MQGET çağrımının, MQGMO_WAIT seçeneğini kullandığınızda kuyruğa bir ileti gelmesi için bekleyeceği sürenin üst sınırını (milisaniye cinsinden) belirtir. *WaitInterval*içinde belirtilen süre içinde herhangi bir ileti gelmezse, çağrı tamamlanır ve kuyruktaki seçim ölçütlerinizle eşleşen bir ileti olmadığını gösteren bir neden kodu döndürülür.

z/OS IBM MQ for z/OS' ta MQGMO_SET_SIGNAL seçeneğini kullanırsanız, *WaitInterval* alanı sinyalin ayarlandığı saati belirtir.

Bu seçenekler hakkında daha fazla bilgi için bkz. “İleti bekleniyor” sayfa 764 z/OS ve “Sinyal” sayfa 765 .

z/OS ***Signal1***
Signal1 yalnızca IBM MQ for z/OSüzerinde desteklenir.

Uygun bir ileti geldiğinde uygulamanızın bilgilendirilmesini istemek için MQGMO_SET_SIGNAL seçeneğini kullanırsanız, *Signal1* alanında sinyal tipini belirtirsiniz. Diğer tüm platformlarda IBM MQ içinde, *Signal1* alanı ayrılmıştır ve değeri önemli değildir.

z/OS Daha fazla bilgi için bkz “Sinyal” sayfa 765.

Signal2
Signal2 alanı tüm platformlarda ayrılmıştır ve değeri önemli değildir.

z/OS Daha fazla bilgi için “Sinyal” sayfa 765 başlıklı konuya bakın.

ResolvedQName
ResolvedQName , kuyruk yöneticisinin iletinin alındığı kuyruğun adını (herhangi bir diğer adın çözülmesinden sonra) döndürdüğü bir çıkış alanıdır.

MatchOptions
MatchOptions , MQGET için seçim ölçütlerini denetler.

GroupStatus
GroupStatus , aldığınız iletinin bir grupta olup olmadığını gösterir.

SegmentStatus
SegmentStatus , aldığınız öğenin bir mantıksal iletinin parçası olup olmadığını gösterir.

Segmentation
Segmentation , alınan ileti için bölümlemeye izin verilip verilmediğini belirtir.

MsgToken

MsgToken , bir iletiyi benzersiz olarak tanımlar.

ReturnedLength
ReturnedLength , kuyruk yöneticisinin döndürülen ileti verilerinin uzunluğunu (bayt cinsinden) döndürdüğü bir çıkış alanıdır.

MsgHandle
Kuyruktan alınmakta olan iletinin özellikleriyle doldurulacak bir iletinin tanıtıcısı. Tanıtıcı daha önce bir MQCRTMH çağrısıyla yaratılmış. Bir ileti alınmadan önce, tanıtıcı ile önceden ilişkilendirilmiş özellikler temizlenir.

Arabellek alanının büyüklüğünün belirtilmesi

MQGET çağrısının **BufferLength** değiştirgesinde, aldığınız ileti verilerini bulunduracak arabellek alanının büyüklüğünü belirtin. Bunun ne kadar büyük olacağına üç şekilde karar verirsiniz:

1. Bu programdan beklenecek iletilerin uzunluğunu önceden biliyor olabilirsiniz. Bu durumda, bu büyüklükte bir arabellek belirtin.

   Ancak, MQGET çağrısının arabellek için çok büyük olsa da tamamlanmasını istiyorsanız, MQGMO yapısında MQGMO_ACCEPT_TRUNCATED_MSG seçeneğini kullanabilirsiniz. Bu durumda:

   - Arabellek, tutabildiği kadar iletiyle dolu
   - Çağrı bir uyarı tamamlama kodu döndürür
   - İleti kuyruktan kaldırılır (iletinin geri kalanı atılır) ya da göz atma imleci gelişir (kuyruğa göz atıyorsanız)
   - İletinin gerçek uzunluğu *DataLength* içinde döndürülür.

   Bu seçenek olmadan, çağrı bir uyarıyla tamamlanır, ancak iletiyi kuyruktan kaldırmaz (ya da göz atma imlecini ilerletmez).
2. Arabellek için bir büyüklük tahmin edin (ya da sıfır byte 'lık bir büyüklük belirtin) ve MQGMO_ACCEPT_TRUNCATED_MSG seçeneğini *kullanmayın* . MQGET çağrısı başarısız olursa (örneğin, arabellek çok küçük olduğu için), çağrı **DataLength** değiştirgesinde iletinin uzunluğu döndürülür. (Arabellek, tutabildiği kadar iletiyle dolu, ancak çağrı işleme tamamlanmadı.) Bu iletinin *MsgId* değerini saklayın ve doğru büyüklükte bir arabellek alanı belirterek MQGET çağrıyı ve ilk çağrıda not ettiğiniz *MsgId* öğesini yineleyin.

   Programınız başka programlar tarafından da hizmet verilen bir kuyruğa hizmet veriyorsa, bu programlardan biri, programınız başka bir MQGET çağrısı yayınlamadan önce istediğiniz iletiyi kaldırabilir. Programınız artık var olmayan bir iletiyi aramak için zaman kaybedebilir. Bunu önlemek için, önce istediğiniz iletiyi buluncaya kadar kuyruğa göz atın, *BufferLength* değerini belirtin ve MQGMO_ACCEPT_TRUNCATED_MSG seçeneğini kullanın. Bu işlem, göz atma imlecini istediğiniz iletinin altına konumlar. Daha sonra, MQGMO_MSG_UNDER_CURSOR seçeneğini belirterek MQGET ' i yeniden çağırarak iletiyi alabilirsiniz. Başka bir program göz atma ve kaldırma çağrılarınız arasındaki iletiyi kaldırırsa, göz atma imleciniz altında ileti olmadığı için ikinci MQGET işlemi hemen başarısız olur (tüm kuyrukta arama yapmadan).
3. *MaxMsgLength kuyruk* özniteliği, o kuyruk için kabul edilen ileti uzunluğu üst sınırını belirler; *MaxMsgLength kuyruk yöneticisi* özniteliği, o kuyruk yöneticisi için kabul edilen ileti uzunluğu üst sınırını belirler. İletinin hangi uzunlukta bekleyeceğini bilmiyorsanız, **MaxMsgLength** özniteliği hakkında bilgi alabilir (MQINQ çağrıyı kullanarak) ve bu büyüklükte bir arabellek belirtebilirsiniz.

   Başarımın düşmesini önlemek için, arabellek büyüklüğünü gerçek ileti boyutuna olabildiğince yakın yapmaya çalışın.

   MaxMsgLength özniteliğiyle ilgili daha fazla bilgi için bkz. “İleti uzunluğu üst sınırının artırılması” sayfa 759.

İletilerin kuyruktan alınma sırası

Bir kuyruktan ileti alma sırasını denetleyebilirsiniz. Bu bölüm seçeneklere bakar.

Öncelik

Bir program, iletiyi kuyruğa koyduğunda iletiye öncelik atayabilir (bkz. “İleti öncelikleri” sayfa 25). Eşit önceliğe sahip iletiler, kesinleştirildikleri sırada değil, varış sırasına göre bir kuyrukta saklanır.

Kuyruk yöneticisi, kuyrukları katı FIFO (ilk giren ilk çıkar) sırasına ya da FIFO (ilk giren ilk çıkar) sırasına göre öncelik sırasına göre tutar. Bu, kuyruğun **MsgDeliverySequence** özniteliğinin ayarına bağlıdır. Kuyruğa bir ileti geldiğinde, aynı önceliğe sahip son iletiden hemen sonra eklenir.

Programlar, kuyruktan ilk iletiyi alabilir ya da belirli bir iletiyi kuyruktan alabilir ve bu iletilerin önceliğini yoksayabilir. Örneğin, bir program daha önce gönderdiği belirli bir iletinin yanıtını işlemek isteyebilir. Daha fazla bilgi için “Belirli bir iletiyi alma” sayfa 753 başlıklı konuya bakın.

Bir uygulama bir ileti dizisini kuyruğa koyarsa, başka bir uygulama bu iletileri, yerleştirildikleri sırayla alabilir:

- Tüm iletilerin önceliği aynı
- İletilerin tümü aynı iş birimine konmuş ya da tümü bir iş biriminin dışına konmuş.
- Kuyruk, koyma uygulaması için yereldir

Bu koşullar karşılanmazsa ve uygulamalar belirli bir sırada alınan iletilere bağlı olarak, uygulamaların ileti verilerinde sıralama bilgilerini içermesi ya da bir sonraki gönderilmeden önce bir iletinin alındığını kabul etmek için bir yol oluşturması gerekir.

z/OS IBM MQ for z/OSüzerinde, kuyruktaki MQGET işlemlerinin hızını artırmak için kuyruk özniteliğini (*IndexType*) kullanabilirsiniz. Daha fazla bilgi için “Dizin tipi” sayfa 758 başlıklı konuya bakın.

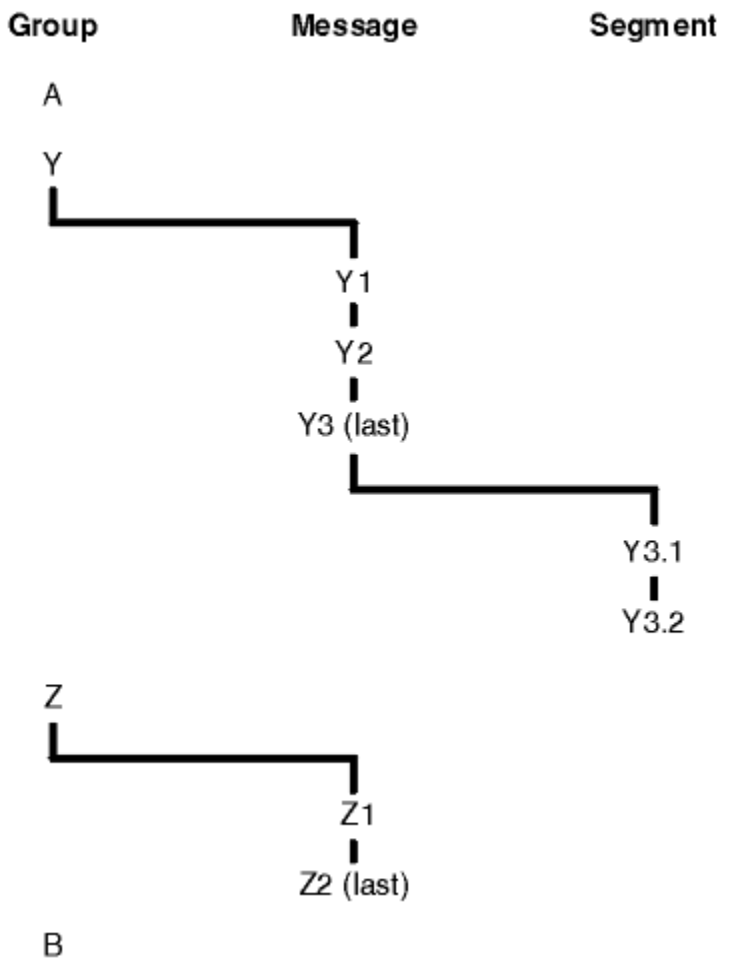
Mantıksal ve fiziksel sıralama
Kuyruklardaki iletiler (her bir öncelik düzeyi içinde) *fiziksel* ya da *mantıksal* sırada oluşabilir.

Fiziksel sıra, iletilerin kuyruğa ulaşma sırasıdır. Mantıksal sıra, bir gruptaki tüm iletilerin ve bölümlerin, gruba ait ilk öğenin fiziksel konumunun belirlediği konumda, yan yana mantıksal sıralarında olması durumudur.

Grupların, iletilerin ve bölümlerin açıklaması için bkz. “İleti grupları” sayfa 42. Bu fiziksel ve mantıksal siparişler farklı olabilir, çünkü:

- Gruplar, farklı uygulamalardan benzer zamanlarda bir hedefe ulaşabilirler, bu nedenle herhangi bir fiziksel sırayı kaybedebilirler.
- Tek bir grup içinde bile, gruptaki bazı iletilerin yeniden yönlendirilmesinden veya geciktirilmesinden dolayı iletiler sırasız olabilir.

Örneğin, mantıksal sıra Şekil Şekil 61 sayfa 743gibi görünebilir:

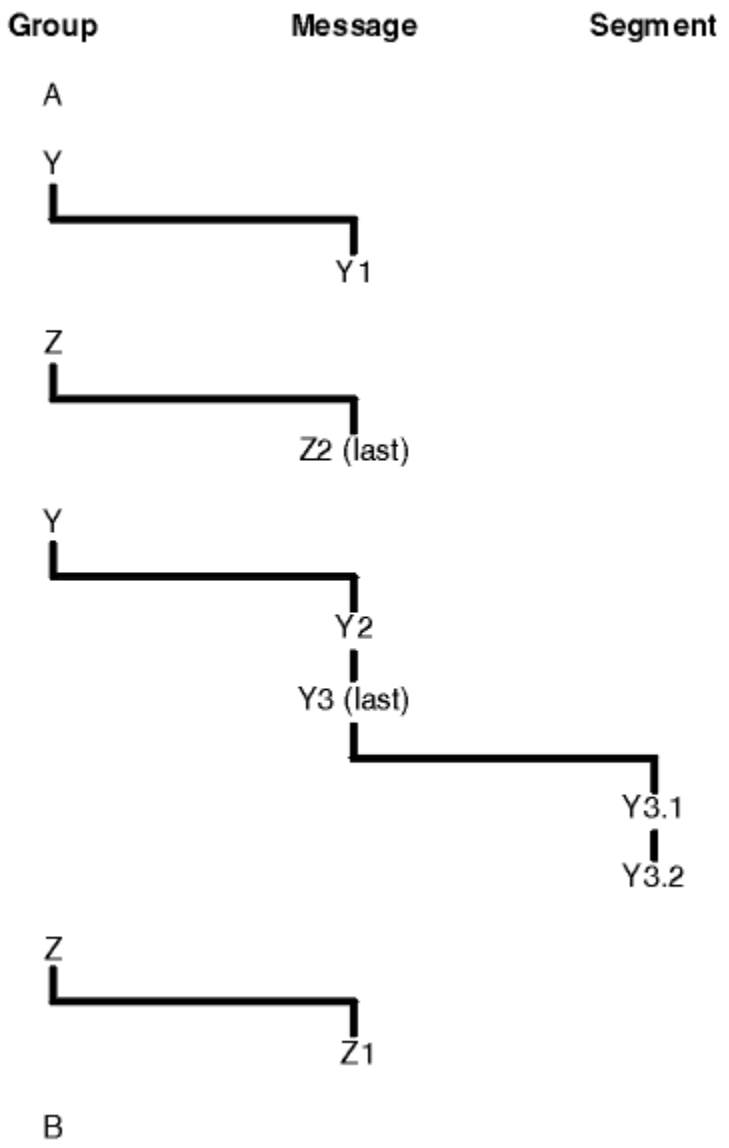


Şekil 61. Kuyruktaki mantıksal sıra

Bu iletiler, bir kuyrukta aşağıdaki mantıksal sırada gerçekleşir:

1. İleti A (bir grupta değil)
2. Y grubunun mantıksal iletisi 1
3. Y grubunun mantıksal iletisi 2
4. Bölüm 1/(son) mantıksal ileti 3/Y grubu
5. (Son) bölüm 2/(son) mantıksal ileti 3/grup Y
6. Z grubunun mantıksal iletisi 1
7. (Son) Z grubunun 2 numaralı mantıksal iletisi
8. İleti B (grupta değil)

Ancak fiziksel düzen tamamen farklı olabilir. Her bir gruptaki *ilk* öğenin fiziksel konumu, tüm grubun mantıksal konumunu belirler. Örneğin, Y ve Z grupları benzer zamanlarda geldiyse ve Z grubunun 2. iletisi aynı gruptaki 1. iletiyi aldıysa, fiziksel sıra Şekil Şekil 62 sayfa 744gibi görünecekti:



Şekil 62. Kuyruktaki fiziksel sıra

Bu iletiler kuyrukta aşağıdaki fiziksel sırada bulunur:

1. İleti A (bir grupta değil)
2. Y grubunun mantıksal iletisi 1
3. Z grubunun mantıksal iletisi 2
4. Y grubunun mantıksal iletisi 2
5. Bölüm 1/(son) mantıksal ileti 3/Y grubu
6. (Son) bölüm 2/(son) mantıksal ileti 3/grup Y
7. Z grubunun mantıksal iletisi 1
8. İleti B (grupta değil)

Not: IBM MQ for z/OSüzerinde, kuyruk GROUPID ile dizinlenirse, kuyruktaki iletilerin fiziksel sırası garanti edilmez.

İletiler alınırken, iletileri fiziksel sıra yerine mantıksal sırayla almak için MQGMO_LOGICAL_ORDER belirtebilirsiniz.

MQGMO_BROWSE_FIRST ve MQGMO_LOGICAL_ORDER ile bir MQGET çağrısı yayınlarsanız, MQGMO_BROWSE_NEXT ile sonraki MQGMO_LOGICAL_ORDER de belirtilmelidir.

Tersine, MQGMO_BROWSE_FIRST içeren MQGET MQGMO_LOGICAL_ORDER belirtmezse, MQGMO_BROWSE_NEXT içeren aşağıdaki MQGETS ' ler de belirtmemelidir.

Kuyruk yöneticisinin kuyruktaki iletilere göz atacak MQGET çağrıları için sakladığı grup ve bölüm bilgileri, kuyruk yöneticisinin kuyruktan ileti kaldran MQGET çağrıları için sakladığı grup ve bölüm bilgilerinden farklıdır. MQGMO_BROWSE_FIRST belirtilirse, kuyruk yöneticisi göz atmak için grup ve bölüm bilgilerini yoksayar ve kuyruğu, yürürlükteki grup ve yürürlükteki mantıksal ileti yokmuş gibi tarar.

Not: MQGMO_LOGICAL_ORDER belirtmeden bir ileti grubunun (ya da bir grupta olmayan mantıksal iletinin) *sonuna* göz atmak için MQGET çağrısı kullanmayın. Örneğin, gruptaki son ileti kuyruktaki ilk iletiden *önceyse* , grubun sonunun ötesine göz atmak için MQGMO_BROWSE_NEXT 'i kullanarak *MsgSeqNumber* değeri 1 olarak ayarlanmış MQMO_MATCH_MSG_SEQ_NUMBER değerini belirterek (sonraki grubun ilk iletisini bulmak için) gruptaki ilk iletiyi yeniden döndürür. Bu durum hemen oluşabilir ya da daha sonra (araya giren gruplar varsa) MQGET çağrıları olabilir.

Göz atmak için kuyruğu *iki kez* açarak sonsuz döngü olasılığından kaçının:

- Her gruptaki yalnızca ilk iletiye göz atmak için ilk tanıtıcıyı kullanın.
- Yalnızca belirli bir gruptaki iletilere göz atmak için ikinci tanıtıcıyı kullanın.
- Gruptaki iletilere göz atmadan önce, ikinci göz atma imlecini ilk göz atma imlecinin konumuna taşımak için MQMO_ * seçeneklerini kullanın.
- Bir grubun sonundan sonra MQGMO_BROWSE_NEXT göz atmayın.

Bununla ilgili daha fazla bilgi için MQGET, MQMDve MQI seçeneklerinin geçerliliğini denetleme kurallarıbaşlıklı konuya bakın.

Çoğu uygulama için, göz atarken büyük olasılıkla mantıksal ya da fiziksel sıralamayı seçersiniz. Ancak, bu kipler arasında geçiş yapmak istiyorsanız, MQGMO_LOGICAL_ORDER ile ilk kez göz attığınızda, mantıksal sıra içindeki konumunuzun oluşturulduğunu unutmayın.

Grup içindeki ilk öğe şu anda yoksa, içinde olduğunuz grup, mantıksal sıranın bir parçası olarak kabul edilmez.

Göz atma imleci bir grup içinde olduğunda, ilk ileti kaldırılmış olsa da, aynı grup içinde devam edebilir. Başlangıçta, ilk öğenin var olmadığı MQGMO_LOGICAL_ORDER öğesini kullanarak hiçbir zaman bir gruba geçmezsiniz.

MQPMO_LOGICAL_ORDER

MQPMO seçeneği, kuyruk yöneticisine uygulamanın iletileri gruplara ve mantıksal ileti bölümlerine nasıl yerleştirdiğini bildirir. Yalnızca MQPUT çağrısında belirtilebilir; MQPUT1 çağrısında geçerli değildir.

MQPMO_LOGICAL_ORDER belirtilirse, uygulamanın aşağıdaki öğeler için ardışık MQPUT çağrıları kullandığını gösterir:

1. Her mantıksal iletideki bölümleri, 0 'dan başlayarak, aralıksız olarak, kesim göreli konumunun artırılmasına göre sırgörüntüleyebilirsiniz?..,. ??? ...
2. Bölümleri sonraki mantıksal iletiye yerleştirmeden önce tüm bölümleri tek bir mantıksal iletiye koyun.
3. Mantıksal iletileri, her ileti grubundaki ileti sıra numarasının artma sırasına göre, 1 'den başlayarak, boşluk olmadan koyun. IBM MQ , ileti sıra numarasını otomatik olarak artırır.
4. Mantıksal iletileri sonraki ileti grubuna koymadan önce, tüm mantıksal iletileri bir ileti grubuna koyun.

Uygulama kuyruk yöneticisine iletileri mantıksal ileti gruplarına ve bölümlerine nasıl yerleştirdiğini anlattığı için, kuyruk yöneticisi bu bilgileri sakladığı ve güncellediği için uygulamanın her MQPUT çağrısına ilişkin grup ve bölüm bilgilerini tutması ve güncellemesi gerekmez. Özellikle, kuyruk yöneticisi bu alanları uygun değerlere ayarladığından, uygulamanın MQMD ' deki *GroupId*, *MsgSeqNumber*ve *Offset* alanlarını ayarlamasına gerek olmadığı anlamına gelir. Uygulama, iletilerin gruplara ait olduğunu ya da mantıksal ileti parçaları olduğunu belirtmek ve mantıksal iletinin bir grubundaki ya da son bölümündeki son iletiyi belirtmek için yalnızca MQMD ' deki *MsgFlags* alanını ayarlamalıdır.

Bir ileti grubu ya da mantıksal ileti başlatıldıktan sonra, sonraki MQPUT çağrıları MQMD ' deki *MsgFlags* içinde uygun MQMF_ * işaretlerini belirtmelidir. Uygulama, sonlandırılmamış bir ileti grubu olduğunda grupta olmayan bir iletiyi yerleştirmeyi ya da sonlandırılmamış bir mantıksal ileti olduğunda kesim olmayan bir iletiyi yerleştirmeyi denerse, çağrı MQRC_INCOMPLETE_GROUP ya da MQRC_INCOMPLETE_MSG neden koduyla başarısız olur. Ancak, kuyruk yöneticisi yürürlükteki ileti grubuna ya da yürürlükteki mantıksal iletiye ilişkin bilgileri alıkoyar ve uygulama, MQMF_LAST_MSG_IN_GROUP ya da MQMF_LAST_SEGMENT ' i uygun şekilde belirten bir ileti (büyük olasılıkla uygulama iletisi verileri olmadan) göndererek bunları sonlandırabilir. MQPUT çağrısı, grupta olmayan ya da olmayan iletiyi koymak için yeniden yayınlanmadan önce.

Şekil 62 sayfa 744 , geçerli seçenek ve işaret birleşimlerini ve kuyruk yöneticisinin her durumda kullandığı *GroupId*, *MsgSeqNumber*ve *Offset* alanlarının değerlerini gösterir. Tabloda gösterilmeyen seçenek ve işaret birleşimleri geçerli değil. Çizelgedeki kolonlar aşağıdaki anlamlara sahiptir; Evet ya da Hayır anlamına gelir:

GüNLüK ORDı
Çağrıda MQPMO_LOGICAL_ORDER seçeneğinin belirtilip belirtilmediğini belirler.

MIG
Çağrıda MQMF_MSG_IN_GROUP ya da MQMF_LAST_MSG_IN_GROUP seçeneğinin belirlenip belirlenmediğini belirler.

SİZ
Çağrıda MQMF_SEGMENT ya da MQMF_LAST_SEGMENT seçeneğinin belirtilip belirtilmediğini belirler.

SEG OK (TAMAM)
Çağrıda MQMF_SEGMENTATION_ALLOWED seçeneğinin belirtilip belirtilmediğini belirler.

Cur grp
Çağrıdan önce geçerli bir ileti grubunun bulunup bulunmadığını belirler.

Cur günlük iletisi
Çağrıdan önce geçerli bir mantıksal iletinin bulunup bulunmadığını belirler.

Diğer sütunlar
Kuyruk yöneticisinin kullandığı değerleri gösterir. Önceki, kuyruk tanıtıcısı için önceki iletide alan için kullanılan değeri belirtir.

Çizelge 116. Mantıksal ileti grupları ve bölümlerindeki iletilerle ilgili MQPUT seçenekleri

Belirlediğiniz seçenekler	Belirlediğiniz seçenekler	Belirlediğiniz seçenekler	Belirlediğiniz seçenekler	Çağrıdan önce grup ve günlük-ileti durumu	Çağrıdan önce grup ve günlük-ileti durumu	Kuyruk yöneticisinin kullandığı değerler	Kuyruk yöneticisinin kullandığı değerler	Kuyruk yöneticisinin kullandığı değerler
GüNLüK ORDı	MIG	SİZ	SEG OK (TAMAM)	Cur grp	Cur günlük iletisi	**GroupI**d	**MsgSeqNumber**	**Offset**
Evet	Hayır	Hayır	Hayır	Hayır	Hayır	MQGI_NONE	1	0
Evet	Hayır	Hayır	Evet	Hayır	Hayır	Yeni grup tanıtıcısı	1	0
Evet	Hayır	Evet	Herhangi biri	Hayır	Hayır	Yeni grup tanıtıcısı	1	0

Çizelge 116. Mantıksal ileti grupları ve bölümlerindeki iletilerle ilgili MQPUT seçenekleri (devamı var)

Belirlediğiniz seçenekler	Belirlediğiniz seçenekler	Belirlediğiniz seçenekler	Belirlediğiniz seçenekler	Çağrıdan önce grup ve günlük-ileti durumu	Çağrıdan önce grup ve günlük-ileti durumu	Kuyruk yöneticisinin kullandığı değerler	Kuyruk yöneticisinin kullandığı değerler	Kuyruk yöneticisinin kullandığı değerler
Evet	Hayır	Evet	Herhangi biri	Hayır	Evet	Önceki grup tanıtıcısı	1	Önceki göreli konum + önceki kesim uzunluğu
Evet	Evet	Herhangi biri	Herhangi biri	Hayır	Hayır	Yeni grup tanıtıcısı	1	0
Evet	Evet	Herhangi biri	Herhangi biri	Evet	Hayır	Önceki grup tanıtıcısı	Önceki sıra numarası + 1	0
Evet	Evet	Evet	Herhangi biri	Evet	Evet	Önceki grup tanıtıcısı	Önceki sıra numarası	Önceki göreli konum + önceki kesim uzunluğu
Hayır	Hayır	Hayır	Hayır	Herhangi biri	Herhangi biri	MQGI_NONE	1	0
Hayır	Hayır	Hayır	Evet	Herhangi biri	Herhangi biri	MQGI_NONE ise yeni grup tanıtıcısı, alanda else değeri	1	0
Hayır	Hayır	Evet	Herhangi biri	Herhangi biri	Herhangi biri	MQGI_NONE ise yeni grup tanıtıcısı, alanda else değeri	1	Alandaki değer
Hayır	Evet	Hayır	Herhangi biri	Herhangi biri	Herhangi biri	MQGI_NONE ise yeni grup tanıtıcısı, alanda else değeri	Alandaki değer	0
Hayır	Evet	Evet	Herhangi biri	Herhangi biri	Herhangi biri	MQGI_NONE ise yeni grup tanıtıcısı, alanda else değeri	Alandaki değer	Alandaki değer

Not:

- MQPMO_LOGICAL_ORDER, MQPUT1 çağrısında geçerli değil.
- *MsgId* alanı için kuyruk yöneticisi, MQPMO_NEW_MSG_ID ya da MQMI_NONE belirtilirse yeni bir ileti tanıtıcısı oluşturur; tersi durumda, alandaki değeri kullanır.
- *CorrelId* alanı için, MQPMO_NEW_CORREL_ID belirtilirse kuyruk yöneticisi yeni bir ilinti tanıtıcısı oluşturur; tersi durumda, alandaki değeri kullanır.

MQPMO_LOGICAL_ORDER belirtilirse, kuyruk yöneticisi bir gruptaki ve mantıksal iletideki bölümlerdeki tüm iletilerin MQMD ' deki *Persistence* alanında aynı değerle konmasını, yani tümünün kalıcı olmasını ya da kalıcı olmamasını gerektirir. Bu koşul karşılanmazsa, MQPUT çağrısı MQRC_INPERSIST_PERSISTENCE neden koduyla başarısız olur.

MQPMO_LOGICAL_ORDER seçeneği, iş birimlerini aşağıdaki gibi etkiler:

- Bir grup ya da mantıksal iletideki ilk fiziksel ileti bir iş birimine yerleştirilirse, aynı kuyruk tanıtıcısı kullanılırsa, gruptaki ya da mantıksal iletideki diğer tüm fiziksel iletilerin bir iş birimine yerleştirilmesi gerekir. Ancak, bu iletilerin aynı iş birimine yerleştirilmesi gerekmez; bu, birçok fiziksel iletiden

oluşan bir ileti grubunun ya da mantıksal iletinin, kuyruk tanıtıcısı için iki ya da daha fazla ardışık iş birimine bölünmesine izin verir.

- Bir grup ya da mantıksal iletideki ilk fiziksel ileti bir iş birimine yerleştirilmezse, aynı kuyruk tanıtıcısı kullanılırsa, gruptaki ya da mantıksal iletideki diğer fiziksel iletilerin hiçbiri bir iş birimine yerleştirilemez.

Bu koşullar karşılanmazsa, MQPUT çağrısı başarısız olur; neden kodu MQRC_INTUTARLI t_uow.

MQPMO_LOGICAL_ORDER belirtildiğinde, MQPUT çağrısında belirtilen MQMD, MQMD_VERSION_2' den küçük olmamalıdır. Bu koşul karşılanmazsa, çağrı MQRC_YANLIŞ _md_version neden koduyla başarısız olur.

MQPMO_LOGICAL_ORDER belirtilmezse, gruplar ve mantıksal ileti bölümlerindeki iletiler herhangi bir sıraya konabilir ve tüm ileti gruplarını ya da tam mantıksal iletileri koymak gerekmez. *GroupId*, *MsgSeqNumber*, *Offset*ve *MsgFlags* alanlarının uygun değerlere sahip olduğundan emin olmak uygulamanın sorumluluğundadır.

Bir sistem arızası oluştuktan sonra ortada bir ileti grubunu ya da mantıksal iletiyi yeniden başlatmak için bu tekniği kullanın. Sistem yeniden başlatıldığında, uygulama *GroupId*, *MsgSeqNumber*, *Offset*, *MsgFlags*ve *Persistence* alanlarını uygun değerlere ayarlayabilir ve MQPMO_syncpoint ya da MQPMO_NO_SYNCPOINT ile MQPUT çağrıyı gerektiği gibi, ancak MQPMO_LOGICAL_ORDER belirtilmeden verebilir. Bu çağrı başarılı olursa, kuyruk yöneticisi grup ve bölüm bilgilerini korur ve bu kuyruk tanıtıcısını kullanan sonraki MQPUT çağrıları MQPMO_LOGICAL_ORDER ' i normal olarak belirtebilir.

Kuyruk yöneticisinin MQPUT çağrısı için sakladığı grup ve bölüm bilgileri, MQGET çağrısı için sakladığı grup ve bölüm bilgilerinden ayrıdır.

Belirli bir kuyruk tanıtıcısı için, uygulama MQPMO_LOGICAL_ORDER belirten MQPUT çağrılarını MQPUT çağrılarıyla karıştırabilir, ancak aşağıdaki noktaları göz önünde bulundurun:

- MQPMO_LOGICAL_ORDER belirtilmezse, her başarılı MQPUT çağrısı kuyruk yöneticisinin kuyruk tanıtıcısı için grup ve bölüm bilgilerini uygulama tarafından belirtilen değerlere ayarlamasına ve kuyruk tanıtıcısı için kuyruk yöneticisi tarafından tutulan grup ve bölüm bilgilerini değiştirmesine neden olur.
- MQPMO_LOGICAL_ORDER belirtilmezse, yürürlükteki ileti grubu ya da mantıksal ileti varsa çağrı başarısız olmaz; çağrı bir MQCC_WARNING tamamlanma koduyla başarılı olabilir. Çizelge 117 sayfa 748 , ortaya çıkabilecek çeşitli durumları gösterir. Bu durumlarda, tamamlama kodu MQCC_OK değilse, neden kodu aşağıdakilerden biridir (uygun olduğu şekilde):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_INPERSIST_PERSISTENCE
  - MQRC_TUTARSIZ_UOW

  Not: Kuyruk yöneticisi, MQPUT1 çağrıyla ilgili grup ve bölüm bilgilerini denetlemez.

Çizelge 117. MQPUT ya da MQCLOSE çağrısı grup ve bölüm bilgileriyle tutarlı olmadığında sonuç

Geçerli arama:	**Önceki çağrı MQPMO_LOGICAL_ORDER ile MQPUT idi**	**Önceki çağrı MQPMO_LOGICAL_ORDER olmadan MQPUT idi**
MQPMO_LOGICAL_ORDER ile MQPUT	MQCC_FAILED	MQCC_FAILED
MQPMO_LOGICAL_ORDER olmadan MQPUT	MQCC_UYARıSı	MQCC_OK
Sonlandırılmamış grup ya da mantıksal ileti ile MQCLOSE	MQCC_UYARıSı	MQCC_OK

İletileri ve bölümleri mantıksal sıraya koyan uygulamalar için, MQPMO_LOGICAL_ORDER belirtin; bu en basit seçenektir. Kuyruk yöneticisi bu bilgileri yönettiği için, bu seçenek grup ve bölüm bilgilerini yönetme gereksinimini ortadan kaldırır. Ancak, her MQPUT ya da MQPUT1 çağrısından önce, özelleştirilmiş uygulamaların MQPMO_LOGICAL_ORDER seçeneği tarafından sağlanandan daha fazla denetime gereksinimi olabilir; bu seçenek belirtilmezse, MQMD ' deki *GroupId*, *MsgSeqNumber*, *Offset*ve *MsgFlags* alanlarının doğru ayarlandığından emin olmanız gerekir.

Örneğin, aldığı fiziksel iletileri iletmek isteyen bir uygulama, bu iletilerin gruplar halinde mi, yoksa mantıksal ileti bölümlerinde mi olduğuna bakılmaksızın, iki nedenden ötürü MQPMO_LOGICAL_ORDER belirtmemelidir:

- İletiler alınıp sıraya konursa, MQPMO_LOGICAL_ORDER belirtilirse, iletilere yeni bir grup tanıtıcısı atar; bu, iletilerin yaratıcısının ileti grubundan kaynaklanan yanıt ya da rapor iletilerini ilintilendirmesini zorlaştırabilir ya da olanaksız kılabilir.
- Gönderme ve alma kuyruk yöneticileri arasında birden çok yolu olan karmaşık bir ağda, fiziksel iletiler sıradışı gelebilir. MQGET çağrısında MQPMO_LOGICAL_ORDER ve MQGMO_LOGICAL_ORDER belirtilmediğinde, iletme uygulaması her fiziksel iletiyi gelir gelmez, mantıksal sırada bir sonraki iletiyi beklemeden alabilir ve iletebilir.

Mantıksal ileti gruplarındaki ya da bölümlerindeki iletiler için rapor iletileri oluşturan uygulamalar, rapor iletisini koyarken MQPMO_LOGICAL_ORDER belirtmemelidir.

MQPMO_LOGICAL_ORDER, diğer MQPMO_ * seçeneklerinden herhangi biriyle belirtilebilir.

Mantıksal Olarak Sıralı Grupları Kümelenmiş Kuyruğa Koyma (MQOO_BIND_ON_GROUP)

MQOO_BIND_ON_OPEN seçeneği, bu uygulamadaki ve dolayısıyla tüm gruplardaki tüm iletilerin tek bir yönetim ortamına yöneltilmesini sağlar. Bu, uygulama trafiğinin bir küme kuyruğunun birden çok eşgörünümünde dengelenmemiş olması nedeniyle olumsuz bir durumdur. İleti gruplarını sağlam tutarken iş yükü dengelemeyi etkinleştirmek için aşağıdaki seçenekleri ayarlamanız gerekir:

- MQPUT çağrısı MQPMO_LOGICAL_ORDER öğesini belirtmelidir
- MQOPEN çağrısı aşağıdaki iki seçenekten birini belirtmelidir:
  - MQOO_BIND_ON_GROUP
  - MQOO_BIND_AS_Q_DEF ve kuyruk tanımlaması DEFBIND (GROUP) belirtmelidir

İş yükü dengeleme, kuyruğun MQCLOSE ve MQOPEN gerektirmeden iletilerin *grupları arasında* yönlendirilir. *Gruplar arasında* , MQMD (v2) ya da MQMDE 'de MQMF_MSG_IN_GROUP' un ayarlandığı ve devam eden kısmen tamamlanmış bir grup olmadığı anlamına gelir. Bir grup devam ederken, nesne tanıtıcısında çözülen kuyruk yöneticisi ve kuyruk adı yeniden kullanılır.

Önceki ileti MQPMO_LOGICAL_ORDER ve/ya da MQMF_MSG_IN_GROUP ise, ancak yürürlükteki ileti grubun bir parçası değilse, PUT çağrısı MQRC_INCOMPLETE_GROUPile başarısız olur.

Tek bir MQPUT MQPMO_LOGICAL_ORDER belirtmezse ve yürürlükteki grup etkin değilse, bu ileti için iş yükü dengeleme işlemi yönlendirilir (MQOPEN çağrısı MQOO_BIND_NOT_FIXED belirtmiş gibi).

MQOO_BIND_ON_GROUP kullanılarak bir hedefe bağlanan iletiler için yeniden tahsis gerçekleşmez. Yeniden ayırmayla ilgili daha fazla bilgi için bkz. “İleti grupları” sayfa 42.

Mantıksal iletileri gruplama

Bir gruptaki mantıksal iletileri kullanmak için iki ana neden vardır:

- İletileri belirli bir sırayla işlemeniz gerekebilir.
- Bir gruptaki her iletiyi ilgili bir şekilde işlemeniz gerekebilir.

Her iki durumda da, aynı alma uygulaması örneğine sahip tüm grubu alın.

Örneğin, grubun dört mantıksal iletiden oluştuğunu varsayın. Koyma uygulaması şöyle görünüyor:

```
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
```

Alma uygulaması, gruptaki ilk ileti için MQGMO_ALL_MSGS_KULLANILABILIR seçeneğini belirtiyor. Bu, grup içindeki tüm iletiler gelene kadar işlemenin başlamamasını sağlar. MQGMO_ALL_MSGS_KULLANILABILIR seçeneği, grup içindeki sonraki iletiler için yoksayılır.

Grubun ilk mantıksal iletisi alındığında, grubun geri kalan mantıksal iletilerinin sırayla alındığından emin olmak için MQGMO_LOGICAL_ORDER işlevini kullanabilirsiniz.

Yani, başvuru formu şöyle görünüyor:

```
/* 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
```

İletileri gruplamaya ilişkin diğer örnekler için bkz. “Mantıksal iletilerin uygulama bölümlemesi” sayfa 762 ve “İş birimlerini kapsayan bir grubu koyma ve alma” sayfa 750.

Uyarı: Bir konuya ileti göndermek (ya da bir konu diğer adına ileti yerleştirmek) için yayınlama/abone olma kullanılırken, ileti gruplamaya ve bölümlemeye izin verilmez.

Abonelikler yayın etkinliğinden bağımsız olarak yaratılabileceği ve kaldırılabileceği için, bir abonenin tam bir ileti grubu ya da iletinin tüm bölümlerini alacağı garanti edilemez; bkz. RC2417: MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Bir uygulamanın bir grup iletinin küme kuyrukları için aynı hedef yönetim ortamına ayrılmasını istemesine izin verme hakkında bilgi için bkz. DefBind.

İş birimlerini kapsayan bir grubu koyma ve alma

Önceki durumda, iletiler ya da kesimler düğümden ayrılmaya başlayamaz (hedefi uzaksa) ya da tüm grup konup iş birimi kesinleştirilinceye kadar alınmaya başlayamaz. Tüm grubun yerleştirilmesi uzun zaman alıyorsa ya da düğümde kuyruk alanı sınırlıysa, istediğiniz bu olmayabilir. Bunu aşmak için, grubu birkaç iş birimine koyun.

Grup birden çok iş birimine yerleştirilirse, koyma uygulaması başarısız olduğunda bile grubun bir kısmı kesinleştirilebilir. Bu nedenle uygulama, tamamlanmamış bir grubu sürdürmek için yeniden başlatmadan sonra kullanabileceği her iş birimiyle kesinleştirilen durum bilgilerini kaydetmelidir. Bu bilgilerin kaydedileceği en basit yer STATUS kuyruğudur. Tam bir grup başarıyla yerleştirildiyse, STATUS kuyruğu boştur.

Bölümleme söz konusuysa, mantık benzerdir. Bu durumda **StatusInfo** , *Offset*içermelidir.

Aşağıda, grubu birkaç iş birimine yerleştirmenin bir örneği verilmiştir:

```
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
```

Tüm iş birimleri kesinleştirildiyse, tüm grup başarıyla konmuştur ve STATUS kuyruğu boştur. Değilse, grubun durum bilgileriyle belirtilen noktada sürdürülmesi gerekir. MQPMO_LOGICAL_ORDER ilk koyma için kullanılamaz, ancak bundan sonra kullanılabilir.

Yeniden başlatma işlemi şöyle görünür:

```
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
```

Alma uygulamasından, tüm grup gelmeden önce bir gruptaki iletileri işlemeye başlamak isteyebilirsiniz. Bu, grup içindeki iletilerde yanıt sürelerini artırır ve tüm grup için depolama gerekmediği anlamına da gelir. Avantajları gerçekleştirmek için, her ileti grubu için birkaç iş birimi kullanın. Kurtarma nedenleriyle, bir iş birimi içindeki her iletiyi almanız gerekir.

İlgili koyma uygulamasında olduğu gibi, her iş birimi kesinleştirilirken durum bilgilerinin otomatik olarak bir yere kaydedilmesini gerektirir. Yine, bu bilgilerin kaydedileceği en basit yer bir STATUS kuyruğudur. Tam bir grup başarıyla işlendiyse, STATUS kuyruğu boştur.

Not: Ara iş birimleri için, STATUS kuyruğundaki MQGET çağrılarını önlemek için, durum kuyruğundaki her MQPUT ' nin bir iletinin parçası olduğunu (MQMF_SEGMENT işaretini ayarlayarak) belirterek, her iş birimi için yeni bir ileti koymayı önleyebilirsiniz. Son iş biriminde, son bölüm MQMF_LAST_SEGMENT belirtilerek durum kuyruğuna yerleştirilir ve durum bilgileri MQGMO_COMPLETE_MSG belirten bir MQGET ile temizlenir.

Yeniden başlatma işlemi sırasında, olası bir durum iletisi almak için tek bir MQGET kullanmak yerine, son bölüme ulaşıncaya kadar (başka bir bölüm döndürülünceye kadar) MQGMO_LOGICAL_ORDER ile durum kuyruğuna göz atın. Yeniden başlatmadan sonraki ilk iş biriminde, durum bölümünü koyarken göreli konumu da belirtik olarak belirtin.

Aşağıdaki örnekte, ileti bölümlenmiş olsun ya da olmasın, uygulamanın arabelleğinin iletinin tamamını tutacak kadar her zaman büyük olduğunu varsayarak, yalnızca bir grup içindeki iletileri göz önünde bulunduruyoruz. Bu nedenle, her MQGET için MQGMO_COMPLETE_MSG belirtildi. Bölümleme söz konusu olduğunda aynı ilkeler geçerlidir (bu durumda, StatusInfo *Offset*içermelidir).

Basitlik için, tek bir UOW içinde en fazla 4 iletinin alındığını varsayalım:

```
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
```

Tüm iş birimleri kesinleştirildiyse, tüm grup başarıyla alındı ve STATUS kuyruğu boş. Değilse, grubun durum bilgileriyle belirtilen noktada sürdürülmesi gerekir. MQGMO_LOGICAL_ORDER ilk alma için kullanılamaz, ancak bundan sonra kullanılabilir.

Yeniden başlatma işlemi şöyle görünür:

```
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
```

Belirli bir iletiyi alma

Belirli bir iletiyi kuyruktan almanın birkaç yolu vardır. Bunlar: MsgId ve CorrelIdöğelerini seçerek, GroupId, MsgSeqNumber and Offset öğelerini seçerek ve MsgTokenöğesini seçerek. Kuyruğu açarken bir seçim dizgisi de kullanabilirsiniz.

Belirli bir iletiyi kuyruktan almak için MQMD yapısının `MsgId` ve `CorrelId` alanlarını kullanın. Ancak, uygulamalar bu alanları belirtik olarak ayarlayabilir; böylece, belirttiğiniz değerler benzersiz bir iletiyi tanımlamayabilir. Çizelge 118 sayfa 753 , bu alanların olası ayarları için hangi iletinin alınacağını gösterir. MQGET çağrısının **`GetMsgOpts`** değiştirgesinde MQGMO_MSG_UNDER_CURSOR belirtirseniz, girişte bu alanlar yoksayılır.

Çizelge 118. İleti ve ilinti tanıtıcılarının kullanılması

Geri almak için ...	**`MsgId`**	**`CorrelId`**
Kuyruktaki ilk ileti	`MQMI_NONE`	`MQCI_NONE`
`MsgId` ile eşleşen ilk ileti	Sıfır olmayan	`MQCI_NONE`
`CorrelId` ile eşleşen ilk ileti	`MQMI_NONE`	Sıfır olmayan
Hem *`MsgId`* hem de *`CorrelId`* ile eşleşen ilk ileti	Sıfır olmayan	Sıfır olmayan

Her durumda *first* , seçim ölçütlerini karşılayan ilk ileti anlamına gelir (MQGMO_BROWSE_NEXT belirtilmedikçe, seçim ölçütlerini karşılayan sıradaki *sonraki* iletisi anlamına geliyorsa).

Dönüşte, MQGET çağrısı *`MsgId`* ve *`CorrelId`* alanlarını döndürülen iletinin ileti ve ilinti tanıtıcılarına (varsa) ayarlar.

MQMD yapısının *`Version`* alanını 2 olarak ayarlarsanız, *`GroupId`*, *`MsgSeqNumber`*ve *`Offset`* alanlarını kullanabilirsiniz. Çizelge 119 sayfa 753 , bu alanların olası ayarları için hangi iletinin alınacağını gösterir.

Çizelge 119. Grup tanıtıcısının kullanılması

Geri almak için ...	**Eşleştirme seçenekleri**
Kuyruktaki ilk ileti	MQMO_NONE
`MsgId` ile eşleşen ilk ileti	MQMO_MATCH_MSG_ID
`CorrelId` ile eşleşen ilk ileti	MQMO_MATCH_CORREL_ID
`GroupId` ile eşleşen ilk ileti	MQMO_MATCH_GROUP_ID
`MsgSeqNumber` ile eşleşen ilk ileti	MQMO_MATCH_MSG_SEQ_NUMBER
`MsgToken` ile eşleşen ilk ileti	MQMO_MATCH_MSG_TOKEN
`Offset` ile eşleşen ilk ileti	MQMO_MATCH_OFFSET

Notlar:

1. MQMO_MATCH_XXX, MQMD yapısındaki *XXX* alanının eşleştirilecek değere ayarlandığını belirtir.
2. MQMO işaretleri birlikte kullanılabilir. Örneğin, MQMO_MATCH_GROUP_ID, MQMO_MATCH_MSG_SEQ_NUMBER ve MQMO_MATCH_OFFSET, `GroupId`, `MsgSeqNumber`ve `Offset` alanlarıyla tanıtılan kesimi vermek için birlikte kullanılabilir.
3. MQGMO_LOGICAL_ORDER belirtirseniz, almaya çalıştığınız ileti, kuyruk tanıtıcısı için denetlenen durum bilgilerine bağlı olduğundan etkilenir. Bu konuda bilgi için bkz. “Mantıksal ve fiziksel sıralama” sayfa 742 ve Seçenekler.

MQGET çağrısı genellikle bir kuyruktan ilk iletiyi alır. MQGET çağrısına ilişkin belirli bir iletiyi belirtirseniz, kuyruk yöneticisi o iletiyi buluncaya kadar kuyrukta arama yapmak zorundadır. Bu, uygulamanızın performansını etkileyebilir.

MQGMO yapısının Sürüm 2 ya da üstünü kullanıyorsanız ve MQMO_MATCH_MSG_ID ya da MQMO_MATCH_CORREL_ID işaretlerini belirtmezseniz, MQGETs arasındaki `MsgId` ya da `CorrelId` alanlarını ilk durumuna getirmeniz gerekmez.

z/OS IBM MQ for z/OSüzerinde, kuyruktaki MQGET işlemlerinin hızını artırmak için `IndexType` kuyruk özniteliği kullanılabilir. Daha fazla bilgi için bkz "Dizin tipi" sayfa 758.

Bir kuyruktan belirli bir iletiyi, MQGMO yapısında MsgToken ve MatchOption MQMO_MATCH_MSG_TOKEN öğelerini belirterek alabilirsiniz. MsgToken , ilk olarak o iletiyi kuyruğa koyan MQPUT çağrısı ya da kuyruk yöneticisi yeniden başlatılmadıkça önceki MQGET işlemleri tarafından döndürülür ve değişmez olarak kalır.

Kuyruktaki iletilerin yalnızca bir altkümesiyle ilgileniyorsanız, MQOPEN ya da MQSUB çağrısıyla bir seçim dizgisi kullanarak işlemek istediğiniz iletileri belirtebilirsiniz. MQGET daha sonra, o seçim dizgisini karşılayan sonraki iletiyi alır. Seçim dizgileri hakkında daha fazla bilgi için bkz. "Seçiciler" sayfa 29.

Kalıcı olmayan iletilerin performansını artırma

Bir istemci sunucudan gelen bir iletiyi gerektiriyorsa, sunucuya bir istek gönderir. Kullandığı her ileti için ayrı bir istek gönderir. Bu istek iletilerini göndermek zorunda kalmaktan kaçınarak kalıcı olmayan iletileri kullanan bir istemcinin performansını artırmak için bir istemci *önden okuma*özelliğini kullanacak şekilde yapılandırılabilir. Önden okuma, iletilerin bir uygulama tarafından istenmeden istemciye gönderilmesini sağlar.

Önden okuma etkinleştirildiğinde, istemcideki *önden okuma arabelleği*adı verilen bir arabelleğe iletiler gönderilir. İstemcinin, önden okuma özelliği etkinleştirilmiş olarak açmış olduğu her kuyruk için bir önden okuma arabelleği vardır. Önden okuma arabelleğindeki iletiler kalıcı olarak saklanmaz. İstemci, belirli aralıklarla sunucuyu tükettiği veri miktarına ilişkin bilgilerle günceller.

MQOO_READ_ÖNCEDEN ile MQOPEN ' i çağırdığınızda, IBM MQ istemcisi belirli koşullar karşılandığında yalnızca önden okuma seçeneğini etkinleştirir. Bu koşullar şunlardır:

- İstemci uygulaması derlenmeli ve iş parçacıklı IBM MQ MQI istemcisi kitaplıklarına bağlanmalıdır.
- İstemci kanalı TCP/IP protokolünü kullanıyor olmalıdır
- Kanal, istemci ve sunucu kanalı tanımlarında sıfır dışında bir SharingConversations (SHARECNV) ayarı içermelidir.

Önden okuma özelliğinin kullanılması, bir istemci uygulamasındaki kalıcı olmayan iletileri kullanırken başarımı artırabilir. Bu performans iyileştirmesi hem MQI hem de JMS uygulamaları tarafından kullanılabilir. MQGET ya da zamanuyumsuz tüketim kullanan istemci uygulamaları, kalıcı olmayan iletileri kullanırken performans iyileştirmelerinden yararlanacaktır.

Önden okuma etkinleştirildiğinde MQGET çağrıları arasında tutarlı olması için bazı seçenekler desteklenmediğinden, tüm istemci uygulaması tasarımları önden okuma özelliğini kullanmaya uygun değildir. Bir istemci MQGET çağrıları arasında seçim ölçütlerini değiştirirse, önden okuma arabelleğinde saklanan iletiler istemcinin önden okuma arabelleğinde kalır.

Önceki seçim ölçütlerine sahip bir birikmiş ileti birikimi artık gerekli değilse, istemcide bu iletileri istemciden otomatik olarak temizlemek için yapılandırılabilir bir temizleme aralığı ayarlanabilir. Temizleme aralığı, istemci tarafından belirlenen okuma önden ayarlama seçeneklerinden biridir. Bu seçeneklerin gereksinimlerinizi karşılayacak şekilde ayarlanması mümkündür.

Bir istemci uygulaması yeniden başlatılırsa, önden okuma arabelleğindeki iletiler kaybolabilir. Tersi durumda, önden okuma arabelleğine taşınan bir ileti temeldeki kuyruktan silinebilir; bu, iletinin arabellekten kaldırılmasına neden olmaz; dolayısıyla, önden okuma özelliğini kullanan bir MQGET çağrısı artık var olmayan bir ileti döndürebilir.

Önden okuma yalnızca istemci bağ tanımları için gerçekleştirilir. Öznitelik, diğer tüm bağ tanımları için yoksayılır.

Önden okuma tetikleme üzerinde bir etki yaratmaz. Bir ileti istemci tarafından önden okunduğunda tetikleyici iletisi oluşturulmaz. Önden okuma, etkinleştirildiğinde muhasebe ve istatistik bilgileri oluşturmaz.

Yayınlama aboneliği ileti sistemiyle önden okuma özelliğini kullanma

Abone olan bir uygulama, yayınların gönderileceği hedef kuyruğu belirttiğinde, varsayılan önden okuma değeri olarak belirtilen kuyruğun DEFREADA değeri kullanılır.

IBM MQ ' in yayınların gönderildiği hedefi yönettiği abone olan bir uygulama istediğinde, yönetilen bir kuyruk, önceden tanımlanmış bir model kuyruğuna dayalı olarak dinamik bir kuyruk olarak yaratılır. Varsayılan önden okuma değeri olarak kullanılan model kuyruğunun DEFREADA değeri. Varsayılan model kuyrukları SYSTEM.DURABLE.PUBLICATIONS.MODEL ya da SYSTEM.NONDURABLE.PUBLICATIONS.MODEL , bu konu ya da üst konu için bir model kuyruğu tanımlanmadıkça kullanılır.

İlgili kavramlar

"AIX üzerinde kalıcı olmayan iletiler için performansı ayarlama" sayfa 757
AIX V5.3 ya da sonraki bir yayın düzeyini kullanıyorsanız, ayarlama değiştirgesini kalıcı olmayan iletiler için tam başarımı kullanacak şekilde ayarlayın.

İlgili görevler

"Önden okumayı etkinleştirme ve devre dışı bırakma" sayfa 756
Önden okuma varsayılan olarak devre dışıdır. Kuyrukta ya da uygulama düzeyinde önden okumayı etkinleştirebilirsiniz.

İlgili başvurular

"MQGET seçenekleri ve önden okuma" sayfa 755
Önden okuma etkinleştirildiğinde tüm MQGET seçenekleri desteklenmez; bazı seçeneklerin MQGET çağrıları arasında tutarlı olması gerekir.

MQGET seçenekleri ve önden okuma
Önden okuma etkinleştirildiğinde tüm MQGET seçenekleri desteklenmez; bazı seçeneklerin MQGET çağrıları arasında tutarlı olması gerekir.

MQOO_READ_ÖNCEDEN ile MQOPEN ' i çağırdığınızda, IBM MQ istemcisi belirli koşullar karşılandığında yalnızca önden okuma seçeneğini etkinleştirir. Bu koşullar şunlardır:

- İstemci uygulaması derlenmeli ve iş parçacıklı IBM MQ MQI istemcisi kitaplıklarına bağlanmalıdır.
- İstemci kanalı TCP/IP protokolünü kullanıyor olmalıdır
- Kanal, istemci ve sunucu kanalı tanımlarında sıfır dışında bir SharingConversations (SHARECNV) ayarı içermelidir.

Aşağıdaki çizelge, önden okuma ile kullanılmak üzere hangi seçeneklerin desteklendiğini ve bunların MQGET çağrıları arasında değiştirilip değiştirilemeyeceğini gösterir.

Çizelge 120. MQGET seçenekleri ve önden okuma

MQGET değerleri ve seçenekleri	**Önden okuma etkinleştirildiğinde izin verilir ve MQGET çağrıları arasında değiştirilebilir [5]**	**Önden okuma etkinleştirildiğinde izin verilir, ancak MQGET çağrıları arasında değiştirilemez [1]**	**Önden okuma etkinleştirildiğinde izin verilmeyen MQGET Seçenekleri [2]**
MQGET MQMD değerleri	MsgId [3] CorrelId [3]	Kodlama CodedCharSetId	

Çizelge 120. MQGET seçenekleri ve önden okuma (devamı var)			
MQGET değerleri ve seçenekleri	**Önden okuma etkinleştirildiğinde izin verilir ve MQGET çağrıları arasında değiştirilebilir [5]**	**Önden okuma etkinleştirildiğinde izin verilir, ancak MQGET çağrıları arasında değiştirilemez [1]**	**Önden okuma etkinleştirildiğinde izin verilmeyen MQGET Seçenekleri [2]**
MQGET MQGMO Seçenekleri	• MQGMO_NO_WAIT • MQGMO_BROWSE_MESSAGE _UNDER_CURSOR • MQGMO_BROWSE_FIRST • MQGMO_BROWSE_NEXT • MQGMO_FAIL_IF_QUIESCING	• MQGMO_SYNCPOINT_EĞER _PERSISTENT • MQGMO_NO_SYNCPOINT • MQGMO_ACCEPT_TRUNCATED _MSG • MQGMO_CONVERT	• MQGMO_SET_SIGNAL • MQGMO_SYNCPOINT • MQGMO_MARK_SKIP _BACKOUT • MQGMO_MSG_UNDER _CURSOR 4 • MQGMO_KİLİK • MQGMO_UNLOCK • MQGMO_LOGICAL_ORDER • MQGMO_COMPLETE_MSG • MQGMO_ALL_MSGS_VAR • MQGMO_ALL_SEGMENTS_ KULLANıLABILIR

Notlar:

1. MQGET çağrıları arasında bu seçenekler değiştirilirse, bir MQRC_OPTIONS_CHANGED neden kodu döndürülür.
2. Bu seçenekler ilk MQGET çağrısında belirtilirse, önden okuma geçersiz kılınır. Sonraki bir MQGET çağrısında bu seçenekler belirtilirse, MQRC_OPTIONS_ERROR neden kodu döndürülür.
3. Bir istemci uygulaması MQGET çağrıları arasında MsgId ve CorrelId değerlerini değişirse, önceki değerleri olan iletiler istemciye önceden gönderilmiş olabilir ve tüketilinceye (ya da otomatik olarak temizleninceye) kadar istemci önden okuma arabelleğinde kalır.
4. MQGMO_MSG_UNDER_CURSOR, önden okuma ile gerçekleştirilemez. Kuyruk açılırken hem MQOO_BROWSE hem de MQOO_INPUT_SHARED ya da MQOO_INPUT_EXCLUSIVE seçeneklerinden biri belirtilirse önden okuma geçersiz kılınır.
5. Önden okuma etkinleştirildiğinde, iletilere göz atılacağını ya da kuyruktan alınacağını ilk MQGET belirler. İstemci uygulaması değiştirilmiş seçeneklerle MQGET kullanıyorsa (örneğin, ilk alma sonrasında göz atma girişiminde bulunma ya da ilk göz atma sonrasında alma girişiminde bulunma gibi), bir MQRC_OPTIONS_CHANGED neden kodu döndürülür.

Bir istemci MQGET çağrıları arasındaki seçim ölçütlerini değiştirirse, önden okuma arabelleğinde saklanan ve ilk seçim ölçütleriyle eşleşen iletiler istemci uygulaması tarafından tüketilmez ve istemci önden okuma arabelleğinde kalır. İstemcinin önden okuma arabelleğinin çok sayıda sarf edilmiş ileti içerdiği durumlarda, önden okuma ile ilişkili yararlar kaybolur ve tüketilen her ileti için sunucuya ayrı bir istek gerekir. Önden okuma işlevinin verimli bir şekilde kullanılıp kullanılmadığını belirlemek için READA bağlantı durumu parametresini kullanabilirsiniz.

İlk MQGET çağrısında belirtilen uyumsuz seçenekler nedeniyle bir uygulama tarafından istendiğinde önden okuma işlemi engellenebilir. Bu durumda bağlantı durumu, önden okunma durumunu engelleniyor olarak gösterir.

MQGET ile ilgili bu kısıtlamalar nedeniyle, bir istemci uygulaması tasarımının önden okumaya uygun olmadığına karar verirseniz, MQOPEN seçeneği MQOO_READ_AHEAD_NO değerini belirtin. Diğer bir seçenek olarak, açılmakta olan kuyruğun varsayılan önden okuma değerini NO ya da DISABLED olarak ayarlayın.

Önden okumayı etkinleştirme ve devre dışı bırakma
Önden okuma varsayılan olarak devre dışıdır. Kuyrukta ya da uygulama düzeyinde önden okumayı etkinleştirebilirsiniz.

Bu görev hakkında

MQOO_READ_ÖNCEDEN ile MQOPEN ' i çağırdığınızda, IBM MQ istemcisi belirli koşullar karşılandığında yalnızca önden okuma seçeneğini etkinleştirir. Bu koşullar şunlardır:

- İstemci uygulaması derlenmeli ve iş parçacıklı IBM MQ MQI istemcisi kitaplıklarına bağlanmalıdır.
- İstemci kanalı TCP/IP protokolünü kullanıyor olmalıdır
- Kanal, istemci ve sunucu kanalı tanımlarında sıfır dışında bir SharingConversations (SHARECNV) ayarı içermelidir.

Önden okumayı etkinleştirmek için:

- Kuyruk düzeyinde önden okuma özelliğini yapılandırmak için kuyruk özniteliğini DEFREADA için YES değerini belirleyin.
- Uygulama düzeyinde önden okuma özelliğini yapılandırmak için:
  - MQOPEN işlev çağrısındaki MQOO_READ_ÖNDEN seçeneğini kullanarak önden okuma özelliğini kullanabilirsiniz. DEFREADA kuyruk özniteliği DISABLED olarak ayarlandıysa, istemci uygulamasının önden okuma özelliğini kullanması olanaklı değildir.
  - Bir kuyrukta önden okuma etkinleştirildiğinde yalnızca önden okuma özelliğini kullanmak için MQOPEN işlev çağrısında MQOO_READ_AHEAD_AS_Q_DEF seçeneğini kullanın.

Bir istemci uygulaması tasarımı önden okumaya uygun değilse, bunu geçersiz kılabilirsiniz:

- kuyruk özniteliğini ayarlayarak kuyruk düzeyinde DEFREADA, bir istemci uygulaması tarafından istenmedikçe önden okuma özelliğinin kullanılmasını istemiyorsanız DEFREADA, önden okuma özelliğinin bir istemci uygulaması tarafından gerekip gerekmediğine bakılmaksızın kullanılmasını istemiyorsanız DISABLED değerini kullanın.
- MQOPEN işlev çağrısında MQOO_NO_READ_ÖNDEN seçeneğini kullanarak uygulama düzeyinde.

İki MQCLOSE seçeneği, kuyruk kapalıysa, önden okuma arabelleğinde saklanmakta olan iletilere ne olacağını yapılandırmanızı sağlar.

- Önden okuma arabelleğindeki iletileri atmak için MQCO_IMMEDIATE komutunu kullanın.
- Kuyruk kapatılmadan önce önden okuma arabelleğindeki iletilerin uygulama tarafından tüketilmesini sağlamak için MQCO_QUIESCE ' yi kullanın. MQCO_QUIESCE ile MQCLOSE yayınlandığında ve önden okuma arabelleğinde kalan iletiler olduğunda, MQRC_READ_AHEAD_MSGS MQCC_WARNING ile geri döner.

AIX üzerinde kalıcı olmayan iletiler için performansı ayarlama
AIX V5.3 ya da sonraki bir yayın düzeyini kullanıyorsanız, ayarlama değiştirgesini kalıcı olmayan iletiler için tam başarımı kullanacak şekilde ayarlayın.

Ayarlama değiştirgesini hemen yürürlüğe girecek şekilde ayarlamak için kök kullanıcı olarak aşağıdaki komutu verin:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

Ayarlama değiştirgesinin hemen yürürlüğe girmesi ve yeniden önyüklemeye devam etmesi için ayarlama değiştirgesini ayarlamak üzere kök kullanıcı olarak aşağıdaki komutu verin:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Normalde, kalıcı olmayan iletiler yalnızca bellekte tutulur, ancak AIX ' in kalıcı olmayan iletileri diske yazılacak şekilde zamanlayabileceği durumlar vardır. Diske yazılmak üzere zamanlanan iletiler, disk yazma işlemi tamamlanıncaya kadar MQGET için kullanılamaz. Önerilen ayarlama komutu bu eşiği değiştiriyor; 16 kilobayt veri kuyruğa gönderildiğinde iletileri diske yazılacak şekilde zamanlamak yerine, diske yazma işlemi yalnızca makinedeki gerçek saklama alanı dolunaya yaklaştığında gerçekleşir. Bu genel bir değişmedir ve diğer yazılım bileşenlerini etkileyebilir.

AIX'de, çoklu iş parçacıklı uygulamalar kullanırken ve özellikle birden çok işlemcili makinelerde çalışırken, daha iyi performans ve daha sağlam zamanlama için `.profile` mqm ID' de AIXTHREAD_SCOPE=S ayarını ya da uygulamayı başlatmadan önce ortamda AIXTHREAD_SCOPE=S ayarını belirlemenizi şiddetle öneririz. Örneğin:

```
export AIXTHREAD_SCOPE=S
```

AIXTHREAD_SCOPE=S ayarı, varsayılan özniteliklerle yaratılan kullanıcı iş parçacıklarının sistem genelinde çekişme kapsamına yerleştirildiği anlamına gelir. Bir kullanıcı iş parçacığı sistem genelinde çekişme kapsamıyla yaratıldıysa, bir çekirdek iş parçacığına bağlanır ve çekirdek tarafından zamanlanır. Temel çekirdek iş parçacığı başka bir kullanıcı iş parçacığıyla paylaşılmıyor.

Dosya tanımlayıcıları

Aracı işlemi gibi çok iş parçacıklı bir işlemi çalıştırırken, dosya tanımlayıcıları için geçici sınıra ulaşabilirsiniz. Bu sınır size IBM MQ neden kodu `MQRC_UNEXPECTED_ERROR (2195)` ve yeterli dosya tanımlayıcısı varsa bir IBM MQ FFST™ dosyası verir.

Bu sorunu önlemek için, dosya tanımlayıcı sayısına ilişkin işlem sınırını artırabilirsiniz. Bunu yapmak için, mqm kullanıcı kimliği için ya da varsayılan kıtada `/etc/security/limits` içindeki `nofiles` özniteliğini 10.000 olarak değiştirin.

Sistem Kaynağı Sınırları

Bir komut isteminde aşağıdaki komutları kullanarak veri kesimi ve yığın kesimine ilişkin sistem kaynağı sınırını sınırsız olarak ayarlayın:

```
ulimit -d unlimited
ulimit -s unlimited
```

Dizin tipi

Kuyruk özniteliği *IndexType*, kuyruk yöneticisinin kuyruktaki MQGET işlemlerinin hızını artırmak için sakladığı dizin tipini belirtir.

Not: Yalnızca IBM MQ for z/OSüzerinde desteklenir.

Beş seçeneğiniz var:

Değer	Açıklama
YOK	Dizin tutulmaz. İletileri sıralı olarak alırken bunu kullanın (bkz. “Öncelik” sayfa 741).
GRUPID	Grup tanıtıcılarından oluşan bir dizin korunur. İleti gruplarının mantıksal sıralamasını istiyorsanız bu dizin tipini kullanmalısınız (bkz. “Mantıksal ve fiziksel sıralama” sayfa 742).
MSGID	İleti tanıtıcılarından oluşan bir dizin korunur. MQGET çağrısında seçim ölçütü olarak *MsgId* alanını kullanan iletileri alırken bunu kullanın (bkz. “Belirli bir iletiyi alma” sayfa 753).
MSGTOKEN	İleti simgelerinden oluşan bir dizin korunur.
KORRELID	İlinti tanıtıcılarından oluşan bir dizin korunur. MQGET çağrısında seçim ölçütü olarak *CorrelId* alanını kullanan iletileri alırken bunu kullanın (bkz. “Belirli bir iletiyi alma” sayfa 753).

Not:

1. MSGID seçeneğini ya da CORRELID seçeneğini kullanarak dizinleme yapıyorsanız, MQMD ' deki göreli **MsgId** ya da **CorrelId** değiştirgelerini ayarlayın. İkisini de ayarlamak yararlı olmaz.

2. Bir kuyruk aşağıdaki tüm koşullarla eşleşirse, Göz At işlevi iletiyi bulmak için dizin mekanizmasını kullanır:
   - MSGID, CORRELID ya da GROUPID dizin tipine sahip
   - Aynı tanıtıcı tipiyle göz atılır
   - Yalnızca bir önceliğe sahip iletileri var
3. Yeniden başlatma süresini etkilediği için binlerce ileti içeren kuyruklardan (*MsgId* ya da *CorrelId*tarafından dizinlenen) kaçının. (Bu, yeniden başlatma sırasında silindiği için kalıcı olmayan iletiler için geçerli değildir.)
4. MSGTOKEN, z/OS iş yükü yöneticisi tarafından yönetilen kuyrukları tanımlamak için kullanılır.

IndexType özniteliğinin tam açıklaması için bkz. IndexType. **IndexType** özniteliğiyle ilgili daha fazla bilgi için bkz. "z/OS uygulamaları için tasarım ve performans konuları" sayfa 62.

4 MB ' den uzun iletileri işleme

İletiler uygulama, kuyruk ya da kuyruk yöneticisi için çok büyük olabilir. Ortama bağlı olarak IBM MQ , 4 MB ' den uzun iletilerle başa çıkmanın çeşitli yollarını sağlar.

MaxMsgLength özniteliğini, V6 ya da sonraki yayın düzeylerinde tüm IBM MQ sistemlerinde 100 MB ' ye kadar artırabilirsiniz. Bu değeri, kuyruğu kullanan iletilerin boyutunu yansıtacak şekilde ayarlayın. IBM MQ for z/OSdışındaki IBM MQ sistemlerinde şunları da yapabilirsiniz:

1. Bölümlenmiş iletileri kullanın. (İletiler, uygulama ya da kuyruk yöneticisi tarafından kesimlere ayrılabilir.)
2. Başvuru iletilerini kullanın.

Bu yaklaşımların her biri bu bölümün geri kalanında açıklanmıştır.

İleti uzunluğu üst sınırının artırılması

MaxMsgLength kuyruk yöneticisi özniteliği, bir kuyruk yöneticisi tarafından işlenebilecek ileti uzunluğu üst sınırını tanımlar. Benzer şekilde, **MaxMsgLength** kuyruk özniteliği de bir kuyruk tarafından işlenebilecek ileti uzunluğu üst sınırıdır. Desteklenen varsayılan ileti uzunluğu üst sınırı, çalıştığınız ortama bağlıdır.

Büyük iletileri işliyor iseniz, bu öznitelikleri z/OSdışındaki platformlarda bağımsız olarak değiştirebilirsiniz. Kuyruk yöneticisi öznitelik değerini 32768 bayt-100 MB aralığında ayarlayabilirsiniz.

Uyarı: IBM MQ for z/OS üzerinde, kuyruk yöneticisinin **MaxMsgLength** özniteliği 100 MB ' de sabit olarak kodlanır.

Tüm platformlarda, kuyruk özniteliği değerini 0-100 MB aralığında ayarlayabilirsiniz.

MaxMsgLength özniteliklerinden birini ya da her ikisini değiştirdikten sonra, değişikliklerin yürürlüğe girdiğinden emin olmak için uygulamalarınızı ve kanallarınızı yeniden başlatın.

Bu değişiklikler yapıldığında, ileti uzunluğu hem kuyruk hem de kuyruk yöneticisi **MaxMsgLength** öznitelikleriden az ya da bu değere eşit olmalıdır. Ancak, var olan iletiler her iki öznitelikten de uzun olabilir.

İleti kuyruk için çok büyükse, MQRC_MSG_TOO_BIG_FOR_Q döndürülür. Benzer şekilde, ileti kuyruk yöneticisi için çok büyükse, MQRC_MSG_TOO_BIG_FOR_Q_MGR döndürülür.

Bu büyük iletileri işleme yöntemi kolay ve kullanışlıdır. Ancak, bunu kullanmadan önce aşağıdaki etkenleri göz önünde bulundurun:

- Kuyruk yöneticileri arasındaki uyumsuzluk azalır. İleti verileri büyüklüğü üst sınırı, iletinin yerleştirileceği her kuyruk (iletim kuyrukları da içinde olmak üzere) için *MaxMsgLength* tarafından belirlenir. Bu değer genellikle, özellikle iletim kuyrukları için, kuyruk yöneticisinin *MaxMsgLength*değerine ayarlanır. Bu, uzak bir kuyruk yöneticisine seyahat etmek için bir iletinin çok büyük olup olmadığını tahmin etmeyi zorlaştırır.

- Sistem kaynaklarının kullanımı artırılmıştır. Örneğin, uygulamalar daha büyük arabelleklere ihtiyaç duyar ve bazı platformlarda paylaşılan depolama kullanımı artırılabilir. Kuyruk saklama alanı yalnızca daha büyük iletiler için gerçekten gerekliyse etkilenmelidir.
- Kanal toplu işleri etkilendi. Büyük bir ileti, toplu iş sayısına doğru yalnızca bir ileti olarak sayılır, ancak iletimin daha uzun sürmesi gerekir, böylece diğer iletiler için yanıt süreleri artar.

Multi *İleti bölümlemesi*

İletilerin bölümlenmesine ilişkin bilgi edinmek için bu bilgileri kullanın. Bu özellik IBM MQ for z/OS üzerinde ya da IBM MQ classes for JMSkullanan uygulamalar tarafından desteklenmez.

"İleti uzunluğu üst sınırının artırılması" sayfa 759 konusunda açıklandığı gibi ileti uzunluğu üst sınırını artırmanın bazı olumsuz etkileri vardır. Ayrıca, iletinin kuyruk ya da kuyruk yöneticisi için çok büyük olmasına neden olabilir. Bu durumlarda, bir iletiyi bölebilir. Bölümler hakkında bilgi için, bkz. "İleti grupları" sayfa 42.

Sonraki bölümler, iletileri bölümlere ayırmaya ilişkin ortak kullanımları inceler. MQPUT ya da MQGET çağrılarının *her zaman* bir iş birimi içinde çalıştığı varsayılır. Ağda eksik grupların bulunma olasılığını azaltmak için her zaman bu tekniği kullanmayı düşünün. Kuyruk yöneticisi tarafından tek fazlı kesinleştirme kabul edilir, ancak diğer eşgüdüm teknikleri de aynı derecede geçerlidir.

Ayrıca, alma uygulamalarında, birden çok sunucu aynı kuyruğu işliyorsa, her bir sunucunun benzer bir kod yürüttüğü varsayılır; böylece bir sunucu, orada olmasını beklediği bir iletiyi ya da bölümü asla bulamaz (çünkü daha önce MQGMO_ALL_MSGS_KULLANILABILIR ya da MQGMO_ALL_SEGMENTS_MEVCUTTUR).

Uyarı: Bir konuya ileti göndermek (ya da bir konu diğer adına ileti yerleştirmek) için yayınlama/abone olma kullanılırken, ileti gruplamaya ve bölümlemeye izin verilmez.

Abonelikler yayın etkinliğinden bağımsız olarak yaratılabileceği ve kaldırılabileceği için, bir abonenin tam bir ileti grubu ya da iletinin tüm bölümlerini alacağı garanti edilemez; bkz. RC2417: MQRC_MSG_NOT_ALLOWED_IN_GROUP.

İş birimlerini kapsayan bölümlenmiş bir ileti koyma ve alma

Bir iş birimini "İş birimlerini kapsayan bir grubu koyma ve alma" sayfa 750' e benzer şekilde kapsayan bölümlenmiş bir ileti koyabilir ve alabilirsiniz.

Ancak, genel bir iş birimine bölümlenmiş iletiler koyamaz ya da alamazsınız.

Multi *Kuyruk yöneticisine göre bölümleme ve yeniden çevirme*

Bu, bir uygulamanın bir iletiyi başka bir uygulama tarafından alınmak üzere koyduğu en basit senaryodur. İleti büyük olabilir: koyma ya da alma uygulamasının tek bir arabellekte işlemesi için çok büyük değil, ancak kuyruk yöneticisi ya da iletinin konacağı bir kuyruk için çok büyük.

Bu uygulamalar için gereken tek değişiklik, koyma uygulamasının kuyruk yöneticisine gerektiğinde bölümleme gerçekleştirmesi için yetki vermesine ilişkidir:

```
   PMO.Options = (existing options)
   MD.MsgFlags = MQMF_SEGMENTATION_ALLOWED
   MD.Version = MQMD_VERSION_2
   memcpy(MD.GroupId, MQGI_NONE, MQ_GROUP_ID_LENGTH)
MQPUT
```

ve alma uygulamasının kuyruk yöneticisinden, kesimlere ayrılmışsa iletiyi yeniden birleştirmesini istemesi için:

```
   GMO.Options = MQGMO_COMPLETE_MSG | (existing options)
   MQGET
```

Bu en basit senaryoda, kuyruk yöneticisinin her ileti için benzersiz bir grup tanıtıcısı oluşturabilmesi için, uygulamanın MQPUT çağrısının öncesinde GroupId alanını MQGI_NONE olarak sıfırlaması gerekir.

Bu yapılmazsa, ilgisiz iletiler aynı grup tanıtıcısına sahip olabilir ve bu da daha sonra yanlış işlemeye yol açabilir.

Uygulama arabelleği, yeniden birleştirilen iletiyi içerecek kadar büyük olmalıdır (MQGMO_ACCEPT_TRUNCATED_MSG seçeneğini eklemediğiniz sürece).

Bir kuyruğun MAXMSGLEN özniteliği, ileti bölümlemesini kapsayacak şekilde değiştirilirse, şunları göz önünde bulundurun:

- Yerel kuyrukta desteklenen ileti kesimi alt sınırı 16 bayttır.
- Bir iletim kuyruğu için, MAXMSGLEN üstbilgiler için gereken alanı da içermelidir. İletim kuyruğuna konabilecek herhangi bir ileti kesiminde beklenen kullanıcı verileri uzunluğu üst sınırından en az 4000 bayt daha büyük bir değer kullanmayı düşünün.

Veri dönüştürme gerekiyorsa, alma uygulamasının bunu MQGMO_CONVERT belirtilerek yapması gerekebilir. Veri dönüştürme çıkışı eksiksiz bir iletiyle birlikte sunulduğundan bu anlaşılır olmalıdır. İleti bölümlendiyse, gönderici kanalındaki verileri dönüştürme girişiminde bulunmayın ve verilerin biçimi, veri dönüştürme çıkışının tamamlanmamış verilerde dönüştürmeyi gerçekleştiremeyeceği biçimindedir.

Multi *Uygulama bölümlemesi*

Kuyruk yöneticisi bölümlemesi yeterli olmadığında ya da uygulamalar belirli kesim sınırlarıyla veri dönüştürme gerektirdiğinde uygulama bölümlemesi kullanılır.

Uygulama bölümlemesi iki temel nedenden ötürü kullanılır:

1. İleti uygulamalar tarafından tek bir arabellekte işlenemeyecek kadar büyük olduğundan, kuyruk yöneticisi bölümlemesi tek başına yeterli değil.
2. Veri dönüştürme işlemi gönderen kanallar tarafından yapılmalı ve biçim, tek bir kesimin dönüştürülmesinin mümkün olabilmesi için bölüm sınırlarının nerede olacağını belirlemesi gerekecektir.

Ancak, veri dönüştürme bir sorun değilse ya da alma uygulaması her zaman MQGMO_COMPLETE_MSG kullanıyorsa, kuyruk yöneticisi bölümlemesine MQMF_SEGMENTATION_ALLOWED belirtilerek de izin verilebilir. Örneğimizde, uygulama iletiyi dört bölüme ayırır:

```
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
```

MQPMO_LOGICAL_ORDER kullanmıyorsanız, uygulama *Offset* değerini ve her bir kesimin uzunluğunu ayarlamalıdır. Bu durumda, mantıksal durum otomatik olarak korunmaz.

Alma uygulaması, yeniden bir araya getirilmiş iletiyi tutacak kadar büyük bir arabelleğe sahip olacağını garanti edemez. Bu nedenle, bölümlerin tek tek işlenmesine hazır olmalıdır.

Bölümlenmiş iletiler için bu uygulama, mantıksal iletiyi oluşturan tüm kesimler bulununcaya kadar tek bir kesimi işlemeye başlamak istemez. Bu nedenle, ilk bölüm için MQGMO_ALL_SEGMENTS_KULLANILABILIR. MQGMO_LOGICAL_ORDER belirtirseniz ve yürürlükteki bir mantıksal ileti varsa, MQGMO_ALL_SEGMENTS_KULLANILABILIR yoksayılır.

Mantıksal iletinin ilk bölümü alındıktan sonra, mantıksal iletinin geri kalan bölümlerinin sırayla alındığından emin olmak için MQGMO_LOGICAL_ORDER komutunu kullanın.

Farklı gruplar içindeki iletiler dikkate alınmaz. Bu tür iletiler oluşursa, bunlar her iletinin ilk bölümünün kuyrukta gerçekleştiği sırayla işlenir.

```
GMO.Options = MQGMO_SYNCPOINT | MQGMO_LOGICAL_ORDER
            | MQGMO_ALL_SEGMENTS_AVAILABLE | MQGMO_WAIT
do while ( SegmentStatus == MQSS_SEGMENT )
```

```
   MQGET
   /* Process each remaining segment of the logical message */
   ...

MQCMIT
```

Multi *Mantıksal iletilerin uygulama bölümlemesi*

İletiler bir grup içinde mantıksal sırada tutulmalıdır ve bazıları ya da tümü uygulama bölümlemesi gerektirecek kadar büyük olabilir.

Bizim örneğimizde, dört mantıksal mesajdan oluşan bir grup konulacaktır. Üçüncü ileti dışında tümü büyük ve bölümleme gerektiriyor, bu da koyma uygulaması tarafından gerçekleştirilir:

```
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
```

Alma uygulamasında, ilk MQGET ' de MQGMO_ALL_MSGS_VAR belirtildi. Bu, tüm grup kullanılabilir oluncaya kadar bir grubun iletilerinin ya da bölümlerinin alınmadığı anlamına gelir. Bir grubun ilk fiziksel iletisi alındığında, grubun bölümlerinin ve iletilerinin sırayla alındığından emin olmak için MQGMO_LOGICAL_ORDER kullanılır:

```
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
```

Not: MQGMO_LOGICAL_ORDER belirtirseniz ve yürürlükteki bir grup varsa, MQGMO_ALL_MSGS_CURRENT yoksayılır.

Başvuru iletileri

Başvuru iletileriyle ilgili daha fazla bilgi edinmek için bu bilgileri kullanın.

Not: IBM MQ for z/OSiçinde desteklenmez.

Bu yöntem, büyük bir nesnenin kaynak ya da hedef düğümlerdeki IBM MQ kuyruklarında saklanmadan bir düğümden diğerine aktarılmasını sağlar. Bu, özellikle, veriler başka bir formda (örneğin, posta uygulamaları için) bulunduğunda yarar sağlar.

Bunu yapmak için, bir kanalın her iki ucunda bir ileti çıkışı belirtirsiniz. Bunun nasıl yapılacağını öğrenmek için bkz. “Kanal iletisi çıkış programları” sayfa 938.

IBM MQ , başvuru iletisi üstbilgisinin (MQRMH) biçimini tanımlar. Bu konuya ilişkin açıklamalar için bkz. MQRMH . Bu, tanımlı bir biçim adıyla tanınır ve bunu gerçek veriler izler.

Büyük bir nesnenin aktarılmasını başlatmak için uygulama, izleyen veri olmadan bir başvuru iletisi üstbilgisinden oluşan bir ileti yerleştirebilir. Bu ileti düğümden ayrıldıkça, ileti çıkışı nesneyi uygun bir

şekilde alır ve başvuru iletisine ekler. Daha sonra, iletiyi alan MCA ' ya iletilmek üzere gönderen İleti Kanal Aracısı 'na (şimdi öncekinden daha büyük) döndürür.

Alan MCA ' da başka bir ileti çıkışı yapılandırıldı. Bu ileti çıkışı bu iletilerden birini aldığında, eklenen nesne verilerini kullanarak nesneyi yaratır ve bu ileti *olmadan* başvuru iletisine iletir. Başvuru iletisi artık bir uygulama tarafından alınabilir ve bu uygulama nesnenin (ya da en azından bu başvuru iletisiyle gösterilen kısmının) bu düğümde yaratıldığını bilir.

Gönderen ileti çıkışının başvuru iletisine ekleyebileceği nesne verisi miktarı üst sınırı, kanala ilişkin kararlaştırılan ileti uzunluğu üst sınırıyla sınırlıdır. Çıkış, iletilen her ileti için MCA ' ya yalnızca tek bir ileti döndürebilir; bu nedenle, koyma uygulaması bir nesnenin aktarılmasına neden olacak birkaç ileti yerleştirebilir. Her ileti, sonuna eklenecek nesnenin *mantıksal* uzunluğunu ve göreli konumunu tanımlamalıdır. Ancak, nesnenin toplam boyutunu ya da kanal tarafından izin verilen boyut üst sınırını bilmenin mümkün olmadığı durumlarda, gönderen ileti çıkışını, koyma uygulamasının tek bir ileti koyacağı şekilde tasarlayın ve çıkış, iletilen iletiye mümkün olduğunca çok veri eklediğinde, bir sonraki iletiyi iletim kuyruğuna koyar.

Büyük iletilerle uğraşmak için bu yöntemi kullanmadan önce aşağıdaki noktaları göz önünde bulundurun:

- MCA ve ileti çıkışı bir IBM MQ kullanıcı kimliği altında çalışır. İleti çıkışının (ve bu nedenle kullanıcı kimliğinin) nesneyi gönderme sonunda alması ya da alma sonunda yaratması için nesneye erişmesi gerekir; bu yalnızca nesnenin evrensel olarak erişilebilir olduğu durumlarda uygulanabilir olabilir. Bu bir güvenlik sorununu gündeme getiriyor.
- Sonuna toplu veri eklenmiş başvuru iletisinin, hedefine ulaşmadan önce birden çok kuyruk yöneticisiyle seyahat etmesi gerekiyorsa, toplu veriler, araya giren düğümlerdeki IBM MQ kuyruklarında bulunur. Ancak, bu durumlarda özel destek veya çıkış sağlanmasına gerek yoktur.
- Yeniden yönlendirmeye ya da ileti kuyruğuna izin verildiğinde ileti çıkışınızın tasarlanması zorlaşır. Bu durumlarda, nesnenin bölümleri sıradışı gelebilir.
- Bir başvuru iletisi hedefine ulaştığında, alan ileti çıkışı nesneyi oluşturur. Ancak bu, MCA ' nın iş birimiyle eşitlenmez; bu nedenle, toplu iş geri çekilirse, nesnenin aynı bölümünü içeren başka bir başvuru iletisi daha sonra bir toplu işe gelir ve ileti çıkışı nesnenin aynı bölümünü yeniden oluşturmayı deneyebilir. Nesne, örneğin, bir veritabanı güncellemeleri serisi ise, bu kabul edilemez olabilir. Bu durumda, ileti çıkışı hangi güncellemelerin uygulandığını içeren bir günlüğü tutmalıdır; bu bir IBM MQ kuyruğunun kullanılmasını gerektirebilir.
- Nesne tipinin özelliklerine bağlı olarak, nesnenin artık gerekli olmadığında silinebilmesi için, ileti çıkışlarının ve uygulamalarının kullanım sayılarının korunmasında işbirliği yapması gerekebilir. Bir eşgörünüm tanıtıcısı da gerekli olabilir; başvuru iletisi üstbilgisinde bunun için bir alan sağlanmıştır (bkz. MQRMH).
- Bir başvuru iletisi dağıtım listesi olarak konursa, sonuçta ortaya çıkan her dağıtım listesi ya da o düğümdeki tek tek hedef için nesne alınabilmelidir. Kullanım sayılarını korumalısınız. Ayrıca, bir düğümün listedeki bazı hedefler için son düğüm, ancak diğerleri için bir ara düğüm olma olasılığını da göz önünde bulundurun.
- Toplu veriler genellikle dönüştürülmez. Bunun nedeni, dönüştürme işleminin ileti çıkışı çağrılmadan *önce* gerçekleşmesi olabilir. Bu nedenle, kaynak gönderen kanalında dönüştürme istenmemelidir. Başvuru iletisi bir ara düğümden geçerse, istenirse, toplu veriler ara düğümden gönderildiğinde dönüştürülür.
- Başvuru iletileri bölümlenemez.

MQRMH ve MQMD yapılarının kullanılması

Başvuru iletisi üstbilgisindeki ve ileti tanımlayıcısındaki alanların açıklaması için MQRMH ve MQMD bölümüne bakın.

MQMD yapısında, *Format* alanını MQFMT_REF_MSG_HEADER olarak ayarlayın. MQGET üzerinde istendiğinde MQHREF biçimi, izleyen toplu verilerle birlikte IBM MQ tarafından otomatik olarak dönüştürülür.

Aşağıda, MQRMH ' nin *DataLogicalOffset* ve *DataLogicalLength* alanlarının kullanımına bir örnek verilmiştir:

Bir koyma uygulaması aşağıdaki özelliklere sahip bir başvuru iletisi koyabilir:

- Fiziksel veri yok
- *DataLogicalLength* = 0 (bu ileti tüm nesneyi gösterir)
- *DataLogicalOffset* = 0.

Nesnenin 70 000 bayt uzunluğunda olduğu varsayılarak, gönderen ileti çıkışı, aşağıdaki bilgileri içeren bir başvuru iletisinde kanal boyunca ilk 40 000 baytı gönderir:

- MQRMH ' yi izleyen 40 000 bayt fiziksel veri
- *DataLogicalLength* = 40000
- *DataLogicalOffset* = 0 (nesnenin başlangıcından itibaren).

Daha sonra, iletim kuyruğuna şunları içeren başka bir ileti yerleştirir:

- Fiziksel veri yok
- *DataLogicalLength* = 0 (nesnenin sonuna kadar). Burada 30.000 değerini belirtebilirsiniz.
- *DataLogicalOffset* = 40000 (bu noktadan başlayarak).

Bu ileti çıkışı gönderen ileti çıkışı tarafından görüldüğünde, geri kalan 30.000 bayt veri eklenir ve alanlar şu değere ayarlanır:

- MQRMH ' yi izleyen 30.000 bayt fiziksel veri
- *DataLogicalLength* = 30000
- *DataLogicalOffset* = 40000 (bu noktadan başlayarak).

MQRMHF_LAST işareti de ayarlanır.

Başvuru iletilerinin kullanımı için sağlanan örnek programlara ilişkin açıklamalar için bkz. “Multiplatforms üzerinde örnek programların kullanılması” sayfa 1014.

İleti bekleniyor

Bir programın kuyruğa bir ileti gelinceye kadar beklemesini istiyorsanız, MQGMO yapısının *Options* alanında MQGMO_WAIT seçeneğini belirtin.

Belirtmek için MQGMO yapısının *WaitInterval* alanını kullanın Bir MQGET çağrısına bir iletinin kuyruğa ulaşmasını bekleme süresi üst sınırı (milisaniye cinsinden).

İleti bu süre içinde gelmezse, MQGET çağrısı MQRC_NO_MSG_AVAILABLE neden koduyla tamamlanır.

WaitInterval alanında MQWI_UNLIMITED değişmezini kullanarak sınırsız bekleme aralığı belirtebilirsiniz. Ancak, denetiminizin dışındaki olaylar programınızın uzun süre beklemesine neden olabilir, bu nedenle bu sabiti dikkatli kullanın. IMS uygulamaları, IMS sisteminin sonlandırılmasını önleyeceği için sınırsız bekleme aralığı belirtmemelidir. (IMS sona erdiğinde, tüm bağımlı bölgelerin sona ermesini gerektirir.) Bunun yerine, IMS uygulamaları sonlu bir bekleme aralığı belirtebilir; daha sonra, çağrı o aralıktan sonra ileti almadan tamamlanırsa, bekleme seçeneğiyle başka bir MQGET çağrısı yürütün.

Not: Bir iletiyi *kaldırmak* için aynı paylaşılan kuyrukta birden çok program bekliyorsa, yalnızca bir program gelen ileti tarafından etkinleştirilir. Ancak, bir iletiye göz atmak için birden çok program bekliyorsa, tüm programlar etkinleştirilebilir. Daha fazla bilgi için, MQGMO' daki MQGMO yapısının *Options* alanının tanımına bakın.

Bekleme aralığı sona ermeden önce kuyruğun ya da kuyruk yöneticisinin durumu değişirse, aşağıdaki işlemler gerçekleşir:

- Kuyruk yöneticisi susturma durumuna girerse ve MQGMO_FAIL_IF_QUIESCING seçeneğini kullandıysanız, bekleme iptal edilir ve MQGET çağrısı MQRC_Q_MGR_QUIESCING neden koduyla tamamlanır. Bu seçenek olmadan, arama beklemede kalır.

- z/OSz/OSsistemlerinde, bağlantı (bir CICS ya da IMS uygulaması için) susturma durumuna girerse ve MQGMO_FAIL_IF_QUIESCING seçeneğini kullandıysanız, bekleme iptal edilir ve MQGET çağrısı MQRC_CONN_QUIESCING neden koduyla tamamlanır. Bu seçenek olmadan, arama beklemede kalır.
- Kuyruk yöneticisi durdurulmaya zorlanıyorsa ya da iptal edildiyse, MQGET çağrısı MQRC_Q_MGR_STOP ya da MQRC_CONNECTION_BROKEN neden koduyla tamamlanır.
- Kuyruğun öznitelikleri (ya da kuyruk adının çözüldüğü bir kuyruk), alma isteklerinin engelleneceği şekilde değiştirilirse, bekleme iptal edilir ve MQGET çağrısı MQRC_GET_INLENEN neden koduyla tamamlanır.
- Kuyruğun öznitelikleri (ya da kuyruk adının çözüldüğü bir kuyruk) FORCE seçeneğinin gerekli olduğu şekilde değiştirilirse, bekleme iptal edilir ve MQGET çağrısı MQRC_OBJECT_CHANGED neden koduyla tamamlanır.

z/OSUygulamanızın birden çok kuyrukta beklemesini istiyorsanız, IBM MQ for z/OS sinyal olanağını kullanın (bkz. “Sinyal” sayfa 765). Bu işlemlerin gerçekleştiği durumlarla ilgili daha fazla bilgi için bkz. MQGMO.

Sinyal

Sinyal yalnızca IBM MQ for z/OSüzerinde desteklenir.

İşaret verme, MQGET çağrısındaki, işletim sisteminin bunu bildirmesine (ya da *sinyal*) izin veren bir seçenektir. Beklenen bir ileti kuyruğa geldiğinde bir program. Bu, programınızın sinyali beklerken başka bir işle devam etmesine izin verdiğinden, “İleti bekleniyor” sayfa 764 konusunda açıklanan *get with wait* işlevine benzer. Ancak, sinyali kullanırsanız, uygulama iş parçacığını serbest bırakabilir ve bir ileti geldiğinde programı bilgilendirmek için işletim sistemine güvenebilirsiniz.

Bir sinyal ayarlamak için

Bir sinyal ayarlamak için, MQGET çağrısında kullandığınız MQGMO yapısında aşağıdakileri yapın:

1. `Options` alanında MQGMO_SET_SIGNAL seçeneğini ayarlayın.
2. `WaitInterval` alanında işaretin maksimum ömrünü ayarlayın. Bu, IBM MQ ' in kuyruğu izlemesini istediğiniz süreyi (milisaniye cinsinden) belirler. Sınırsız kullanım ömrü belirtmek için MQWI_UNLIMITED değerini kullanın.

   Not: IMS uygulamaları, IMS sisteminin sonlandırılmasını önleyeceği için sınırsız bekleme aralığı belirtmemelidir. (IMS sona erdiğinde, tüm bağımlı bölgelerin sona ermesini gerektirir.) Bunun yerine, IMS uygulamaları ECB ' nin durumunu düzenli aralıklarla inceleyebilir (3. adıma bakın). Bir program, aynı anda birden çok kuyruk tanıtıcısında işaretler ayarlayabilir:
3. `Signal1` alanında *Event Control Block* (ECB) adresini belirtin. Bu size sinyalinizin sonucunu bildirir. ECB saklama alanı, kuyruk kapatılıncaya kadar kullanılabilir durumda kalmalıdır.

Not: MQGMO_SET_SIGNAL seçeneğini MQGMO_WAIT seçeneğiyle kullanamazsınız.

İleti geldiğinde

Uygun bir ileti geldiğinde, ECB ' ye bir tamamlanma kodu döndürülür.

Tamamlanma kodu aşağıdakilerden birini açıklar:

- Sinyali ayarladığınız ileti kuyruğa geldi. İleti, sinyal isteğinde bulunan program için ayrılmamış; bu nedenle, programın iletiyi almak için bir MQGET çağrısı yeniden yürütmesi gerekir.

  Not: Başka bir uygulama, sinyali alma ve başka bir MQGET çağrısı verme arasındaki zamanda iletiyi alabilir.
- Ayarladığınız bekleme aralığının süresi doldu ve sinyali ayarladığınız ileti kuyruğa gelmedi. IBM MQ işareti iptal etti.

- Sinyal iptal edildi. Örneğin, kuyruk yöneticisi durursa ya da kuyruğun özniteliği değiştirilirse, MQGET çağrılarına artık izin verilmeyecek şekilde bu durum oluşur.

Kuyrukta uygun bir ileti olduğunda, MQGET çağrısı, sinyal vermeden MQGET çağrısıyla aynı şekilde tamamlanır. Ayrıca, hemen bir hata saptanırsa, çağrı tamamlanır ve dönüş kodları ayarlanır.

Çağrı kabul edildiğinde ve hemen kullanılabilir bir ileti olmadığında, diğer çalışmalara devam edebilmesi için denetim programa döndürülür. İleti tanımlayıcıdaki çıkış alanlarının hiçbiri ayarlanmadı, ancak **CompCode** değiştirgesi MQCC_WARNING olarak ayarlandı ve **Reason** değiştirgesi MQRC_SIGNAL_REQUEST_ACCEPTED olarak ayarlandı.

IBM MQ ' in sinyal göndermeyi kullanarak MQGET çağrısı yaptığında uygulamanıza nelerin geri dönebileceği hakkında bilgi için bkz. MQGET.

Programın ECB 'nin gönderilmesini beklerken yapması gereken başka bir iş yoksa, ECB' nin aşağıdaki işlemleri kullanmasını bekleyebilir:

- CICS Transaction Server for z/OS programı için EXEC CICS WAIT EXTERNAL komutu
- Toplu iş ve IMS programları için z/OS WAIT makrosu

İşaret ayarlanırken kuyruğun ya da kuyruk yöneticisinin durumu değişirse (yani, ECB henüz gönderilmemişse), aşağıdaki işlemler gerçekleşir:

- Kuyruk yöneticisi susturma durumuna girerse ve MQGMO_FAIL_IF_QUIESCING seçeneğini kullandıysanız, sinyal iptal edilir. ECB, MQEC_Q_MGR_QUIESCING tamamlama koduyla gönderilir. Bu seçenek olmadan, sinyal ayarlanmış olarak kalır.
- Kuyruk yöneticisi durdurulmaya zorlanıyorsa ya da iptal edildiyse, sinyal iptal edilir. Sinyal, MQEC_WAIT_İPTAL edilen tamamlanma koduyla teslim edilir.
- Alma isteklerinin engellenmesi için kuyruğun öznitelikleri (ya da kuyruk adının çözüldüğü bir kuyruk) değiştirilirse, sinyal iptal edilir. Sinyal, MQEC_WAIT_İPTAL edilen tamamlanma koduyla teslim edilir.

Not:

1. Bir iletiyi kaldırmak için aynı paylaşılan kuyrukta birden fazla program bir sinyal belirlediyse, yalnızca bir program gelen bir ileti tarafından etkinleştirilir. Ancak, bir iletiye göz atmak için birden çok program bekliyorsa, tüm programlar etkinleştirilebilir. Hangi uygulamaların etkinleştirileceğine karar verirken kuyruk yöneticisinin izlediği kurallar, bekleyen uygulamalarla aynıdır: Daha fazla bilgi için, MQGMO-Get-message optionsiçindeki MQGMO yapısının *Options* alanının açıklamasına bakın.
2. Aynı iletiyi bekleyen birden çok MQGET çağrısı varsa, bekleme ve sinyal seçenekleri karışımıyla birlikte, her bekleyen çağrı eşit olarak değerlendirilir. Daha fazla bilgi için MQGMO-Get-message seçenekleriiçindeki MQGMO yapısının *Options* alanının tanımına bakın.
3. Bazı koşullarda, hem bir MQGET çağrısının bir iletiyi alması, hem de bir sinyalin (aynı iletinin gelişinden kaynaklanan) teslim edilmesi mümkündür. Bu, programınız başka bir MQGET çağrısı yayınladığında (sinyal teslim edildiği için), kullanılabilir bir ileti olmadığı anlamına gelir. Bu durumu test etmek için programınızı tasarlayın.

Bir sinyalin nasıl ayarlanacağına ilişkin bilgi için Signal1içindeki MQGMO_SET_SIGNAL seçeneğinin tanımına ve *Signal1* alanına bakın.

Geriletme atlanıyor

MQGET çağrısında **MQGMO_MARK_SKIP_BACKOUT** seçeneğini belirterek bir uygulama programının *MQGET-error-backout* döngüsüne girmesini önleyebilirsiniz.

Not: Yalnızca IBM MQ for z/OSüzerinde desteklenir.

Bir iş biriminin bir parçası olarak, bir uygulama programı kuyruktan ileti almak için bir ya da daha çok MQGET çağrısı yayınlayabilir. Uygulama programı bir hata saptarsa, iş birimini geri alabilir. Bu, iş birimi sırasında güncellenen tüm kaynakları, iş birimi başlamadan önce bulundukları duruma geri yükler ve MQGET çağrıları tarafından alınan iletileri yeniden işler.

Bu iletiler, uygulama programı tarafından yayınlanan sonraki MQGET çağrıları için kullanılabilir. Birçok durumda bu, uygulama programı için bir soruna neden olmaz. Ancak, geriletme durumuna yol açan

hatanın engellenememesi durumunda, iletinin kuyrukta yeniden yürürlüğe girmesi, uygulama programının *MQGET-error-backout* döngüsüne girmesine neden olabilir.

Bu sorunu önlemek için MQGET çağrısında MQGMO_MARK_SKIP_BACKOUT seçeneğini belirtin. Bu, MQGET isteğini uygulama tarafından başlatılan arka plana dahil edilmiyor olarak işaretler; yani, geri çekilmemelidir. Bu seçeneğin kullanılması, bir geriletme oluştuğunda, diğer kaynaklarda yapılan güncellemelerin gerektiği gibi geriletildiği, ancak işaretli iletinin yeni bir iş birimi altında alınmış gibi işlem gördüğü anlamına gelir.

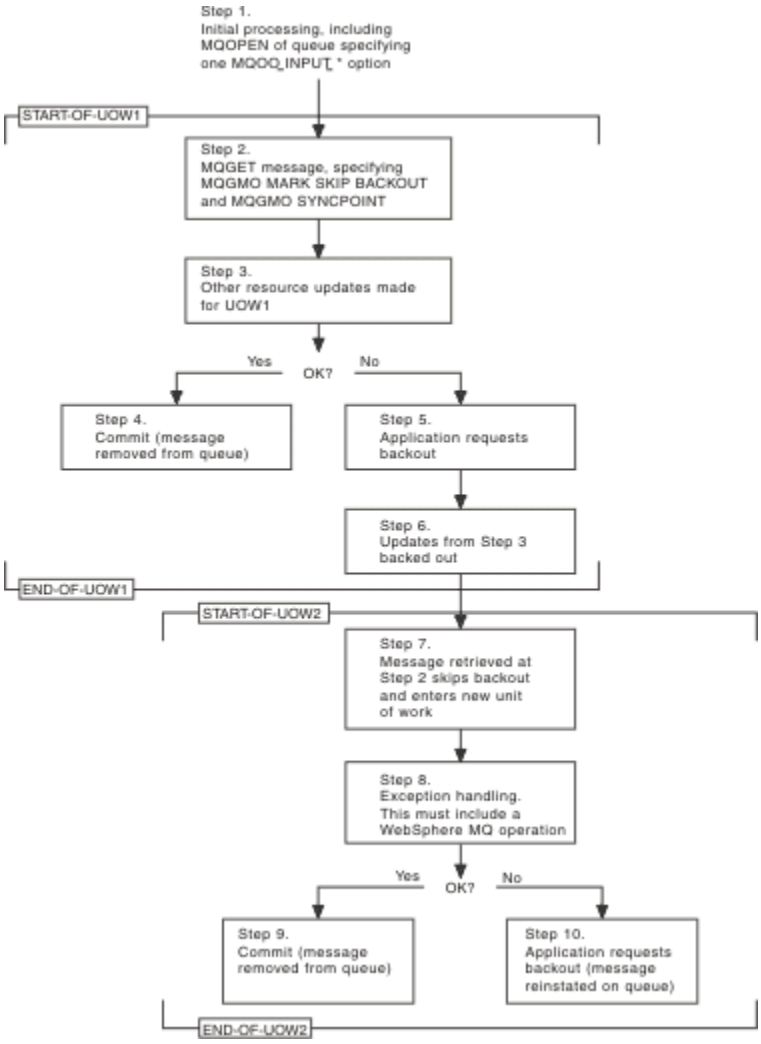
Uygulama programı, yeni iş birimini kesinleştirmek ya da yeni iş birimini geri almak için bir IBM MQ çağrısı yayınlamalıdır. Örneğin, program, iletinin atıldığını oluşturana bildirmek ve iş birimini kesinleştirerek iletiyi kuyruktan kaldırmak gibi kural dışı durum işlemleri gerçekleştirebilir. Yeni iş birimi yedeklenirse (herhangi bir nedenle) ileti kuyrukta yeniden görüntülenir.

Bir iş birimi içinde, geriye doğru atlanıyor olarak işaretlenmiş tek bir MQGET isteği olabilir; ancak, geri dönüş atlanıyor olarak işaretlenmemiş başka iletiler de olabilir. Bir ileti geri dönüş atlanıyor olarak işaretlendikten sonra, iş birimi içinde MQGMO_MARK_SKIP_BACKOUT belirten diğer MQGET çağrıları MQRC_SECOND_MARK_NOT_ALLOWED neden koduyla başarısız olur.

Not:

1. İşaretli ileti, yalnızca bunu içeren iş birimi, bunu desteklemek için bir uygulama isteği tarafından sonlandırılırsa geriletme atlar. İş birimi başka bir nedenle geriletildiyse, ileti, geri çekilmek üzere işaretlenmediğinde olduğu gibi kuyruğa geri çekilir.
2. RRS tarafından denetlenen iş birimlerine katılan Db2 saklanmış yordamlarında geriletme atlanması desteklenmez. Örneğin, MQGMO_MARK_SKIP_BACKOUT seçeneğini içeren bir MQGET çağrısı, MQRC_OPTION_ENVIRONMENT_ERROR neden koduyla başarısız olur.

Şekil 63 sayfa 768 içinde, geriletme işlemini atlamak için bir MQGET isteği gerektiğinde uygulama programının içerebileceği tipik bir adım sırası gösterilir.



Şekil 63. MQGMO_MARK_SKIP_BACKOUT kullanılarak geriletme atlanıyor

Şekil 63 sayfa 768 içindeki adımlar şunlardır:

Adım 1
Kuyruğun açılması için bir MQOPEN çağrısı (Adım 2 'deki kuyruktan ileti almak için MQOO_INPUT_* seçeneklerinden birinin belirtilmesi) de içinde olmak üzere, hareket içinde ilk işleme gerçekleştirilir.

Adım 2
MQGMO_SYNCPOINT ve MQGMO_MARK_SKIP_BACKOUT ile MQGET çağrıldı. MQGMO_SYNCPOINT gereklidir; MQGET, MQGMO_MARK_SKIP_BACKOUT için bir iş birimi içinde olmalıdır. Şekil 63 sayfa 768 içinde bu iş birimine UOW1denir.

Adım 3
Diğer kaynak güncellemeleri UOW1' in bir parçası olarak yapılır. Bunlar başka MQGET çağrılarını da içerebilir (MQGMO_MARK_SKIP_BACKOUT olmadan verilir).

Adım 4
Adım 2 ve 3 'teki tüm güncelleştirmeler gerektiği şekilde tamamlandı. Uygulama programı güncellemeleri kesinleştirir ve UOW1 sona erer. Adım 2 'de alınan ileti kuyruktan kaldırılır.

Adım 5
Adım 2 ve 3 'teki güncellemelerden bazıları gerektiği şekilde tamamlanmıyor. Uygulama programı, bu adımlar sırasında yapılan güncellemelerin geriletilmesini ister.

Adım 6
Adım 3 'te yapılan güncellemeler geri çekilmiştir.

Adım 7
Adım 2 'de yapılan MQGET isteği, geriletme işlemini atlar ve yeni bir iş biriminin (UOW2) parçası olur.

Adım 8
UOW2 , geriletilmekte olan UOW1 ' e yanıt olarak kural dışı durum işleme gerçekleştirir. (Örneğin, başka bir kuyruğa yönelik MQPUT çağrısı, UOW1 ' in geriletilmesine neden olan bir sorunun oluştuğunu gösterir.)

Adım 9
Adım 8 gerektiği gibi tamamlanır, uygulama programı etkinliği kesinleştirir ve UOW2 sona erer. MQGET isteği UOW2 ' nin bir parçası olduğundan (bkz. Adım 7), bu kesinleştirme iletinin kuyruktan kaldırılmasına neden olur.

Adım 10
Adım 8, gerektiği şekilde tamamlanmaz ve uygulama programı UOW2' yi geri çeviriyor. İleti alma isteği UOW2 ' nin bir parçası olduğundan (bkz. Adım 7), bu istek de geri çekilip kuyrukta yeniden kullanılır. Artık bu ya da başka bir uygulama programı tarafından yayınlanan başka MQGET çağrıları için de kullanılabilir (kuyruktaki diğer iletilerle aynı şekilde).

Uygulama verilerini dönüştürme

Gerekli olduğunda, MCA ' lar ileti tanımlayıcıyı ve üstbilgi verilerini gerekli karakter kümesine ve kodlamaya dönüştürür. Bağlantının her iki ucu (yerel MCA ya da uzak MCA) dönüştürmeyi yapabilir.

Bir uygulama bir kuyruğa ileti koyduğunda, yerel kuyruk yöneticisi, kuyruk yöneticileri ve MCA ' lar tarafından işlendiklerinde iletilerin denetimini kolaylaştırmak için ileti tanımlayıcılarına denetim bilgileri ekler. Ortama bağlı olarak, ileti üstbilgisi veri alanları, yerel sistemin karakter kümesinde ve kodlamasında yaratılır.

İletileri sistemler arasında taşıdığınızda, bazen uygulama verilerini alan sistemin gerektirdiği karakter kümesine ve kodlamaya dönüştürmeniz gerekir. Bu, alan sistemdeki uygulama programlarından ya da gönderen sistemdeki MCA ' lar tarafından yapılabilir. Alan sistemde veri dönüştürme destekleniyorsa, uygulama verilerini dönüştürmek için, gönderen sistemde önceden oluşan dönüştürmeye bağlı olarak değil, uygulama programlarını kullanın.

Bir MQGET çağrısına geçirilen MQGMO yapısının *Options* alanında MQGMO_CONVERT seçeneğini belirttiğinizde, uygulama verileri bir uygulama programı içinde dönüştürülür ve *tüm* deyimler doğru olduğunda:

- Kuyruktaki iletiyle ilişkilendirilmiş MQMD yapısında ayarlanan *CodedCharSetId* ya da *Encoding* alanları, MQGET çağrısında belirtilen MQMD yapısında ayarlanan *CodedCharSetId* ya da *Encoding* alanlarından farklıdır.
- İletiyle ilişkilendirilmiş MQMD yapısındaki *Format* alanı MQFMT_NONE değil.
- MQGET çağrısında belirtilen *BufferLength* sıfır değil.
- İleti verileri uzunluğu sıfır değil.
- Kuyruk yöneticisi, iletiyle ve MQGET çağrısıyla ilişkili MQMD yapılarında belirtilen *CodedCharSetId* ve *Encoding* alanları arasında dönüştürmeyi destekler. Desteklenen kodlanmış karakter kümesi tanıtıcılarının ve makine kodlamasının ayrıntıları için bkz. CodedCharSetId ve Encoding .
- Kuyruk yöneticisi, ileti biçiminin dönüştürülmesini destekler. İletiyle ilişkili MQMD yapısının *Format* alanı yerleşik biçimlerden biriyse, kuyruk yöneticisi iletiyi dönüştürebilir. *Format* yerleşik biçimlerden biri değilse, iletiyi dönüştürmek için bir veri dönüştürme çıkışı yazmanız gerekir.

Gönderen MCA verileri dönüştürmek için ise, dönüştürmenin gerekli olduğu her gönderenin ya da sunucu kanalının tanımında CONVERT (YES) anahtar sözcüğünü belirtin. Veri dönüştürme başarısız olursa, ileti gönderen kuyruk yöneticisinde DLQ ' ya gönderilir ve MQDLH yapısının *Feedback* alanı nedeni gösterir. İleti DLQ ' ya konamazsa, kanal kapanır ve dönüştürülmemiş ileti iletim kuyruğunda kalır. MCA ' ların gönderilmesi yerine, uygulamalar içinde veri dönüştürme bu durumu önler.

Kural olarak, yerleşik biçimde ya da veri dönüştürme çıkışında *karakter* verisi olarak açıklanan iletideki veriler, ileti tarafından kullanılan kodlanmış karakter kümesinden istenen değere dönüştürülür ve *sayısal* alanlar istenen kodlamaya dönüştürülür.

Yerleşik biçimleri dönüştürürken kullanılan dönüştürme işleme kurallarına ilişkin daha fazla ayrıntı ve kendi veri dönüştürme çıkışlarınızı yazmaya ilişkin bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941. Dil desteği çizelgeleri ve desteklenen makine kodlamaları hakkında bilgi için Ulusal diller ve Makine kodlamaları başlıklı konuya bakın.

EBCDIC yeni satır karakterlerinin dönüştürülmesi

EBCDIC altyapısından ASCII altyapısına gönderdiğiniz verilerin, geri aldığınız verilerle aynı olduğundan emin olmanız gerekiyorsa, EBCDIC yeni satır karakterlerinin dönüştürülmesini denetlemeniz gerekir.

Bunu, IBM MQ ' i değiştirilmemiş dönüştürme çizelgelerini kullanmaya zorlayan, platforma bağımlı bir anahtar kullanarak yapabilirsiniz, ancak bunun sonucunda ortaya çıkan tutarsız davranışın farkında olmanız gerekir.

Sorun, EBCDIC yeni satır karakterinin platformlar ya da dönüştürme çizelgeleri arasında tutarlı olarak dönüştürülmemesi nedeniyle ortaya çıkar. Sonuç olarak, veriler bir ASCII altyapısında görüntülenirse, biçimlendirme yanlış olabilir. Bu, örneğin, bir IBM i sistemini RUNMQSC kullanarak bir ASCII altyapısından uzaktan denetlemenizi zorlaştırır.

EBCDIC biçimli verilerin ASCII biçimine dönüştürülmesine ilişkin ek bilgi için Veri dönüştürme başlıklı konuya bakın.

Kuyruktaki iletilere göz atma

MQGET çağrısını kullanarak bir kuyruktaki iletilere göz atma hakkında bilgi edinmek için bu bilgileri kullanın.

Bir kuyruktaki iletilere göz atmak üzere MQGET çağrıyı kullanmak için:

1. MQOO_BROWSE seçeneğini belirterek kuyruğu göz atmak üzere açmak için MQOPEN ' i çağırın.
2. Kuyruktaki ilk iletiye göz atmak için MQGET öğesini MQGMO_BROWSE_FIRST seçeneğiyle çağırın. İstediğiniz iletiyi bulmak için, MQGET ' i MQGMO_BROWSE_NEXT seçeneğiyle çağırarak birçok iletiyi adım adım ilerletin.

   Tüm iletileri görmek için, MQMD yapısının *MsgId* ve *CorrelId* alanlarını her MQGET çağrısından sonra boş değere ayarlamalısınız.
3. Kuyruğu kapatmak için MQCLOSE ' yi çağırın.

Göz atma imleci

Göz atmak üzere bir kuyruk açtığınızda (MQOPEN), arama, göz atma seçeneklerinden birini kullanan MQGET çağrılarıyla kullanılmak üzere bir göz atma imleci oluşturur. Göz atma imlecini, kuyruktaki ilk iletiden önce konumlandırılmış mantıksal bir gösterge olarak düşünebilirsiniz.

Aynı kuyruk için birden çok MQOPEN isteği yayınlayarak (tek bir programdan) birden çok göz atma imleciniz etkin olabilir.

MQGET ' i göz atmak üzere çağırdığınızda, MQGMO yapınızda aşağıdaki seçeneklerden birini kullanın:

MQGMO_BROWSE_FIRST
MQMD yapınızda belirtilen koşulları karşılayan ilk iletinin bir kopyasını alır.

MQGMO_BROWSE_NEXT
MQMD yapınızda belirtilen koşulları karşılayan sonraki iletinin bir kopyasını alır.

MQGMO_BROWSE_MSG_UNDER_CURSOR
İmleç tarafından gösterilen iletinin bir kopyasını alır; yani, en son MQGMO_BROWSE_FIRST ya da MQGMO_BROWSE_NEXT seçeneği kullanılarak alınan iletinin bir kopyasını alır.

Her durumda, ileti kuyrukta kalır.

Bir kuyruğu açtığınızda, göz atma imleci kuyruktaki ilk iletiden hemen önce mantıksal olarak konumlandırılır. MQOPEN çağrısından hemen sonra MQGET çağrıyı yaparsanız, ilk iletiye göz atmak için MQGMO_BROWSE_NEXT seçeneğini kullanabilirsiniz; MQGMO_BROWSE_FIRST seçeneğini kullanmanız gerekmez.

İletilerin kuyruktan kopyalanacağı sıra, kuyruğun **MsgDeliverySequence** özniteliği tarafından belirlenir. (Daha fazla bilgi için bkz. "İletilerin kuyruktan alınma sırası" sayfa 741.)

FIFO sırasındaki kuyruklar (ilk giren ilk çıkar)

Bu sırada kuyruktaki ilk ileti, kuyrukta en uzun süre kalan iletidir.

Kuyruktaki iletileri sıralı olarak okumak için MQGMO_BROWSE_NEXT öğesini kullanın. Göz atarken kuyruğa konan iletileri görürsünüz; bu sırada, kuyruktaki iletiler sona yerleştirilir. İmleç kuyruğun sonuna ulaştığını fark ettiğinde, göz atma imleci bulunduğu yerde kalır ve MQRC_NO_MSG_AVAILABLE ile geri döner. Daha sonra, iletiyi orada daha fazla ileti beklerken bırakabilir ya da MQGMO_BROWSE_FIRST çağrısıyla kuyruğun başına getirebilirsiniz.

Öncelik sırasındaki kuyruklar

Bu sırada bir kuyruktaki ilk ileti, kuyrukta en uzun olan ve MQOPEN çağrısının yayınlandığı sırada en yüksek önceliğe sahip iletidir.

Kuyruktaki iletileri okumak için MQGMO_BROWSE_NEXT komutunu kullanın.

Göz atma imleci, en düşük önceliğe sahip iletiyle tamamlanacak ilk iletinin önceliğinden başlayarak sonraki iletiyi gösterir. Yürürlükteki göz atma imlecinin belirlediği iletiye eşit ya da bu iletiden daha düşük olduğu sürece, bu süre içinde kuyruğa konan iletilere göz atmanızı sağlar.

Daha yüksek önceliğe sahip kuyruğa konan iletilere yalnızca şunlar göz atabilir:

- Yeni bir göz atma imlecinin oluşturulduğu noktada, yeniden göz atma kuyruğunun açılması
- MQGMO_BROWSE_FIRST seçeneğinin kullanılması

Kesinleştirilmemiş iletiler

Kesinleştirilmemiş bir ileti göz atma işlemi tarafından hiçbir zaman görülmez; göz atma imleci bu iletiyi atlar.

İş birimi içindeki iletilere, iş birimi kesinleştirilinceye kadar göz atılamaz. İletiler kesinleştirildiğinde kuyruktaki konumlarını değiştirmez; bu nedenle, MQGMO_BROWSE_FIRST seçeneğini kullanmadığınız ve kuyrukta yeniden çalışmadığınız sürece, kesinleştirilmemiş iletiler *kesinleştirildiğinde* bile görülmez.

Kuyruk sırasına değiştir

Kuyrukta iletiler varken ileti teslim sırası öncelikten FIFO ' ya çevrilirse, kuyruğa alınmış iletilerin sırası değişmez. Kuyruğa daha sonra eklenen iletiler, kuyruğun varsayılan önceliğini alır.

Kuyruğun dizinini kullanma

Yalnızca tek bir önceliğe sahip iletileri (kalıcı ya da kalıcı olmayan ya da her ikisi) içeren dizinlenmiş bir kuyruğa göz attığınızda, kuyruk yöneticisi, belirli göz atma biçimleri kullanıldığında göz atmak için dizini kullanır.

Not: Yalnızca IBM MQ for z/OSüzerinde desteklenir.

Dizinlenmiş bir kuyruk yalnızca tek öncelikli iletiler içerdiğinde aşağıdaki göz atma biçimlerinden herhangi biri kullanılır:

1. Kuyruk MSGID ile dizinlendiyse, MQMD yapısında bir MSGID ' yi geçen isteklere göz atın, hedef iletiyi bulmak için dizin kullanılarak işlenir.
2. Kuyruk CORRELID tarafından dizinlendiyse, MQMD yapısında CORRELID geçen isteklere göz atın; hedef iletiyi bulmak için dizin kullanılarak işlenir.
3. Kuyruk GROUPID ile dizinlendiyse, MQMD yapısında GROUPID geçen isteklere göz at, hedef iletiyi bulmak için dizin kullanılarak işlenir.

Göz atma isteği MQMD yapısında bir MSGID, CORRELID ya da GROUPID geçirmezse, kuyruk dizinlenir ve bir ileti döndürülür, iletiye ilişkin dizin girişi bulunmalı ve göz atma imlecini güncellemek için bu ileti içindeki bilgiler kullanılmalıdır. Geniş bir dizin değeri seçimi kullanırsanız, bu, göz atma isteğine önemli ölçüde fazladan işlem eklemez.

İleti uzunluğu bilinmediğinde iletilere göz atılıyor

İletinin boyutunu bilmiyorsanız ve iletiyi bulmak için *MsgId*, *CorrelId*ya da *GroupId* alanlarını kullanmak istemiyorsanız, MQGMO_BROWSE_MSG_UNDER_CURSOR seçeneğini kullanabilirsiniz:

1. MQGET ' i bununla yayınla:
   - MQGMO_BROWSE_FIRST ya da MQGMO_BROWSE_NEXT seçeneği
   - MQGMO_ACCEPT_TRUNCATED_MSG seçeneği
   - Arabellek uzunluğu sıfır

   Not: Aynı iletiyi başka bir program alacaksa, MQGMO_LOCK seçeneğini de kullanmayı düşünün. MQRC_TRUNCATED_MSG_ACCEPTED döndürülmelidir.
2. Gereken saklama alanını ayırmak için döndürülen *DataLength* öğesini kullanın.
3. MQGMO_BROWSE_MSG_UNDER_CURSOR ile bir MQGET verin.

Gösterilen ileti, alınan son iletidir; göz atma imleci taşınmaz. MQGMO_LOCK seçeneğini kullanarak iletiyi kilitlemeyi ya da MQGMO_UNLOCK seçeneğini kullanarak kilitli bir iletinin kilidini açmayı seçebilirsiniz.

Kuyruk açıldıktan sonra MQGMO_BROWSE_FIRST ya da MQGMO_BROWSE_NEXT seçeneklerine sahip bir MQGET başarıyla yayınlanmamışsa çağrı başarısız olur.

Göz atdığınız bir iletiyi kaldırma
Kuyruktan, iletileri kaldırmak ve göz atmak için kuyruğu açmanız koşuluyla, önceden göz attığınız bir iletiyi kaldırabilirsiniz. (MQOPEN çağrınızda MQOO_INPUT_ * seçeneklerinden birini ve MQOO_BROWSE seçeneğini belirtmeniz gerekir.)

İletiyi kaldırmak için MQGET öğesini yeniden çağırın, ancak MQGMO yapısının *Options* alanında MQGMO_MSG_UNDER_CURSOR belirtin.Bu durumda MQGET çağrısı, MQMD yapısının *MsgId*, *CorrelId*ve *GroupId* alanlarını yoksayar.

Göz atma ve kaldırma adımlarınız arasındaki süre içinde, başka bir program, göz atma imleciniz altındaki ileti de içinde olmak üzere kuyruktan iletileri kaldırmış olabilir. Bu durumda, MQGET çağRınız, iletinin kullanılamadığını bildiren bir neden kodu döndürür.

İletilere mantıksal sırayla göz atma

“Mantıksal ve fiziksel sıralama” sayfa 742 , bir kuyruktaki iletilerin mantıksal ve fiziksel sırası arasındaki farkı açıklar. Bu ayrım, bir kuyruğa göz atarken özellikle önemlidir, çünkü genel olarak, iletiler silinmez ve göz atma işlemleri kuyruğun başından başlamaz.

Bir uygulama bir grubun çeşitli iletilerine göz atarsa (mantıksal sırayı kullanarak), bir sonraki grubun ilk iletisinden sonra bir grubun son iletisi fiziksel olarak *oluşabileceğinden* , sonraki grubun başlangıcını gerçekleştirmek için mantıksal sıranın izlenmesi önemlidir. MQGMO_LOGICAL_ORDER seçeneği, bir kuyruk taranırken mantıksal sırayı izlemesini sağlar.

Göz atma işlemleri için MQGMO_ALL_MSGS_VAR (ya da MQGMO_ALL_SEGMENTS_VAR) değerini kullanın. MQGMO_ALL_MSGS_VAR olan mantıksal iletilerin büyük/küçük harf durumunu göz önünde bulundurun. Bunun sonucu, mantıksal bir iletinin yalnızca gruptaki geri kalan tüm iletiler de varsa kullanılabilir olması

olur. Değilse, ileti iletilir. Bu, eksik mesajlar daha sonra geldiğinde, bir sonraki göz atma işlemi tarafından fark edilmedikleri anlamına gelebilir.

Örneğin, aşağıdaki mantıksal iletiler varsa,

```
   Logical message 1 (not last) of group 123
   Logical message 1 (not last) of group 456
   Logical message 2 (last)     of group 456
```

Bir göz atma işlevi MQGMO_ALL_MSGS_, kullanılabilir durumdayken, 456 grubunun ilk mantıksal iletisi döndürülür ve göz atma imleci bu mantıksal iletide bırakılır. 123 grubunun ikinci (son) iletisi şimdi gelirse:

```
   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
```

ve aynı sonraki göz atma işlevi yayınlandı, bu grubun ilk iletisi göz atma imlecinden *önce* olduğu için 123 grubunun tamamlandığı fark edilmedi.

Bazı durumlarda (örneğin, grup bütünlüğü içinde olduğunda iletiler yıkıcı bir şekilde alınırsa), MQGMO_ALL_MSGS_FIRST ile birlikte MQGMO_BROWSE_FIRST komutunu kullanabilirsiniz. Aksi takdirde, kaçırılan yeni gelen iletileri not almak için göz atma taramasını yinelemeniz gerekir; yalnızca MQGMO_WAIT komutunu MQGMO_BROWSE_NEXT ile birlikte vermeniz ve MQGMO_ALL_MSGS_MEVCUT iletileri dikkate almamanız gerekir. (Bu, iletiler tarandıktan sonra gelebilecek yüksek öncelikli iletilere de olur.)

Sonraki bölümler, bölümlenmemiş iletilerle ilgili tarama örneklerini inceler; bölümlenmiş iletiler benzer ilkeleri izler.

Gruplar halinde iletilere göz atma
Bu örnekte uygulama, kuyruktaki her iletiye mantıksal sırayla göz atmaktadır.

Kuyruktaki iletiler gruplanmış olabilir. Gruplanmış iletiler için uygulama, içindeki tüm iletiler gelene kadar herhangi bir grubu işlemeye başlamak istemez. Bu nedenle, gruptaki ilk ileti için MQGMO_ALL_MSGS_KULLANILABILIR belirtildi; gruptaki sonraki iletiler için bu seçenek gereksiz.

Bu örnekte MQGMO_WAIT kullanılır. Ancak, yeni bir grup gelirse bekleme yerine getirilebilirse de, “İletilere mantıksal sırayla göz atma” sayfa 772içindeki nedenlerden ötürü, göz atma imlecinin bir gruptaki ilk mantıksal iletiyi önceden geçirmesi ve geri kalan iletilerin şimdi gelmesi yeterli olmaz. Bununla birlikte, uygun bir zaman aralığı beklemek, uygulamanın yeni ileti ya da bölüm beklerken sürekli döngü yapmamasını sağlar.

MQGMO_LOGICAL_ORDER, taramanın mantıksal sırada olduğundan emin olmak için tüm süreç boyunca kullanılır. Bu, her bir grubun kaldırıldığı için, bir gruptaki ilk (ya da tek) iletiyi ararken MQGMO_LOGICAL_ORDER ' in kullanılmadığı yıkıcı MQGET örneğine karşılık gelir.

İletinin bölümlenmiş olsun ya da olmasın, uygulamanın arabelleğinin her zaman iletinin tamamını tutacak kadar büyük olduğu varsayılır. Bu nedenle, her MQGET için MQGMO_COMPLETE_MSG belirtildi.

Aşağıda, bir gruptaki mantıksal iletilere göz atma örneği verilmiştir:

```
/* 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 */
   ...
```

Grup, MQRC_NO_MSG_AVAILABLE döndürülünceye kadar yinelenir.

Yıkıcı bir şekilde göz atma ve alma
Bu örnekte, uygulama bir grup içindeki mantıksal iletilerin her birine göz atarken, o grubun yıkıcı bir şekilde alınıp alınmayacağına karar veriyor.

Bu örneğin ilk kısmı öncekine benzer. Bununla birlikte, bu durumda, bütün bir gruba göz attık, geri dönmeye ve onu yıkıcı bir şekilde geri almaya karar verdik.

Bu örnekte her grup kaldırıldığından, MQGMO_LOGICAL_ORDER, bir gruptaki ilk ya da tek ileti ararken kullanılmaz.

Aşağıda, göz atma ve yıkıcı bir şekilde alma örneği verilmiştir:

```
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 */
...
```

Göz atılan iletilerin tekrarlanan teslimini önleme
Belirli açık seçenekleri ve get-message seçeneklerini kullanarak, iletileri, yürürlükteki ya da diğer işbirliği yapan uygulamalar tarafından yeniden alınmaması için göz atılmış olarak işaretleyebilirsiniz. İletilerin yeniden göz atılabilmesi için açık ya da otomatik olarak işareti kaldırılabilir.

Bir kuyruktaki iletilere göz atarken, iletileri yıkıcı bir şekilde almanız durumunda alma sırasına göre farklı bir sırayla alabilirsiniz. Özellikle, aynı iletiye birden çok kez göz atabilirsiniz; bu, kuyruktan kaldırıldığında mümkün değildir. Bunu önlemek için, iletilere göz atılırken iletileri *işaretleyebilir* ve işaretli iletileri almaktan kaçınabilirsiniz. Buna bazen *işaretle göz at*denir. Göz atılan iletileri işaretlemek için, MQGMO_MARK_BROWSE_HANDLE ileti alma seçeneğini kullanın ve yalnızca işaretli olmayan iletileri almak için MQGMO_UNMARKED_BROWSE_MSG kullanın. MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG ve MQGMO_MARK_BROWSE_HANDLE seçeneklerinin birleşimini kullanır ve yinelenen MQGET' leri yayınlarsanız, kuyruktaki her iletiyi sırayla alırsınız. Bu, iletilerin atlanmamasını sağlamak için MQGMO_BROWSE_FIRST kullanılsa da iletilerin sürekli olarak gönderilmesini önler. Bu seçenek birleşimi tek bir MQGMO_BROWSE_HANDLE değişmeziyle gösterilebilir. Kuyrukta göz atılmamış ileti yoksa, MQRC_NO_MSG_AVAILABLE döndürülür.

Aynı kuyruğa birden çok uygulama göz atıyorsa, kuyruk MQOO_CO_OP ve MQOO_BROWSE seçenekleriyle açılabilir. Her MQOPEN tarafından döndürülen nesne tanıtıcısı, bir işbirliği grubunun parçası olarak kabul

edilir. MQGMO_MARK_BROWSE_CO_OP seçeneğini belirten bir MQGET çağrısının döndürdüğü her iletinin, bu birlikte çalışan tanıtıcı kümesi için işaretlenmiş olduğu varsayılır.

Bir ileti bir süre imlendiyse, kuyruk yöneticisi tarafından otomatik olarak işareti kaldırılabilir ve yeniden göz atılabilir. Kuyruk yöneticisi özniteliği MsgMarkBrowseInterval , bir iletinin işbirliği yapan tanıtıcı kümesi için imlenmiş olarak kalacağı süreyi milisaniye cinsinden verir. MsgMarkBrowseInterval (-1), iletilerin hiçbir zaman otomatik olarak işaretinin kaldırılamayacağını gösterir.

İletileri işaretleyen tek bir işlem ya da işbirliği işlemi kümesi durduğunda, işaretli iletiler işaretlenmemiş olur.

İşbirliğine göz atma örnekleri

Bir kuyruktaki iletilere göz atmak ve her iletinin içeriğine dayalı olarak bir tüketici başlatmak için bir dağıtıcı uygulamasının birden çok kopyasını çalıştırabilirsiniz. Her dağıtıcıda, kuyruğu MQOO_CO_OP ile açın. Bu, dağıtıcıların işbirliği yaptıklarını ve birbirlerinin işaretli iletilerini bildiklerini gösterir. Daha sonra her dağıtıcı, MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG ve MQGMO_MARK_BROWSE_CO_OP seçeneklerini belirterek yinelenen MQGET çağrıları yapar (bu seçenek birleşimini göstermek için tek bir MQGMO_BROWSE_CO_OP değişmezini kullanabilirsiniz). Daha sonra her dağıtıcı uygulaması, yalnızca diğer işbirliği yapan dağıtıcılar tarafından işaretlenmemiş iletileri alır. Dağıtıcı bir tüketiciyi başlatır ve MQGET tarafından döndürülen MsgToken ' ı tüketiciye iletir; bu da iletiyi kuyruktan yıkıcı bir şekilde alır. Tüketici iletinin MQGET iletisini geri çevirirse, ileti artık imlenmediği için tarayıcılardan birinin yeniden dağıtılması için kullanılabilir. Tüketici iletide MQGET işlemi yapmazsa, MsgMarkBrowseInterval (İleti İşareti) geçtikten sonra kuyruk yöneticisi, işbirliği yapan tanıtıcı kümesi için iletiyi işaretlemez ve yeniden dağıtılabilir.

Aynı dağıtıcı uygulamasının birden çok kopyası yerine, her biri kuyruktaki iletilerin bir alt kümesini işlemek için uygun olan, kuyruğa göz atan farklı dağıtıcı uygulamalarınız olabilir. Her dağıtıcıda, kuyruğu MQOO_CO_OP ile açın. Bu, dağıtıcıların işbirliği yaptıklarını ve birbirlerinin işaretli iletilerini bildiklerini gösterir.

- Tek bir dağıtıcı için ileti işleme sırası önemliyse, her dağıtıcı MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG ve MQGMO_MARK_BROWSE_HANDLE (ya da MQGMO_BROWSE_HANDLE) seçeneklerini belirterek yinelenen MQGET çağrıları yapar. Göz atılan ileti bu dağıtıcının işlenmesi için uygunsa, MQMO_MATCH_MSG_TOKEN, MQGMO_MARK_BROWSE_CO_OP ve önceki MQGET çağrısıyla döndürülen MsgToken ile bir MQGET çağrısı yapar. Çağrı başarılı olursa, dağıtıcı tüketiciyi kullanıma hazırlar ve MsgToken ' ı iletir.
- İleti işleme sırası önemli değilse ve dağıtıcının karşılaştığı iletilerin çoğunu işlemesi bekleniyorsa, MQGMO_BROWSE_FIRST, MQGMO_UNMARKED_BROWSE_MSG ve MQGMO_MARK_BROWSE_CO_OP (ya da MQGMO_BROWSE_CO_OP) seçeneklerini kullanın. Dağıtıcı işlemediği bir iletiye göz atarsa, MQGET seçeneğini MQMO_MATCH_MSG_TOKEN, MQGMO_UNMARK_BROWSE_CO_OP seçeneğiyle ve MsgToken daha önce döndürülerek iletiyi çözer.

MQGET çağrılarının başarısız olduğu bazı durumlar

Bir MQOPEN ve MQGET çağrısı arasında bir komutta FORCE seçeneği kullanılarak bir kuyruğun belirli öznitelikleri değiştirilirse, MQGET çağrısı başarısız olur ve MQRC_OBJECT_CHANGED neden kodunu döndürür.

Kuyruk yöneticisi, nesne tanıtıcısını artık geçerli değil olarak işaretler. Bu durum, değişiklikler kuyruk adının çözüldüğü herhangi bir kuyruğa uygulandıysa da oluşur. Tanıtıcıyı bu şekilde etkileyen öznitelikler, MQOPENiçinde MQOPEN çağrısının tanımında listelenir. Çağrınız MQRC_OBJECT_CHANGED neden kodunu döndürürse, kuyruğu kapatın, yeniden açın ve iletiyi yeniden almaya çalışın.

İleti alma girişiminde bulunduğunuz bir kuyruk (ya da kuyruk adının çözüldüğü herhangi bir kuyruk) için alma işlemleri engellenirse, MQGET çağrısı başarısız olur ve MQRC_GET_INENGELLENEN neden kodunu döndürür. Göz atmak için MQGET çağrısını kullanıyor olsanız bile bu oluşur. Daha sonra MQGET çağrısına çalışırsanız, uygulamanın tasarımı diğer programların kuyrukların özniteliklerini düzenli olarak değiştirmesini gerektirecek şekilde başarılı bir şekilde ileti alabilir.

Dinamik bir kuyruk (geçici ya da kalıcı) silindiyse, önceden edinilen bir nesne tanıtıcısını kullanan MQGET çağrıları başarısız olur ve MQRC_Q_DELETED neden kodunu döndürür.

Yayınlama/abone olma uygulamaları yazılıyor

Yayınlama/abone olma IBM MQ uygulamaları yazmaya başlayın.

Yayınlama/abone olma kavramlarına genel bakış için bkz. İleti alışverişi yayınlama/abone olma.

Farklı tipte yayınlama/abone olma uygulamaları yazmaya ilişkin bilgi için aşağıdaki konulara bakın:

- "Yayınlayıcı uygulamaları yazılıyor" sayfa 776
- "Abone uygulamaları yazılıyor" sayfa 783
- "Yaşam çevrimlerini yayınla/abone ol" sayfa 799
- "İleti özelliklerini yayınla/abone ol" sayfa 804
- "İleti sıralaması" sayfa 805
- "Yayınları engelleme" sayfa 805
- "Yayın seçenekleri" sayfa 813
- "Abonelik seçenekleri" sayfa 813

İlgili kavramlar

"Uygulama geliştirme kavramları" sayfa 6
IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

"IBM MQ için uygulama geliştirilmesi" sayfa 5
İleti göndermek ve almak için uygulamalar geliştirebilir ve kuyruk yöneticilerinizi ve ilgili kaynakları yönetebilirsiniz. IBM MQ , birçok farklı dilde ve çerçevede yazılmış uygulamaları destekler.

"IBM MQ uygulamaları için tasarımla ilgili önemli noktalar" sayfa 47
Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğine karar verdiğinizde, IBM MQtarafından sunulan özellikleri nasıl kullanacağınıza karar vermeniz gerekir.

"Kuyruğa alma için yordam uygulaması yazılması" sayfa 692
Kuyruğa alma uygulamaları yazma, bir kuyruk yöneticisine bağlanma ve bağlantı kesme, yayınlama/abone olma ve nesneleri açma ve kapatma hakkında bilgi edinmek için bu bilgileri kullanın.

"İstemci yordamsal uygulamaları yazılıyor" sayfa 875
IBM MQ üzerinde yordam dili kullanarak istemci uygulamaları yazmak için bilmeniz gerekenleri.

"Yordamsal uygulama oluşturma" sayfa 957
Birkaç yordam dilinden birinde bir IBM MQ uygulaması yazabilir ve uygulamayı birkaç farklı platformda çalıştırabilirsiniz.

"Yordamsal program hatalarının işlenmesi" sayfa 994
Bu bilgiler, bir çağrı yaptığında ya da iletisi son hedefine teslim edildiğinde, uygulamalarınızla ilişkili MQI çağrılarıyla ilgili hataları açıklar.

İlgili görevler

"IBM MQ örnek yordam programlarının kullanılması" sayfa 1013
Bu örnek programlar yordamsal dillerde yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir. Farklı platformlarda IBM MQ programları.

Yayınlayıcı uygulamaları yazılıyor

İki örneği inceleyerek yayıncı uygulamaları yazmaya başlayın. İlki, iletileri kuyruğa koyan bir noktadan noktaya uygulamada olabildiğince yakından modellenir ve ikincisi, yayınlayıcı uygulamaları için daha yaygın bir örüntü olan konuların dinamik olarak yaratılmasını gösterir.

Basit bir IBM MQ yayınlayıcı uygulaması yazmak, iletileri kuyruğa koyan bir IBM MQ noktadan noktaya uygulaması yazmak gibidir (Çizelge 121 sayfa 777). Fark, bir kuyruğa değil, bir konuya MQPUT iletileri göndermeniz.

Çizelge 121. Yayınlama/abone olma IBM MQ program örüntüsüne karşı noktadan noktaya.		
Adım	**MQ çağrısına işaret edin**	**MQ Çağrısı Yayınla**
Kuyruk yöneticisine bağlan	MQCONN	MQCONN
Kuyruğu aç	MQOPEN	
Konuyu aç		MQOPEN
Koyma iletileri	MQPUT	MQPUT
Konuyu kapat		MQCLOSE
Kuyruğu kapat	MQCLOSE	
Kuyruk yöneticisiyle bağlantıyı kes	MQDISC	MQDISC

Bu betonu yapmak için, hisse senedi fiyatlarını yayınlamak için iki uygulama örneği bulunmaktadır. İletileri kuyruğa yerleştirmede yakından modellenen ilk örnekte ("Örnek 1: Sabit bir konuya yayınlayıcı" sayfa 777), denetimci kuyruk yaratmaya benzer bir şekilde bir konu tanımlaması yaratır. Programcı, iletileri bir kuyruğa yazmak yerine konuya yazmak için MQPUT kodlar. İkinci örnekte ("Örnek 2: Bir değişken konusuna yayınlayıcı" sayfa 780), programın IBM MQ ile etkileşim kalıbı benzerdir. Fark, programcının yönetici yerine iletinin yazılacağı konuyu sağlamasıdır. Pratikte bu, genellikle konu dizgisinin bir tarayıcı aracılığıyla insan girişi gibi başka bir kaynak tarafından tanımlanan ya da sağlanan içerik olduğu anlamına gelir.

İlgili kavramlar

"Abone uygulamaları yazılıyor" sayfa 783
Üç örneği inceleyerek abone uygulamaları yazmaya başlayın: IBM MQ uygulaması kuyruktan iletileri tüketen, abonelik oluşturan ve kuyruğa alma bilgisi gerektirmeyen bir uygulama ve son olarak hem kuyruğa alma hem de abonelikleri kullanan bir örnek.

İlgili başvurular

KONUYU TANıMLAYıN
DISPLAYTOPIC
DISPLAYTPSTATUS

Örnek 1: Sabit bir konuya yayınlayıcı

Yönetimsel olarak tanımlanmış bir konuda yayınlamayı gösteren bir IBM MQ programı.

Not: Kompakt kodlama stili, üretim kullanımı yerine okunabilirlik için tasarlanmıştır.

Şekil 65 sayfa 778 içindeki çıkışa bakın

```
#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);
}
```

Şekil 64. Sabit bir konuya ilişkin basit IBM MQ yayınlayıcısı.

```
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
```

Şekil 65. İlk yayınlayıcı örneğinden örnek çıktı

Aşağıdaki seçilen kod satırları, IBM MQiçin bir yayınlayıcı uygulaması yazmanın çeşitli yönlerini göstermektedir.

char topicNameDefault[] = "IBMSTOCKPRICE";
Programda varsayılan bir konu adı tanımlandı. Programın ilk bağımsız değişkeni olarak farklı bir konu nesnesinin adını belirterek bunu geçersiz kılabilirsiniz.

MQCHAR resTopicStr[151];
`resTopicStr` , `td.ResObjectString.VSPtr` tarafından işaret edilir ve `MQOPEN` tarafından çözülen konu dizgisini döndürmek için kullanılır. Boş sonlandırma için yer açmak üzere `resTopicStr` geçirilen uzunluktan büyük bir uzunluk `td.ResObjectString.VSBufSize` değerini belirleyin.

memset (resTopicStr, 0, sizeof(resTopicStr));
Bir `MQCHARV` içinde döndürülen çözülmüş konu dizgisinin boş (null) sonlandırıldığından emin olmak için `resTopicStr` ' i kullanıma hazırlayın.

td.ObjectType = MQOT_TOPIC
Yayınlama/abone olma için yeni bir nesne tipi vardır: *konu nesnesi*.

td.Version = MQOD_VERSION_4;
Yeni nesne tipini kullanmak için, nesne tanımlayıcısının en az *sürüm 4* ' ünü kullanmanız gerekir.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
`topicName` , bazen yönetim konusu nesnesi olarak adlandırılan bir konu nesnesinin adıdır. Örnekte, konu nesnesinin IBM MQ Explorer ya da bu MQSC komutu kullanılarak önceden yaratılması gerekir.

```
DEFINE TOPIC(IBMSTOCKPRICE) TOPICSTR(NYSE/IBM/PRICE) REPLACE;
```

td.ResObjectString.VSPtr = resTopicStr;
Çözülen konu dizgisi, programdaki son `printf` içinde yankılanır. Çözümlenen dizgiyi programa geri döndürmek için IBM MQ için `MQCHARV ResObjectString` yapısını ayarlayın.

MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
Çıkış için konuyu açın; çıkış için bir kuyruk açmak gibi.

pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
Yeni abonelerin yayını alabilmesini ve yayıncıda `MQPMO_RETAIN` belirtilerek, bir aboneyi başlattığınızda, abonenin ilk eşleşen yayını olarak, abone başlamadan önce yayınlanan en son yayını alır. Bunun alternatifi, abonelere yalnızca abone başlatıldıktan sonra yayınlanan yayınları sağlamaktır. Ayrıca bir abone, aboneliğinde `MQSO_NEW_PUBLICATIONS_ONLY` belirterek alıkonan bir yayını almayı reddetme seçeneğine sahiptir.

MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
İleti arabelleğinin bir parçası olarak boş sonlandırma karakterini IBM MQ 'e geçirmek için `MQPUT` ' e iletilen dizginin uzunluğuna 1 ekleyin.

İlk örnek neyi gösteriyor? Bu örnek, IBM MQ programlarını yazmak için denenmiş ve test edilmiş geleneksel kalıbı mümkün olduğunca yakından taklit eder. IBM MQ programlama örüntüsünün önemli bir özelliği, programcının iletilerin gönderildiği yerde endişeli olmamasıdır. Programcının görevi, bir kuyruk yöneticisine bağlanmak ve alıcılara dağıtılacak iletileri aktarmaktır. Noktadan noktaya iletişim paradigmasında programcı, denetimcinin yapılandırdığı bir kuyruğu (büyük olasılıkla bir diğer ad kuyruğu) açar. Diğer ad kuyruğu, iletileri yerel kuyruk yöneticisinde ya da uzak kuyruk yöneticisinde hedef kuyruğa yönlendirir. İletiler teslim edilmeyi beklerken, kaynak ve hedef arasında bir yerde kuyruklarda depolanır.

Yayınlama/abone olma örüntüsünde, bir kuyruk açmak yerine programcı bir konu açar. Örneğimizde, konu bir yönetici tarafından bir konu dizgisiyle ilişkilendirilir. Kuyruk yöneticisi, kuyrukları kullanarak yayını, yayının konu dizgisiyle eşleşen abonelikleri olan yerel ya da uzak abonelere iletir. Yayınlar alıkonursa, artık abonesi olmasa da, kuyruk yöneticisi yayının en son kopyasını saklar. Alıkonan yayın, gelecekteki abonelere iletilmek üzere kullanılabilir. Yayınlayıcı uygulaması, yayını seçmede ya da bir hedefe yönlendirmede rol oynamaz; görevi, yayınları oluşturmaktır ve yönetici tarafından tanımlanan konulara yerleştirmektir.

Bu sabit konu örneği, birçok yayınlama/abone olma uygulaması için alışılmadık bir örnektir: statiktir. Bir yöneticinin konu dizgilerini tanımlamasını ve yayınlanan konuları değiştirmesini gerektirir. Genellikle yayınlama/abone olma uygulamalarının konu ağacının bir kısmını ya da tümünü bilmeleri gerekir. Konular

sık sık değişebilir ya da konular çok fazla değişmese de, konu birleşimlerinin sayısı büyüktür ve bir yöneticinin yayınlanması gerekebilecek her konu dizgisi için bir konu düğümü tanımlaması çok fazladır. Konu dizgileri yayından önce bilinmeyebilir; bir yayınlayıcı uygulaması, konu dizgisi belirtmek için yayın içeriğindeki bilgileri kullanabilir ya da tarayıcıdan insan girişi gibi başka bir kaynaktan yayınlanacak konu dizgileriyle ilgili bilgilere sahip olabilir. Daha dinamik yayınlama stillerine hitap etmek için sonraki örnek, yayınlayıcı uygulamasının bir parçası olarak konuların dinamik olarak nasıl oluşturulacağını gösterir.

Konular, yayıncıları ve aboneleri bir araya getirmektedir. Konuların adlandırılması ve bunların konu ağaçlarında düzenlenmesi için kuralların veya mimarinin tasarlanması, bir yayınlama/abone olma çözümü geliştirilmesinde önemli bir adımdır. Konu ağacının kuruluşunun yayıncı ve abone programlarını birbirine bağladığı kapsama dikkatle bakın ve bunları konu ağacının içeriğine bağlar. Konu ağacındaki değişikliklerin yayınlayıcı ve abone uygulamalarını etkileyip etkilemediğini ve etkiyi nasıl en aza indirebileceğinizi kendinize sorun. IBM MQ yayınlama/abone olma modelinin mimarisi, bir konunun kök kısmını ya da kök alt ağacını sağlayan bir yönetim konusu nesnesi kavramıdır. Konu nesnesi, uygulama programlama ve işlemlerini basitleştiren ve sonuç olarak sürdürülebilirliği geliştiren, konu ağacının kök kısmını yönetim olarak tanımlama seçeneği sunar. Örneğin, yalıtılmış konu ağaçları olan birden çok yayınlama/abone olma uygulamasını dağıtıyorsanız, konu ağacının kök kısmını yönetimsel olarak tanımlayarak, farklı uygulamalar tarafından benimsenen konu adlandırma kurallarında tutarlılık olmasa bile, konu ağaçlarının yalıtılmasını garanti edebilirsiniz.

Pratikte, yayıncı uygulamaları, bu örnekte olduğu gibi, yalnızca sabit konuları ve bir sonraki örnekte olduğu gibi değişken konuları kullanarak bir spektrumu kapsar. “Örnek 2: Bir değişken konusuna yayınlayıcı” sayfa 780 , konu ve konu dizgilerinin kullanımını da gösterir.

İlgili kavramlar

“Örnek 2: Bir değişken konusuna yayınlayıcı” sayfa 780
Programsal olarak tanımlanmış bir konuda yayınlama işlemini gösteren bir WebSphere MQ programı.

“Abone uygulamaları yazılıyor” sayfa 783
Üç örneği inceleyerek abone uygulamaları yazmaya başlayın: IBM MQ uygulaması kuyruktan iletileri tüketen, abonelik oluşturan ve kuyruğa alma bilgisi gerektirmeyen bir uygulama ve son olarak hem kuyruğa alma hem de abonelikleri kullanan bir örnek.

Örnek 2: Bir değişken konusuna yayınlayıcı
Programsal olarak tanımlanmış bir konuda yayınlama işlemini gösteren bir WebSphere MQ programı.

Not: Kompakt kodlama stili, üretim kullanımı yerine okunabilirlik için tasarlanmıştır.

Şekil 67 sayfa 781içindeki çıkışa bakın.

```
#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);
}
```

Şekil 66. Değişken bir konuya ilişkin yalın IBM MQ yayınlayıcısı.

```
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
```

Şekil 67. İkinci yayınlayıcı örneğinden örnek çıktı

Bu örnek hakkında not almanız gereken birkaç nokta vardır.

char topicNameDefault[] = "STOCKS";
Varsayılan konu adı STOCKS , konu dizgisinin bir kısmını tanımlar. Bu konu adını programa ilk bağımsız değişken olarak sağlayarak geçersiz kılabilir ya da ilk parametre olarak / belirterek konu adının kullanımını ortadan kaldırabilirsiniz.

char topicString[101] = "IBM/PRICE";
IBM/PRICE , varsayılan konu dizgisidir. Bu konu dizgisini, programın ikinci bağımsız değişkeni olarak sağlayarak geçersiz kılabilirsiniz.

Kuyruk yöneticisi, STOCKS konu nesnesi "NYSE"tarafından sağlanan konu dizgisini, "IBM/PRICE" programı tarafından sağlanan konu dizgisiyle birleştirir ve iki konu dizgisinin arasına bir "/" ekler. Sonuç, çözümlenen "NYSE/IBM/PRICE"konu dizgisidir. Sonuçtaki konu dizgisi, IBMSTOCKPRICE konu nesnesinde tanımlananla aynıdır ve tam olarak aynı etkiye sahiptir.

Çözülen konu dizgisiyle ilişkilendirilmiş denetim konusu nesnesinin, yayınlayıcı tarafından MQOPEN ' e geçirilen konu nesnesiyle aynı olması gerekmez. IBM MQ , hangi denetim konusu nesnesinin yayınla ilişkili öznitelikleri tanımladığını belirlemek için, çözülen konu dizgisindeki örtük ağacı kullanır.

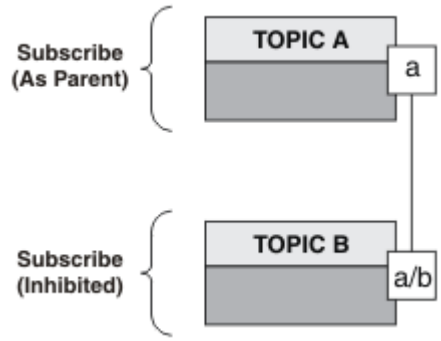
İki konu nesnesi A ve Bolduğunu ve A 'in "a"konusunu tanımladığını ve B ' in "a/b" (Şekil 68 sayfa 782) konusunu tanımladığını varsayalım. Yayınlayıcı programı A konu nesnesine gönderme yapıyorsa ve konu dizgisini "b", konuyu "a/b"konu dizgisine çözüyorsa, konu, Biçin tanımlanan "a/b" konu dizgisiyle eşleştiği için, yayının özelliklerini B konu nesnesinden devralır.

if (strcmp(argv[1],"/"))
argv[1] isteğe bağlı olarak sağlanan topicName' dir. "/" konu adı olarak geçersizdir; burada konu adı olmadığını ve konu dizgisinin tümüyle program tarafından sağlandığını belirtir. Şekil 67 sayfa 781 içindeki çıktı, program tarafından dinamik olarak sağlanan tüm konu dizgisini gösterir.

strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
Varsayılan durum için, IBM MQ Explorer ya da bu MQSC komutu kullanılarak isteğe bağlı topicName önceden yaratılmalıdır:

```
DEFINE TOPIC(STOCKS) TOPICSTR(NYSE) REPLACE;
```

td.ObjectString.VSPtr = topicString;
Konu dizgisi, konu tanımlayıcısındaki bir MQCHARV alanıdır



Şekil 68. Konu nesnesi ilişkilendirmeleri

İkinci örnek neyi gösteriyor? Kod ilk örneğe çok benzer olmasına rağmen-etkili bir şekilde sadece iki satır farkı vardır-sonuç ilkine göre önemli ölçüde farklı bir programdır. Programcı, yayınların gönderileceği hedefleri denetler. Abone uygulamalarını tasarlamak için kullanılan minimum yönetici girişiyle birlikte, yayınları yayıncılardan abonelere yönlendirmek için herhangi bir konu veya kuyruk önceden tanımlanmasına gerek yoktur.

Noktadan noktaya ileti sistemi paradigmasında, iletiler akmadan önce kuyruklar tanımlanmalıdır. Yayınlama/abone olma için, IBM MQ altta yatan kuyruk sistemini kullanarak yayınlama/abone olma

özelliğini uygulasa da, ileti sistemi ve kuyruğa alma ile ilişkili garantili teslim, işlemsellik ve gevşek bağlaşımın avantajları yayınlama/abone olma uygulamaları tarafından devralınır.

Bir tasarımcı, yayıncının ve abonenin, programların temel konu ağacını bilip bilmediğine ve ayrıca abone programlarının kuyruğa alınmayı bilip bilmediğine karar vermeli. Bundan sonra abone örnek uygulamalarını araştırın. Bunlar, genellikle `NYSE/IBM/PRICE`yayınlayan ve abone olan yayıncı örnekleriyle birlikte kullanılmak üzere tasarlanmıştır.

İlgili kavramlar

"Örnek 1: Sabit bir konuya yayınlayıcı" sayfa 777
Yönetimsel olarak tanımlanmış bir konuda yayınlamayı gösteren bir IBM MQ programı.

"Abone uygulamaları yazılıyor" sayfa 783
Üç örneği inceleyerek abone uygulamaları yazmaya başlayın: IBM MQ uygulaması kuyruktan iletileri tüketen, abonelik oluşturan ve kuyruğa alma bilgisi gerektirmeyen bir uygulama ve son olarak hem kuyruğa alma hem de abonelikleri kullanan bir örnek.

Abone uygulamaları yazılıyor

Üç örneği inceleyerek abone uygulamaları yazmaya başlayın: IBM MQ uygulaması kuyruktan iletileri tüketen, abonelik oluşturan ve kuyruğa alma bilgisi gerektirmeyen bir uygulama ve son olarak hem kuyruğa alma hem de abonelikleri kullanan bir örnek.

Çizelge 122 sayfa 783 içinde tüketici ya da abonenin üç stili, onları karakterize eden IBM MQ işlev çağrılarının sıralarıyla birlikte listelenir.

1. İlk stil, MQ Yayın Tüketicisi, yalnızca MQGET işlemini yapan bir noktadan noktaya MQ programıyla aynıdır. Uygulamanın yayınları tükettiği konusunda hiçbir bilgisi yoktur; bu yalnızca kuyruktan ileti okumaktır. Yayınların kuyruğa yöneltilmesine neden olan abonelik, yönetimsel olarak IBM MQ Explorer ya da bir komut kullanılarak oluşturulur.
2. İkinci stil, çoğu abone uygulaması için tercih edilen kalıptır. Abone uygulaması aboneliği oluşturur ve yayınları alır. Kuyruk yönetiminin tümü kuyruk yöneticisi tarafından gerçekleştirilir. Bu, *yönetilen abone*olarak bilinir.
3. Üçüncü stilde, abone uygulaması yayınları tutmak, bu kuyruğu açmak ve kapatmak ve kuyruğu yayınlarla doldurmak için abonelikler vermek için kullanılacak kuyruğu belirlemekten sorumludur. Bu, *yönetilmeyen abone*olarak bilinir.

Bu stilleri anlamanın bir yolu, her stil için Çizelge 122 sayfa 783 içinde listelenen örnek C programlarını incelemektir. Örnekler, "Yayınlayıcı uygulamaları yazılıyor" sayfa 776içinde bulunan yayıncı örneğiyle birlikte çalıştırılacak şekilde tasarlanmıştır.

Çizelge 122. Noktadan noktaya ve abone olun IBM MQ program kalıpları.

Adım	MQ ileti tüketicisi	"Örnek 1: MQ Yayın tüketicisi" sayfa 784	"Örnek 2: Yönetilen MQ abonesi" sayfa 786	"Örnek 3: Yönetilmeyen MQ abonesi" sayfa 791
Kuyruk yöneticisine bağlan	MQCONN	MQCONN	MQCONN	MQCONN
Kuyruğu aç	MQOPEN	MQOPEN		MQOPEN
Abone olun			MQSUB	MQSUB
İletileri al	MQGet	MQGet	MQGet	MQGet
Kuyruğu kapat	MQCLOSE	MQCLOSE	(MQCLOSE)	MQCLOSE
Aboneliği kapat			MQCLOSE	MQCLOSE
Kuyruk yöneticisiyle bağlantıyı kes	MQDISC	MQDISC	MQDISC	MQDISC

MQCLOSE kullanımı her zaman isteğe bağlıdır; kaynakları serbest bırakmak, MQCLOSE seçeneklerini geçirmek ya da yalnızca MQOPEN ile simetri için. Abonelik kuyruğu Yönetilen MQ abonesinde kapatıldığında ve simetri bağımsız değişkeni ilgili olmadığında MQCLOSE seçeneklerini belirtmeniz gerekmediği için, abonelik kuyruğu Örnek 2: Yönetilen MQ abonesiiçinde açık bir şekilde kapatılmamıştır.

Yayınlama/abone olma uygulama kalıplarını anlamanın başka bir yolu da, ilgili farklı varlıklar arasındaki etkileşimlere çok fazla bakmaktır. Yaşam çizgisi veya UML dizi şemaları etkileşimleri incelemek için iyi bir yoldur. Üç ömür çizgisi örneği “Yaşam çevrimlerini yayınla/abone ol” sayfa 799içinde açıklanmıştır.

Örnek 1: MQ Yayın tüketicisi
MQ Yayın tüketicisi, konuların kendisine abone olmayan bir IBM MQ ileti tüketicisidir.

Bu örnek için abonelik ve yayın kuyruğu yaratmak üzere aşağıdaki komutları çalıştırın ya da IBM MQ Explorer 'ı kullanarak nesneleri tanımlayın.

```
DEFINE QLOCAL(STOCKTICKER) REPLACE;
DEFINE SUB(IBMSTOCKPRICESUB) DEST(STOCKTICKER) TOPICOBJ(IBMSTOCKPRICE) REPLACE;
```

IBMSTOCKPRICESUB aboneliği, yayınlayıcı örneği ve yerel kuyruk STOCKTICKERiçin yaratılan IBMSTOCK konu nesnesine gönderme yapar. IBMSTOCK konu nesnesi, abonelikte kullanılan konu dizgisini (NYSE/IBM/PRICE) tanımlar. Abonelik yaratılmadan önce, konu nesnesinin ve yayınları almak için kullanılan kuyruğun tanımlanması gerektiğini unutmayın.

MQ yayın tüketicisi örüntüsüne ilişkin çok sayıda değerli kategori vardır:

1. Multiprocessing: Yayınları okuma çalışmasının dışında paylaşım. Yayınların tümü, abonelik konusuyla ilişkili tek kuyruğa gider. Birden çok tüketici MQOO_INPUT_SHAREDkomutunu kullanarak kuyruğu açabilir.
2. Merkezi olarak yönetilen abonelikler. Uygulamalar kendi abonelik konularını veya aboneliklerini oluşturmaz; yayınların gönderildiği yerden yönetici sorumludur.
3. Abonelik konsantrasyonu: Tek bir kuyruğa birden çok farklı abonelik gönderilebilir.
4. Abonelik dayanıklılığı: Kuyruk, tüketiciler etkin olsun ya da olmasın tüm yayınları alır.
5. Geçiş ve birlikte bulunma: tüketici kodu, bir noktadan noktaya iletişim ve bir yayınlama/abone olma senaryosu için eşit derecede iyi çalışır.

Abonelik, konu dizgisi NYSE/IBM/PRICE ile kuyruk STOCKTICKERarasında bir ilişki yaratır. Güncel olarak tutulan yayınlar da dahil olmak üzere, abonelik oluşturulduğu andan itibaren STOCKTICKER ' e iletilir.

Yönetimle oluşturulan bir abonelik yönetilebilir ya da yönetilemez. Yönetilen abonelik, tıpkı yönetilmeyen bir abonelik gibi, oluşturulduğu anda yürürlüğe girer. Yönetilen abonelik için tüm kalıp kategorileri kullanılamaz. Bkz. “Örnek 3: Yönetilmeyen MQ abonesi” sayfa 791

Not: Kompakt kodlama stili, üretim kullanımı yerine okunabilirlik için tasarlanmıştır.

Sonuçlar Şekil 70 sayfa 785içinde gösterilir.

```
#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);
}
```

Şekil 69. MQ yayın tüketicisi.

```
X:\Subscribe1\Debug>Subscribe1
Optional parameter: subscriptionQueue
Waiting 10 seconds for publications from STOCKTICKER
Received publication "129"
Completion code 0 and Return code 0
```

Şekil 70. MQ yayın tüketicisinin çıkışı

Aşağıdakilere dikkat edilmesi gereken birkaç standart IBM MQ *C* dil programlama ipucu vardır:

memset(publication, 0, sizeof(publicationBuffer));
printfkullanarak kolay biçimlendirme için iletinin sonunda boş değer olduğundan emin olun. Yayınlayıcı örneği, MQPUT 'e geçirilen ileti arabelleğinde sondaki boş değeri (1) strlen(publication)' e ekleyerek içerir. MQCHAR arabelleklerinin boş değere ayarlanması, dizgileri saklamak için arabellekleri kullanan IBM MQ C programları için iyi bir programlama stilidir; boş değer, arabelleği tam olarak doldurmayan bir karakter dizisini izler.

MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen, &CompCode, &Reason);
if (messlen == strlen(publication)); true olursa, döndürülen iletinin sonunda boş değer olmasını sağlamak için, ileti arabelleğinin sonunda bir boş değer ayırın. Bu ipucu öncekini tamamlar ve publicationBuffer içinde publicationiçeriği tarafından üzerine yazılmayan en az bir boş değer olmasını sağlar.

İlgili kavramlar

“Örnek 2: Yönetilen MQ abonesi” sayfa 786
Yönetilen MQ abonesi, çoğu abone uygulaması için tercih edilen kalıptır. Yönetilen abonelik, IBM MQ ' in aboneliği işlediği ve sizin için kayıt yaptırdığı ve kaydını kaldırıldığı bir aboneliktir. Bu örnek, kuyrukların, konuların ya da aboneliklerin *yönetim tanımı* olmasını gerektirir.

“Örnek 3: Yönetilmeyen MQ abonesi” sayfa 791
Yönetilmeyen abone, abone uygulamasının önemli bir sınıfıdır. Bu olanakla, yayınları kuyruğa alma ve kullanma *denetimi* ile yayınlama/abone olma avantajlarını birleştirirsiniz. Yönetilmeyen abonelik, uygulamanın sorumlu olduğu yerdir. aboneliklerin saklandığı kuyruğu belirtmek için. Bu örnek, abonelikleri ve kuyrukları birleştirmenin farklı yollarını gösterir.

“Yayınlayıcı uygulamaları yazılıyor” sayfa 776
İki örneği inceleyerek yayıncı uygulamaları yazmaya başlayın. İlki, iletileri kuyruğa koyan bir noktadan noktaya uygulamada olabildiğince yakından modellenir ve ikincisi, yayınlayıcı uygulamaları için daha yaygın bir örüntü olan konuların dinamik olarak yaratılmasını gösterir.

Örnek 2: Yönetilen MQ abonesi
Yönetilen MQ abonesi, çoğu abone uygulaması için tercih edilen kalıptır. Yönetilen abonelik, IBM MQ ' in aboneliği işlediği ve sizin için kayıt yaptırdığı ve kaydını kaldırıldığı bir aboneliktir. Bu örnek, kuyrukların, konuların ya da aboneliklerin *yönetim tanımı* olmasını gerektirir.

Bu en basit yönetilen abone türü genellikle *dayanıklı olmayan* bir abonelik kullanır. Örnek, sürekli olmayan bir aboneliğe odaklanır. Abonelik, yalnızca MQSUB' in abonelik tanıtıcısının kullanım ömrü boyunca sürer. Abonelik süresi boyunca konu dizgisiyle eşleşen yayınlar abonelik kuyruğuna gönderilir (ve MQSO_NEW_PUBLICATIONS_ONLY işareti ayarlanmamışsa ya da varsayılan olarak belirlenmemişse, konu dizgisiyle eşleşen daha önceki bir yayın korunmuştur ve yayın kalıcı olmuştur ya da yayın yaratıldığından beri kuyruk yöneticisi sonlandırılmamıştır).

Bu kalıpla *sürekli* abonelik de kullanabilirsiniz. Genellikle, yönetilen sürekli abonelik kullanılırsa, herhangi bir hata oluşmadan, aboneden daha uzun yaşayacak bir abonelik oluşturmak yerine, güvenilirlik nedenleriyle yapılır. Yönetilen, yönetilmeyen, sürekli ve sürekli olmayan aboneliklerle ilişkili farklı yaşam döngüleri hakkında daha fazla bilgi için ilgili konular bölümüne bakın.

Sürekli abonelikler genellikle kalıcı yayınlarla ve kalıcı olmayan yayınlarla kalıcı olmayan aboneliklerle ilişkilendirilir, ancak abonelik dayanıklılığı ile yayın sürekliliği arasında gerekli bir ilişki yoktur. Dört süreklilik ve dayanıklılık kombinasyonu da mümkündür.

Sürekli olmayan yönetilen vaka için kuyruk yöneticisi, kuyruk kapatıldığında temizlenen ve silinen bir abonelik kuyruğu oluşturur. Kalıcı olmayan abonelik kapatıldığında yayınlar kuyruktan kaldırılır.

Bu kod tarafından örnek olarak gösterilen yönetilen sürekli olmayan kalıbın değerli yönleri şunlardır:

1. İsteğe bağlı abonelik: Abonelik konusu dizgisi dinamiktir. Çalıştırıldığında uygulama tarafından sağlanır.
2. Kendi kendini yöneten kuyruk: Abonelik kuyruğu kendi kendini tanımlıyor ve yönetiyor.
3. Kendi kendini yöneten abonelik yaşam çevrimi: *sürekli olmayan* abonelikler yalnızca abone uygulaması süresince var olur.

- *Sürekli* yönetilen abonelik tanımlarsanız, kalıcı bir abonelik kuyruğuyla sonuçlanır ve yayınlar, hiçbir abone programı etkin olmadan bu abonelikte saklanmaya devam eder. Kuyruk yöneticisi, yalnızca uygulama ya da denetimci aboneliği silmeyi seçtikten sonra kuyruğu siler (ve kuyruktan alınmayan yayınları temizler). Abonelik, bir yönetim komutu kullanılarak ya da abonelik `MQCO_REMOVE_SUB` seçeneğiyle kapatılarak silinebilir.
- `SubExpiry` ' u sürekli abonelikler için ayarlayarak yayınların kuyruğa gönderilmemesini ve abonenin aboneliği kaldırmadan önce kalan yayınları tüketmesini ve kuyruk yöneticisinin kuyruğu ve kuyruktaki diğer yayınları silmesine neden olmasını önleyebilirsiniz.

4. Esnek konu dizesi devreye alımı: Abonelik konu yönetimi, yönetimsel olarak tanımlanmış bir konu kullanılarak aboneliğin kök kısmının tanımlanmasıyla basitleştirilir. Konu ağacının kök kısmı uygulamadan gizlenir. Bir uygulamanın kök kısmı gizlenerek, uygulama yanlışlıkla başka bir örnek ya da başka bir uygulama tarafından yaratılan başka bir konu ağacıyla çakışan bir konu ağacı yaratmadan konuşlandırılabilir.
5. Yönetilen konular: İlk bölümün yönetimsel olarak tanımlanmış bir konu nesnesiyle eşleştiği bir konu dizesi kullanılarak yayınlar, konu nesnesinin özniteliklerine göre yönetilir.
   - Örneğin, konu dizgisinin ilk kısmı, kümelenmiş bir konu nesnesiyle ilişkilendirilmiş konu dizgisiyle eşleşirse, abonelik kümenin diğer üyelerinden yayınlar alabilir
   - Yönetimsel olarak tanımlanmış konu nesnelerinin ve programsal olarak tanımlanmış aboneliklerin seçici eşleşmesi, her ikisinin avantajlarını birleştirmenizi sağlar. Yönetici, konular için öznitelikler sağlar ve programcı, konuların yönetimiyle ilgilenmeden alt konuları dinamik olarak tanımlar.
   - Genellikle bir ve aynı olmalarına rağmen, `sd.Objectname`eiçinde adı geçen konu nesnesini değil, konuyla ilişkili öznitelikleri sağlayan konu nesnesiyle eşleştirmek için kullanılan sonuçtaki konu dizgisidir. Bkz. "Örnek 2: Bir değişken konusuna yayınlayıcı" sayfa 780.

Aboneliğin örnekte kalıcı olmasını sağlayarak, abone `MQCO_KEEP_SUB` seçeneğiyle aboneliği kapattıktan sonra yayınlar abonelik kuyruğuna gönderilmeye devam eder. Abone etkin olmadığında kuyruk yayınları almaya devam eder. Aboneliği `MQSO_PUBLICATIONS_ON_REQUEST` seçeneğiyle oluşturarak ve alıkonan yayını istemek için `MQSUBRQ` komutunu kullanarak bu davranışı geçersiz kılabilirsiniz.

Abonelik daha sonra `MQCO_RESUME` seçeneğiyle açılarak sürdürülebilir.

`MQSUB` tarafından döndürülen kuyruk tanıtıcısını (`Hobj`) çeşitli şekillerde kullanabilirsiniz.
Kuyruk tanıtıcısı, abonelik kuyruğunun adını sorgulamak için örnekte kullanılır.
Yönetilen kuyruklar, varsayılan model kuyrukları `SYSTEM.NDURABLE.MODEL.QUEUE` ya da `SYSTEM.DURABLE.MODEL.QUEUE`kullanılarak açılır. Abonelikle ilişkili konu nesnesinin özellikleri olarak konu temelinde kendi sürekli ve sürekli olmayan model kuyruklarınızı sağlayarak varsayılanları geçersiz kılabilirsiniz.

Model kuyruklarından devralınan özniteliklerden bağımsız olarak, ek abonelik yaratmak için yönetilen kuyruk tanıtıcısını yeniden kullanamazsınız. Yönetilen kuyruk için, döndürülen kuyruk adını kullanarak yönetilen kuyruğu ikinci kez açarak başka bir tanıtıcı elde edemeyiniz. Kuyruk, dışlayıcı giriş için açılmış gibi davranır.

Yönetilmeyen kuyruklar, yönetilen kuyruklardan daha esnektir. Örneğin, yönetilmeyen kuyrukları paylaşabilir ya da bir kuyrukta birden çok abonelik tanımlayabilirsiniz. Sonraki örnekte, aboneliklerin yönetilmeyen bir abonelik kuyruğuyla nasıl birleştirileceği gösterilmektedir.

Not: Kompakt kodlama stili, üretim kullanımı yerine okunabilirlik için tasarlanmıştır.

Sonuçlar Şekil 73 sayfa 789içinde gösterilir.

```
#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);
}
```

Şekil 71. Yönetilen MQ abonesi-bölüm 1: bildirimler ve parametre işleme.

Bu örnekteki bildirimlerle ilgili yapılacak bazı ek açıklamalar vardır.

MQHOBJ Hobj = MQHO_NONE;
Yayınları almak için kalıcı olmayan bir yönetilen abonelik kuyruğunu belirtik olarak açamazsınız, ancak kuyruk yöneticisi kuyruğunu sizin için açtığında döndürdüğü nesne tanıtıcısı için depolama alanı ayırmanız gerekir. Tutamacı MQHO_OBJECTolarak kullanıma hazırlamanız önemlidir. Bu, kuyruk yöneticisine, abonelik kuyruğuna bir kuyruk tanıtıcısı döndürmesi gerektiğini gösterir.

MQSD sd = {MQSD_DEFAULT};
MQSUBiçinde kullanılan yeni abonelik tanımlayıcısı.

MQCHAR48 qName;
Örnek, abonelik kuyruğu hakkında bilgi gerektirmese de, örnek abonelik kuyruğunun adını sorar- MQINQ bağ tanımı C dilinde biraz garip olduğundan, örneğin bu kısmını incelemek için yararlı bulabilirsiniz.

```
    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;
}
```

Şekil 72. Yönetilen MQ abonesi-bölüm 2: kod gövdesi.

```
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
```

Şekil 73. MQ abonesi

Bu örnekte kodla ilgili yapılması gereken bazı ek açıklamalar vardır.

strncpy(sd.ObjectName, topicName, MQ_Q_NAME_LENGTH);
topicName boş değerli ya da boşsa *(varsayılan değer)*, konu adı çözülen konu dizgisini hesaplamak için kullanılmaz.

sd.ObjectString.VSPtr = topicString;
Yalnızca önceden tanımlanmış bir konu nesnesi kullanmak yerine, bu örnekte programcı MQSUBile birleştirilen bir konu nesnesi ve konu dizgisi sağlar. Konu dizgisinin bir MQCHARV yapısı olduğuna dikkat edin.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
Bir MQCHARV alanının uzunluğunu ayarlamak için bir alternatif.

sd.Options = MQSO_CREATE | MQSO_MANAGED | MQSO_NON_DURABLE | MQSO_FAIL_IF_QUIESCING;
Konu dizgisini tanımladıktan sonra, sd.Options işaretlerinin en dikkatli şekilde dikkat edilmesi gerekir. Birçok seçenek vardır, örnek yalnızca en sık kullanılanıenleri belirtir. Diğer seçenekler varsayılan değerleri kullanır.

1. Abonelik *dayanıksız*olduğundan, yani uygulamada açık abonelik ömrü boyunca olduğundan, MQSO_CREATE işaretini ayarlayın. Okunabilirlik için *(varsayılan)* MQSO_NON_DURABLE işaretini de ayarlayabilirsiniz.
2. MQSO_CREATE , MQSO_RESUMEtamamlayıcı bir üründür. Her iki işaret birlikte ayarlanabilir; kuyruk yöneticisi yeni bir abonelik yaratır ya da uygun olan bir aboneliği sürdürür. Ancak MQSO_RESUME belirtirseniz, sürdürülecek abonelik olmasa da sd.SubNameiçin MQCHARV yapısını da başlatmanız gerekir. SubName kullanıma hazırlanamaması, MQSUB' den 2440: MQRC_SUB_NAME_ERROR dönüş koduyla sonuçlanır.

   Not: MQSO_RESUME , sürekli olmayan yönetilen abonelik için her zaman yoksayılır; ancak, sd.SubName için MQCHARV yapısı kullanıma hazırlanmadan belirtilmesi hataya neden olur.
3. Ayrıca, aboneliğin nasıl açılacağını etkileyen üçüncü bir işaret de vardır, MQSO_ALTER. Doğru izinler verildiğinde, sürdürülmekte olan bir aboneliğin özellikleri MQSUBiçinde belirtilen diğer özniteliklerle eşleşecek şekilde değiştirilir.

   Not: MQSO_CREATE, MQSO_RESUME ve MQSO_ALTER işaretlerinden en az biri belirtilmelidir. Bkz. Seçenekler (MQLONG). "Örnek 3: Yönetilmeyen MQ abonesi" sayfa 791içinde üç bayrağı da kullanma örnekleri vardır.
4. Kuyruk yöneticisinin aboneliği sizin için otomatik olarak yönetmesi için MQSO_MANAGED değerini ayarlayın.

sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
İsteğe bağlı olarak, boş değerli sonlandırılan dizgiler için MQCHARV uzunluğunu ayarlamayı kaldırın ve bunun yerine boş değerli sonlandırıcı işaretini kullanın.

sd.ResObjectString.VSPtr = resTopicStr;
Sonuçtaki konu dizgisi, programdaki ilk printf içinde yankılanır. Çözümlenen dizgiyi programa geri döndürmek için MQCHARV ResObjectString for IBM MQ ayarını kullanın.

Not: resTopicStringBuffer , memset(resTopicStr, 0, sizeof(resTopicStrBuffer))içinde boş değerlerle kullanıma hazırlanır. Döndürülen konu dizgileri sondaki boş değerle bitmez.

sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;
sd.ResObjectString arabellek büyüklüğünü, gerçek boyutundan bir daha az olacak şekilde ayarlayın. Bu, çözülen konu dizgisinin tüm arabelleği doldurması durumunda, sağlanan boş değerli sonlandırıcının üzerine yazılmasını önler.

Not: Konu dizgisi sizeof(resTopicStrBuffer) -1değerinden uzunsa hata döndürülmez. VSLength > VSBufSiz sd.ResObjectString.VSLength içinde döndürülen uzunluk, tam dizginin uzunluğudur ve her zaman döndürülen dizginin uzunluğu değildir. Konu dizgisinin tamamlandığını doğrulamak için sd.ResObjectString.VSLength < sd.ResObjectString.VSBufSiz sınayın.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
MQSUB işlevi bir abonelik yaratır. Kalıcı değilse, büyük olasılıkla adıyla ilgilenmiyorsunuzdur, ancak IBM MQ Explorer 'da durumunu inceleyebilirsiniz. Giriş olarak sd.SubName parametresini sağlayabilirsiniz; böylece hangi adı arayabileceğinizi bilebilirsiniz; açıkça diğer aboneliklerle ad çakışmasını önlemeniz gerekir.

MQCLOSE(Hconn, &Hsub, MQCO_REMOVE_SUB, &CompCode, &Reason);
Hem abonelik hem de abonelik kuyruğunun kapatılması isteğe bağlıdır. Örnekte abonelik kapatılır, ancak kuyruk kapatılır. Abonelik kalıcı olmadığı için, MQCLOSE MQCO_REMOVE_SUB seçeneği bu durumda varsayılan seçenektir. MQCO_KEEP_SUB ' in kullanılması bir hatadır.

Not: Abonelik *kuyruğu* MQSUBtarafından kapatılmaz ve tanıtıcısı Hobj, kuyruk MQCLOSE ya da MQDISCtarafından kapatılıncaya kadar geçerli kalır. Uygulama zamanından önce sona erse, kuyruk ve abonelik, uygulama sonlandırıldıktan bir süre sonra kuyruk yöneticisi tarafından temizlenir.

İlgili kavramlar
“Örnek 1: MQ Yayın tüketicisi” sayfa 784
MQ Yayın tüketicisi, konuların kendisine abone olmayan bir IBM MQ ileti tüketicisidir.

“Örnek 3: Yönetilmeyen MQ abonesi” sayfa 791
Yönetilmeyen abone, abone uygulamasının önemli bir sınıfıdır. Bu olanakla, yayınları kuyruğa alma ve kullanma *denetimi* ile yayınlama/abone olma avantajlarını birleştirirsiniz. Yönetilmeyen abonelik, uygulamanın sorumlu olduğu yerdir. aboneliklerin saklandığı kuyruğu belirtmek için. Bu örnek, abonelikleri ve kuyrukları birleştirmenin farklı yollarını gösterir.

“Yayınlayıcı uygulamaları yazılıyor” sayfa 776
İki örneği inceleyerek yayıncı uygulamaları yazmaya başlayın. İlki, iletileri kuyruğa koyan bir noktadan noktaya uygulamada olabildiğince yakından modellenir ve ikincisi, yayınlayıcı uygulamaları için daha yaygın bir örüntü olan konuların dinamik olarak yaratılmasını gösterir.

Örnek 3: Yönetilmeyen MQ abonesi
Yönetilmeyen abone, abone uygulamasının önemli bir sınıfıdır. Bu olanakla, yayınları kuyruğa alma ve kullanma *denetimi* ile yayınlama/abone olma avantajlarını birleştirirsiniz. Yönetilmeyen abonelik, uygulamanın sorumlu olduğu yerdir. aboneliklerin saklandığı kuyruğu belirtmek için. Bu örnek, abonelikleri ve kuyrukları birleştirmenin farklı yollarını gösterir.

Yönetilmeyen örüntü, *kalıcı olmayan*aboneliklerden daha sık *sürekli* aboneliklerle ilişkilendirilir. Genellikle yönetilmeyen bir abone tarafından oluşturulan bir aboneliğin yaşam çevrimi, abone olan uygulamanın yaşam döngüsünden bağımsızdır. Abonelik sürekli hale getirilerek abonelik, abone olan bir uygulama etkin olmadığında bile yayınları alır.

Aynı sonucu elde etmek için sürekli *yönetilen* abonelikler oluşturabilirsiniz, ancak bazı uygulamalar, yönetilen abonelikle mümkün olandan daha fazla esneklik ve kuyruk ve ileti denetimi gerektirir. Sürekli yönetilen abonelik için kuyruk yöneticisi, abonelik konusuyla eşleşen yayınlar için kalıcı bir kuyruk yaratır. Abonelik silindiğinde kuyruğu ve ilişkili yayınları siler.

Genellikle, uygulamanın yaşam çevrimi ve abonelik temelde aynıysa, ancak garanti edilmesi zorsa, sürekli *yönetilen* abonelikler kullanılır. Aboneliği kalıcı hale getirerek ve yayıncının kalıcı yayınlar oluşturmasını sağlayarak, kuyruk yöneticisi ya da abonenin zamanından önce sona ermesi ve kurtarılması gereken kayıp iletiler yoktur.

JMS dışı uygulamalar ya da paylaşılan abonelik kullanmayan JMS uygulamaları için kuyruk yöneticisi, bir aboneye ilişkin sürekli yönetilen abonelik kuyruğunu örtük olarak açar; böylece, kuyruğun paylaşılan olarak işlenmesi olanaksız olur. Ayrıca, uygulamanız JMS paylaşılan aboneliklerini kullanmıyorsa, her yönetilen kuyruk için birden fazla abonelik yaratılamaz ve kuyruk adları üzerinde daha az denetime sahip olduğunuz için kuyrukları daha zor yönetebilirsiniz. Bu nedenlerle, *yönetilmeyen* MQ abonesinin *yönetilen* MQ abonesine göre sürekli abonelik gerektiren uygulamalar için daha uygun olup olmadığını göz önünde bulundurun.

Şekil 76 sayfa 796 içindeki kod, yönetilmeyen sürekli abonelik kalıbını gösterir. Örnek olarak kod, yönetilmeyen, sürekli olmayan abonelikler de oluşturur. Bu örnek, aşağıdaki örüntü kategorilerini gösterir:

- İsteğe bağlı abonelikler: Abonelik konusu dizgileri dinamiktir. Bunlar, çalıştırıldığında uygulama tarafından sağlanır.
- Basitleştirilmiş abonelik konu yönetimi: abonelik konu yönetimi, yönetici tarafından tanımlanan bir konu kullanılarak abonelik konu dizgisinin kök kısmının tanımlanmasıyla basitleştirilir. Bu, konu ağacının kök kısmını uygulamadan gizler. Kök kısmı gizlenerek bir abone farklı konu ağaçlarına konuşlandırılabilir.
- Esnek abonelik yönetimi: Bir aboneliği yönetici olarak tanımlayabilir ya da bir abone programında isteğe bağlı olarak oluşturabilirsiniz. Aboneliğin nasıl oluşturulduğunu gösteren bir öznitelik dışında, yönetimsel olarak ve programsal olarak oluşturulan abonelikler arasında fark yoktur. Aboneliklerin dağıtımı için kuyruk yöneticisi tarafından otomatik olarak yaratılan üçüncü bir abonelik tipi vardır. Tüm abonelikler IBM MQ Explorer 'da görüntülenir.
- Aboneliklerin kuyruklarla esnek ilişkilendirmesi: Önceden tanımlanmış bir yerel kuyruk, MQSUB işlevi tarafından bir abonelikle ilişkilendirilir. Abonelikleri kuyruklarla ilişkilendirmek için MQSUB ' yi kullanmanın farklı yolları vardır:
  - Bir aboneliği, *var olan* aboneliği olmayan `MQSO_CREATE + (Hobj from MQOPEN)` bir kuyrukla ilişkilendirin.
  - *Yeni* bir aboneliği, var olan abonelikleri olan bir kuyrukla ilişkilendirin, `MQSO_CREATE + (Hobj from MQOPEN)`.
  - Var olan bir aboneliği farklı bir kuyruğa (`MQSO_ALTER + (Hobj from MQOPEN)`) taşıyın.
  - Var olan bir kuyrukla, `MQSO_RESUME + (Hobj = MQHO_NONE)` ya da `MQSO_RESUME + (Hobj = from MQOPEN of queue with existing subscription)` ile ilişkili var olan bir aboneliği sürdürün.
  - `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` ' i farklı birleşimlerde birleştirerek, farklı `sd.Options` değerleriyle birden çok MQSUB sürümünü kodlamak zorunda kalmadan, aboneliğin ve kuyruğun farklı giriş durumlarına hitap edebilirsiniz.
  - Alternatif olarak, belirli bir `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` seçeneği kodlayarak kuyruk yöneticisi bir hata döndürür (Çizelge 123 sayfa 793) MQSUB ' e giriş olarak sağlanan abonelik ve kuyruk durumları `sd.Options`değeriyle tutarsızsa. Şekil 82 sayfa 799 içinde, `sd.Options` işaretinin farklı bireysel ayarlarıyla Abonelik X için MQSUB komutu verilmesinin sonuçları gösterilir ve üç farklı nesne tanıtıcısı geçirilir.

  Bu farklı hata türlerine alışmak için Şekil 75 sayfa 795 içindeki örnek programa farklı girişler keşfedin. Çizelgede listelenen vakalarda içerilmeyen bir ortak hata (`RC = 2440`), abonelik adı hatasıdır. Bunun nedeni genellikle, `MQSO_RESUME` ya da `MQSO_ALTER`ile boş değerli ya da geçersiz bir abonelik adının iletilmesi olabilir.
- Çoklu işlem: Yayınları okuma çalışmalarını birçok tüketici arasında paylaşabilirsiniz. Yayınların tümü, abonelik konusuyla ilişkili tek kuyruğa gider. Tüketiciler, `MQOPEN` kullanarak doğrudan kuyruğu açma ya da `MQSUB`kullanarak aboneliği sürdürme seçeneğine sahip olur.
- Abonelik konsantrasyonu: Aynı kuyrukta birden çok abonelik oluşturulabilir. Çakışan aboneliklere ve aynı yayını birden çok kez almaya neden olabilmesi için bu yeteneğe karşı dikkatli olun. `MQSO_GROUP_SUB` seçeneği, çakışan aboneliklerin neden olduğu yinelenen yayınları ortadan kaldırır.
- Abone ve tüketici ayrımı: Bu örneklerde gösterilen üç tüketici modelinin yanı sıra, başka bir model de tüketiciyi aboneden ayırmaktır. Yönetilmeyen MQ Abonesinin bir varyasyonudur, ancak aynı programda `MQOPEN` ve `MQSUB` yayınlarını yayınlamak yerine, bir program yayınlara abone olur ve başka bir program bunları kullanır. Örneğin, abone bir yayınlama/abone olma kümesinin bir parçası ve tüketici kuyruk yöneticisi kümesinin dışında bir kuyruk yöneticisine bağlı olabilir. Tüketici, abonelik kuyruğunu uzak kuyruk tanımı olarak tanımlayarak, standart dağıtılmış kuyruğa alma yoluyla yayınları alır.

Özellikle bu seçeneklerin birleşimlerini kullanarak kodunuzu basitleştirmeyi planlıyorsanız, `MQSO_CREATE | MQSO_RESUME | MQSO_ALTER` davranışının anlaşılması önemlidir. Farklı kuyruk tanıtıcılarının MQSUB ' ye geçirilmesinin sonuçlarını ve Şekil 77 sayfa 797 to Şekil 82 sayfa 799içinde gösterilen örnek programın çalıştırılmasının sonuçlarını gösteren Çizelge 123 sayfa 793 çizelgesini araştırın.

Çizelgeyi oluşturmak için kullanılan senaryoda bir abonelik X ve iki kuyruk (A ve B) vardır. Abonelik adı parametresi `sd.SubName` , kuyruğa eklenen bir aboneliğin adı olan Xolarak ayarlandı A. B kuyruğuna abonelik eklenmedi.

Çizelge 123 sayfa 793içinde, MQSUB aboneliği geçirir X ve kuyruk tanıtıcısı Akuyruğuna geçirilir. Abonelik seçeneklerinden elde edilen sonuçlar aşağıdaki gibidir:

- Kuyruk tanıtıcısı, zaten Xaboneliği olan A kuyruğuna karşılık geldiğinden MQSO_CREATE başarısız oldu. Bu davranışı başarılı çağrıya karşılaştırın. B kuyruğunun kendisine eklenmiş X aboneliği olmadığı için bu çağrı başarılı olur.
- Kuyruk tanıtıcısı, Xaboneliği olan A kuyruğuna karşılık geldiği için MQSO_RESUME başarılı olur. Buna karşılık, X aboneliği kuyrukta Ayoksa çağrı başarısız olur.
- MQSO_ALTER , aboneliğin ve kuyruğun açılmasına ilişkin olarak MQSO_RESUME ile benzer bir şekilde davranır. Ancak, MQSUB ' e geçirilen abonelik tanımlayıcısında bulunan öznitelikler aboneliğin özniteliklerinden farklıysa, MQSO_RESUME başarısız olur; MQSO_ALTER ise, program eşgörünümünün öznitelikleri değiştirme izni olduğu sürece başarılı olur. Bir abonelikte konu dizgisini hiçbir zaman değiştiremeyeceğinizi, ancak bir hata döndürmek yerine MQSUB ' in abonelik tanımlayıcısındaki konu adını ve konu dizgisi değerlerini yoksaydığını ve var olan abonelikteki değerleri kullandığını unutmayın.

Daha sonra, Çizelge 123 sayfa 793 adresine bakın. Burada MQSUB aboneliği X 'e ve kuyruk tanıtıcısı B kuyruğuna aktarılır. Abonelik seçeneklerinden elde edilen sonuçlar aşağıdaki gibidir:

- MQSO_CREATE başarılı olur ve kuyrukta X abonelik oluşturur B ; bu, kuyrukta yeni bir abonelik B.
- MQSO_RESUME başarısız olur. MQSUB , kuyrukta X aboneliğini arar B ve bulmaz; ancak *RC = 2428-subscription X yok*döndürmek yerine, *RC = 2019-Abonelik kuyruğu, kuyruk nesnesi tanıtıcısı ile eşleşmiyor*döndürür. Üçüncü seçeneğin MQSO_ALTER davranışı, bu beklenmeyen hatanın nedenini gösterir. MQSUB , kuyruk tanıtıcıının aboneliğe sahip bir kuyruğu göstermesini bekler. sd.SubName içinde adı belirtilen aboneliğin var olup olmadığını denetlemeden önce bunu denetler.
- MQSO_ALTER başarılı olur ve aboneliği kuyruktan A kuyruğa Btaşır.

Çizelgede gösterilmeyen bir durum, A kuyruğundaki aboneliğin abonelik adının sd.SubNameiçindeki abonelik adıyla eşleşmemesi durumudur. Bu çağrı *RC = 2428-abonelik X*ile başarısız olur.

Çizelge 123. Farklı kuyruk tanıtıcılarına ve abonelik birleşimlerine sahip MQSUB hataları

Kuyruk tanıtıcıları	**Kuyruk A** **Abonelik X** **Kuyruk B** **Abonelik yok**	**Kuyruk A** **Abonelik yok** **Kuyruk B** **Abonelik yok**
MQSUB ' ye iletilen *Kuyruk A* için Hobj	**MQSO_CREATE** RC = 2432-Abonelik X, A kuyruğunda zaten var **MQSO_RESUME** A kuyruğundaki X aboneliğini sürdürür **MQSO_ALTER** Kuyruk A ' da X aboneliğini sürdürür ve izin verilen değişiklikleri yapar	**MQSO_CREATE** A kuyruğunda X aboneliği yaratır **MQSO_RESUME** RC = 2428-Abonelik X, Kuyruk A ' da yok **MQSO_ALTER** RC = 2428-Abonelik X, Kuyruk A ' da yok
***Kuyruk B* için Hobj, MQSUB ' ye iletildi**	**MQSO_CREATE** Kuyruk B ' de yeni abonelik X oluşturur **MQSO_RESUME** RC = 2019-Abonelik kuyruğu, kuyruk nesnesi tanıtıcısı ile eşleşmiyor **MQSO_ALTER** X aboneliğini A kuyruğundan B kuyruğuna taşı	**MQSO_CREATE** Kuyruk B ' de yeni abonelik X oluşturur **MQSO_RESUME** RC = 2428-abonelik X, Kuyruk B ' de yok **MQSO_ALTER** RC = 2428-abonelik X, Kuyruk B ' de yok

Çizelge 123. Farklı kuyruk tanıtıcılarına ve abonelik birleşimlerine sahip MQSUB hataları (devamı var)		
Kuyruk tanıtıcıları	**Kuyruk A** **Abonelik X** **Kuyruk B** **Abonelik yok**	**Kuyruk A** **Abonelik yok** **Kuyruk B** **Abonelik yok**
MQHO_NONE, MQSUB ' ye iletildi	**MQSO_CREATE** RC = 2019-Nesne tanıtıcısı hatalı: yönetilen abonelik oluşturmak ve yönetilen kuyruk oluşturmak için MQSO_MANAGED işaretini ayarlayın **MQSO_RESUME** A kuyruğundaki X aboneliğini sürdürür ve A kuyruğuna Hobj 'u döndürür. **MQSO_ALTER** A kuyruğundaki X aboneliğini sürdürür, A kuyruğuna Hobj döndürür ve izin verilen değişiklikleri yapar	**MQSO_CREATE** RC = 2019-Nesne tanıtıcısı hatalı: yönetilen abonelik oluşturmak ve yönetilen kuyruk oluşturmak için MQSO_MANAGED işaretini ayarlayın **MQSO_RESUME** RC = 2428-Abonelik X yok **MQSO_ALTER** RC = 2019-Nesne tanıtıcısı hatalı: A ya da B kuyruğu yok

Not: Kompakt kodlama stili, üretim kullanımı yerine okunabilirlik için tasarlanmıştır.

```
#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));
```

Şekil 74. Yönetilmeyen MQ abonesi-bölüm 1: bildirimler.

```
    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);
}
```

Şekil 75. Yönetilmeyen MQ abonesi-bölüm 2: parametre işleme.

Bu örnekte parametre işlemesiyle ilgili ek açıklamalar şunlardır:

switch((argv[5][0]))
Örnekte varsayılan olarak kullanılan MQSUB seçenek ayarının bir kısmının geçersiz kılınmasının etkisini sınamak için parametre 5 'e A lter | C reate | R esume girmeyi seçersiniz. Örnek tarafından kullanılan varsayılan ayar şudur: MQSO_CREATE | MQSO_RESUME | MQSO_DURABLE.

Not: MQSO_ALTER ya da MQSO_RESUME ayarı olmadan MQSO_DURABLE ayarı bir hatadır ve sd.SubName ayarlanmalı ve sürdürülebilen ya da değiştirilebilen bir aboneliğe başvurmalıdır.

***subscriptionQueue = '\0';**
sdOptions = sdOptions + MQSO_MANAGED;
Varsayılan abonelik kuyruğu STOCKTICKER boş değerli bir dizgiyle değiştirilirse, MQSO_CREATE ayarlandığı sürece, örnek MQSO_MANAGED işaretini ayarlar ve dinamik bir abonelik kuyruğu oluşturur. Beşinci parametrede Alter or Resume ayarlanırsa, örneğin davranışı subscriptionNamedeğerine bağlıdır.

***subscriptionName = '\0';**
sdOptions = sdOptions - MQSO_DURABLE;
Varsayılan abonelik (IBMSTOCKPRICESUB) boş değerli bir dizgiyle değiştirilirse, örnek MQSO_DURABLE işaretini kaldırır. Diğer parametreler için varsayılan değerleri sağlayan örneği çalıştırırsanız, STOCKTICKER ' e yönlendirilen ek bir geçici abonelik oluşturulur ve yinelenen yayınlar alır. Örneği, herhangi bir parametre olmadan bir daha çalıştırdığınızda, yalnızca bir yayın yeniden alırsınız.

```
    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;
}
```

Şekil 76. Yönetilmeyen MQ abonesi-bölüm 3: kod gövdesi.

Bu örnekteki kodla ilgili ek açıklamalar şunlardır:

if (strlen(subscriptionQueue))
Abonelik kuyruğu adı yoksa, örnek, Hobjdeğeri olarak MQHO_NONE değerini kullanır.

MQOPEN(...);
Abonelik kuyruğu açılır ve kuyruk tanıtıcısı Hobjiçine kaydedilir.

MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
Abonelik, MQOPEN ' den iletilen Hobj (ya da abonelik kuyruğu adı yoksa MQHO_NONE) kullanılarak açılır. Yönetilmeyen bir kuyruk, bir MQOPENile açık bir şekilde açılmadan sürdürülebilir.

MQCLOSE(Hconn, &Hsub, MQCO_NONE, &CompCode, &Reason);
Abonelik, abonelik tanıtıcısı kullanılarak kapatılır. Aboneliğin sürekli olup olmadığına bağlı olarak, abonelik örtük bir MQCO_KEEP_SUB ya da MQCO_REMOVE_SUBile kapatılır. MQCO_REMOVE_SUBile sürekli bir aboneliği kapatabilirsiniz, ancak MQCO_KEEP_SUBile kalıcı olmayan bir aboneliği kapatamamazsınız. MQCO_REMOVE_SUB işlemi, abonelik kuyruğuna gönderilen diğer yayınları durduran aboneliği kaldırmaktır.

MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
Abonelik yönetilmezse özel bir işlem gerçekleştirilmez. Kuyruk yönetiliyorsa ve abonelik belirtik ya da örtük bir MQCO_REMOVE_SUBile kapatılırsa, tüm yayınlar kuyruktan temizlenir ve kuyruk bu noktada silinir.

gmo.MatchOptions = MQMO_MATCH_CORREL_ID;
memcpy(md.CorrelId, sd.SubCorrelId, MQ_CORREL_ID_LENGTH);
Alınan iletilerin aboneliğimiz için gönderildiğinden emin olun.

Örnekteki sonuçlar, yayınlama/abone olma özelliklerini gösterir:

Şekil 77 sayfa 797 içinde örnek, NYSE/IBM/PRICE konusunda 130 yayınlanarak başlar.

```
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
```

Şekil 77. NYSE/IBM/PRICE ' de 130 yayınla

Şekil 78 sayfa 797 ' de örneğin varsayılan parametreler kullanılarak yürütülmesi, alıkonan yayını alır 130. Şekil 82 sayfa 799içinde gösterildiği gibi, sağlanan konu nesnesi ve konu dizgisi yoksayılır. Konu nesnesi ve konu dizgisi her zaman abonelik nesnesinden alınır, bir nesne sağlandığında ve konu dizgisi değişmez olduğunda. Örneğin gerçek davranışı, MQSO_CREATE, MQSO_RESUMEve MQSO_ALTERseçeneklerine ya da birleşimlerine bağlıdır. Bu örnekte MQSO_RESUME , seçilen seçenektir.

```
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
```

Şekil 78. Alıkonan yayını al

İçinde (Şekil 79 sayfa 798) Sürekli abonelik alıkonan yayını zaten aldığı için hiçbir yayın alınmadı. Bu örnekte, abonelik kuyruk adı olmadan yalnızca abonelik adı sağlanarak sürdürülür. Kuyruk adı sağlandıysa, önce kuyruk açılır ve tanıtıcı MQSUB' e iletilir.

Not: MQINQ ' daki 2038 hatası, MQOO_INQUIRE seçeneğini içermeyen MQSUB tarafından STOCKTICKER ile ilgili örtük MQOPEN nedeniyle ortaya çıktı. Kuyruğu belirtik olarak açarak 2038 dönüş kodundan MQINQ kaçının.

```
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
```

Şekil 79. Aboneliği sürdür

Şekil 80 sayfa 798içinde örnek, hedef olarak STOCKTICKER kullanılarak, sürekli olmayan bir yönetilmeyen abonelik yaratır. Bu yeni bir abonelik olduğundan, alıkonan yayını alır.

```
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
```

Şekil 80. Yeni yönetilmeyen sürekli olmayan abonelikle alıkonan yayını al

Şekil 81 sayfa 798içinde, çakışan abonelikleri göstermek için, alıkonan yayını değiştirerek başka bir yayın gönderilir. Daha sonra, yeni bir kalıcı olmayan, yönetilmeyen abonelik, bir abonelik adı sağlanmadan oluşturulur. Alıkonan yayın, yeni abonelik için bir kez olmak üzere iki kez, STOCKTICKER kuyruğunda hala etkin olan sürekli IBMSTOCKPRICESUB aboneliği için bir kez alınır. Örnek, uygulamanın değil, aboneliklerin olduğu bir kuyruktur. Uygulamanın bu çağrılışında IBMSTOCKPRICESUB aboneliğine atıfta bulunmasa da, uygulama yayını iki kez alır: biri yönetimsel olarak oluşturulan sürekli abonelikten, diğeri de uygulamanın kendisi tarafından oluşturulan sürekli olmayan abonelikten.

```
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
```

Şekil 81. Çakışan abonelikler

Şekil 82 sayfa 799 içinde örnek, yeni bir konu dizgisi ve var olan bir abonelik sağlanmasının, aboneliğin değiştirilmesiyle sonuçlanmadığını gösterir.

1. İlk durumda Resume , beklediğiniz gibi var olan aboneliği sürdürür ve değiştirilen konu dizgisini yoksayar.
2. İkinci durumda, Alter bir hataya neden olur, RC = 2510, Topic not alterable.
3. Üçüncü örnekte Create bir hataya neden olur 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
```

Şekil 82. Abonelik konuları değiştirilemez

İlgili kavramlar

“Örnek 1: MQ Yayın tüketicisi” sayfa 784
MQ Yayın tüketicisi, konuların kendisine abone olmayan bir IBM MQ ileti tüketicisidir.

“Örnek 2: Yönetilen MQ abonesi” sayfa 786
Yönetilen MQ abonesi, çoğu abone uygulaması için tercih edilen kalıptır. Yönetilen abonelik, IBM MQ ' in aboneliği işlediği ve sizin için kayıt yaptırdığı ve kaydını kaldırıldığı bir aboneliktir. Bu örnek, kuyrukların, konuların ya da aboneliklerin *yönetim tanımı* olmasını gerektirir.

“Yayınlayıcı uygulamaları yazılıyor” sayfa 776
İki örneği inceleyerek yayıncı uygulamaları yazmaya başlayın. İlki, iletileri kuyruğa koyan bir noktadan noktaya uygulamada olabildiğince yakından modellenir ve ikincisi, yayınlayıcı uygulamaları için daha yaygın bir örüntü olan konuların dinamik olarak yaratılmasını gösterir.

Yaşam çevrimlerini yayınla/abone ol

Yayınlama/abone olma uygulamalarının tasarlanmasında konuların, aboneliklerin, abonelerin, yayınların, yayıncıların ve kuyrukların yaşam çevrimlerini göz önünde bulundurun.

Abonelik gibi bir nesnenin yaşam çevrimi, oluşturulmasıyla başlar ve silinmesiyle biter. Ayrıca, geçici askıya alma, üst ve alt konulara sahip olma, süre bitimi ve silme gibi diğer durumları ve değişiklikleri de içerebilir.

Geleneksel olarak IBM MQ nesneleri (kuyruklar gibi), denetim amaçlı olarak ya da Programlanır Komut Biçimi (PCF) kullanılarak denetim programları tarafından yaratılır. Yayınlama/abone olma, abonelik oluşturmak ve silmek için MQSUB ve MQCLOSE API fiillerinin sağlanmasında, yalnızca kuyrukları oluşturup silmekle kalmayıp, kullanılmayan iletileri de temizleyen yönetilen abonelikler kavramına sahip olmasında ve yönetimsel olarak oluşturulan konu nesneleri ile programsal ya da yönetimsel olarak oluşturulan konu dizgileri arasında ilişkilendirmelere sahip olmasında farklıdır.

Bu işlevsel zenginlik, geniş bir dizi yayınlama/abone olma gereksinimini karşılar ve aynı zamanda bazı yaygın yayınlama/abone olma uygulaması kalıplarının tasarlamasını kolaylaştırır. Örneğin, yönetilen abonelikler, yalnızca onu oluşturan program kadar uzun sürmesi amaçlanan bir aboneliğin hem programlamasını hem de yönetimini basitleştirir. Yönetilmeyen abonelikler, yayınlara abone olmak ve yayınları tüketmek arasında daha gevşek bir bağlantı olduğu durumlarda programlamayı basitleştirir. Merkezi olarak oluşturulan abonelikler, örneğin uçuş bilgilerinin otomatik kapılara gönderilmesi gibi merkezi bir kontrol modeline dayalı yayın trafiğinin tüketicilere yönlendirilmesinden biri olduğu durumlarda kullanışlıdır, ancak geçit personeli o uçuş için yolcu kayıtlarına abone olmak için bir geçitten bir uçuş numarası girerek programlı olarak oluşturulan abonelikler kullanılabilir.

Bu son örnekte, yönetilen sürekli abonelik uygun olabilir: abonelikler çok sık oluşturulduğundan ve geçit kapandığında net bir uç noktası olduğundan ve abonelik programlı olarak kaldırılabildiğinden; sürekli olarak, geçit abone programı nedeniyle bir yolcu kaydını kaybetmemek için bir nedenden dolayı ya da başka bir nedenden dolayı[8]. Yolcu kayıtlarının geçitte yayınlanmasını başlatmak için, olası bir tasarım, geçit başvurusunun hem kapı numarasını kullanarak yolcu kayıtlarına abone olması hem de kapı numarasını kullanarak kapı açma olayını yayınlaması olacaktır. Yayıncı, yolcu kayıtlarını yayınlayarak geçit açılış etkinliğine yanıt verir-ki bu, daha sonra uçuşun gerçekleştiğini kaydetmek için faturalama gibi diğer ilgili taraflara da gidebilir ve müşteri hizmetlerine, yolcuların kapı numarasının cep telefonlarına bildirim gönderebilir.

[8] Yayıncı, diğer olası hataları önlemek için yolcu kayıtlarını kalıcı mesaj olarak göndermelidir.

Merkezi olarak yönetilen abonelik, her geçit için önceden tanımlanmış bir kuyruk kullanarak yolcu listelerini kapıya yönlendirmek için sürekli bir yönetilmeyen model kullanabilir.

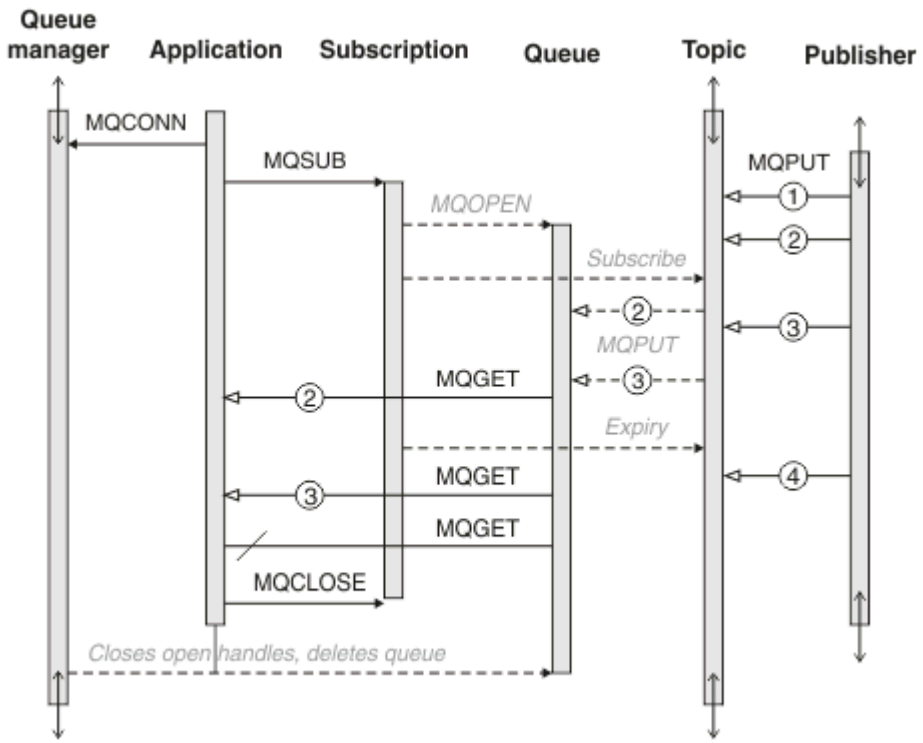
Aşağıdaki yayınlama/abone olma yaşam çevrimlerine ilişkin üç örnek, yönetilen sürekli, yönetilen ve yönetilmeyen abonelerin aboneliklerle, konularla, kuyruklarla, yayıncılarla ve kuyruk yöneticisiyle nasıl etkileşim kurduğunu ve sorumlulukların yönetim ve abone programları arasında nasıl bölünebileceğini göstermektedir.

Yönetilen sürekli olmayan abone

Şekil 83 sayfa 800 içinde, yönetilen kalıcı olmayan abonelik yaratan, abonelikte tanımlanan konuya yayınlanan ve sonlandırılan iki ileti alan bir uygulama gösterilir. İtalik gri yazı tipinde noktalı oklarla etiketlenen etkileşimler örtük olarak gösterilir.

Dikkat etmesi gereken bazı noktalar var.

1. Uygulama, zaten iki kez yayınlanmış bir konuda abonelik oluşturur. Abone ilk yayınını aldığında, alıkonan yayın olan *ikinci* yayını alır.
2. Kuyruk yöneticisi, konu için abonelik yaratmanın yanı sıra geçici bir abonelik kuyruğu yaratır.
3. Aboneliğin bir süre bitimi var. Abonelik süresi sona erdiğinde, bu aboneliğe başka yayın gönderilmez, ancak abone, abonelik süresi dolmadan önce yayınlanan iletileri almaya devam eder. Yayın süre bitimi, abonelik süre bitiminden etkilenmez.
4. Dördüncü yayın abonelik kuyruğuna yerleştirilmez ve sonuç olarak son MQGET bir yayın döndürmez.
5. Abone aboneliğini kapatsa da, kuyruk ya da kuyruk yöneticisiyle bağlantısını kapatmaz.
6. Kuyruk yöneticisi, uygulama sona erdikten kısa bir süre sonra temizleniyor. Abonelik yönetildiği ve kalıcı olmadığı için abonelik kuyruğu silinir.



Şekil 83. Yönetilen sürekli olmayan abone ömür çizgisi

Yönetilen sürekli abone

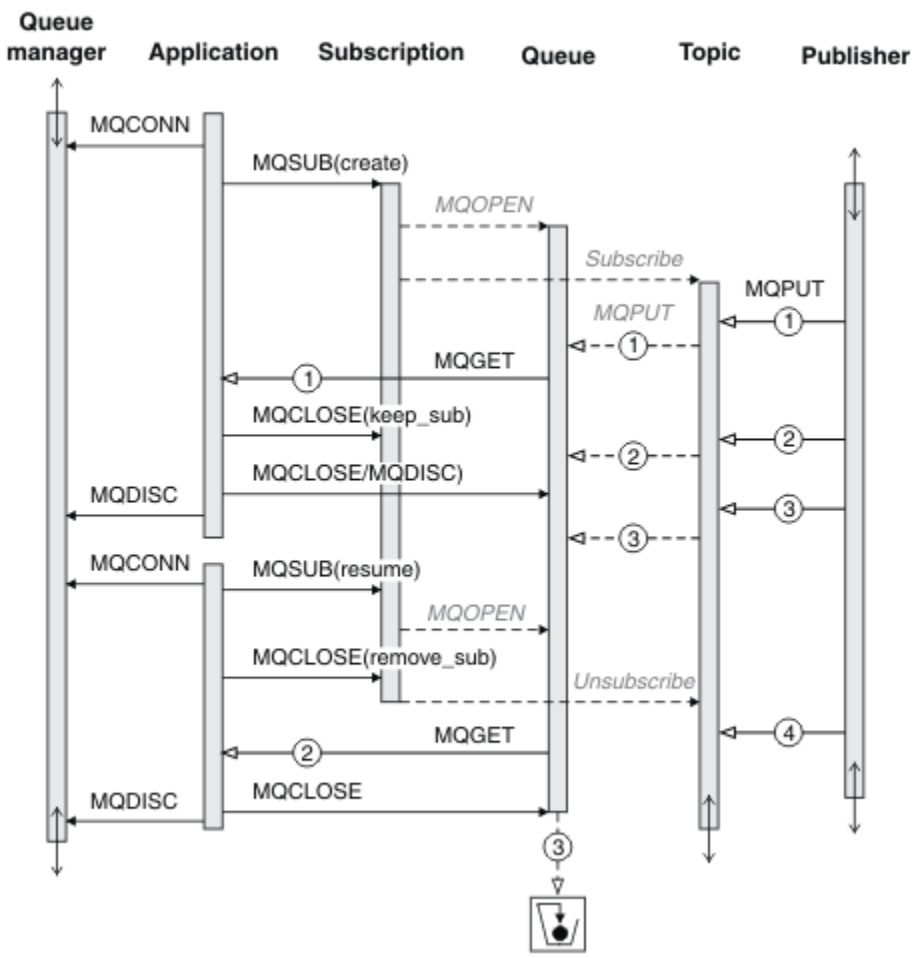
Yönetilen sürekli abone, önceki örneği bir adım daha ileriye taşır ve abone olan uygulamanın sona erdirilmesi ve yeniden başlatılmasına devam eden yönetilen bir aboneliği gösterir.

Dikkat etmesi gereken bazı yeni noktalar var.

1. Bu örnekte, sonuncunun tersine, yayın konusu abonelikte tanımlanmadan önce yoktu.
2. Abone ilk kez sonlandırıldığında, MQCO_KEEP_SUBseçeneğiyle aboneliği kapatır. Yönetilen sürekli bir aboneliğin örtük olarak kapatılması için varsayılan davranış budur.
3. Abone aboneliği sürdürdüğünde, abonelik kuyruğu yeniden açılır.
4. Yeniden açılmadan önce kuyruğa yerleştirilen 2adlı yeni yayın, abonelik kaldırıldıktan sonra bile MQGET' in kullanımına sunulur.

   Abonelik sürekli olsa da, abone, yayıncı tarafından gönderilen tüm iletileri yalnızca *her ikisi* abonelik dayanıklıysa ve iletiler kalıcıysa güvenilir bir şekilde alır. İleti kalıcılığı, yayınlayıcı tarafından gönderilen iletinin MQMD içindeki Persistent alanının ayarına bağlıdır. Bir abonenin bu konuda hiçbir kontrolü yoktur.
5. MQCO_REMOVE_SUB işaretiyle abonelik kapatıldığında abonelik kaldırılır ve abonelik kuyruğuna yerleştirilmekte olan diğer yayınlar durduruluyor. Abonelik kuyruğu kapatıldığında, kuyruk yöneticisi okunmamış yayını 3kaldırır ve sonra kuyruğu siler. Bu işlem, aboneliğin yönetimsel olarak silinmesiyle eşdeğerdir.

   Not: Kuyruğu el ile silmeyin ya da MQCLOSE komutunu MQCO_DELETEya da MQCO_PURGE_DELETEseçeneğiyle verin. Yönetilen aboneliğin görünür uygulama ayrıntıları, desteklenen IBM MQ arabiriminin bir parçası değildir. Kuyruk yöneticisi, tam denetime sahip olmadığı sürece bir aboneliği güvenilir bir şekilde yönetemez.



Şekil 84. Yönetilen sürekli abone ömür çizgisi

Yönetilmeyen sürekli abone

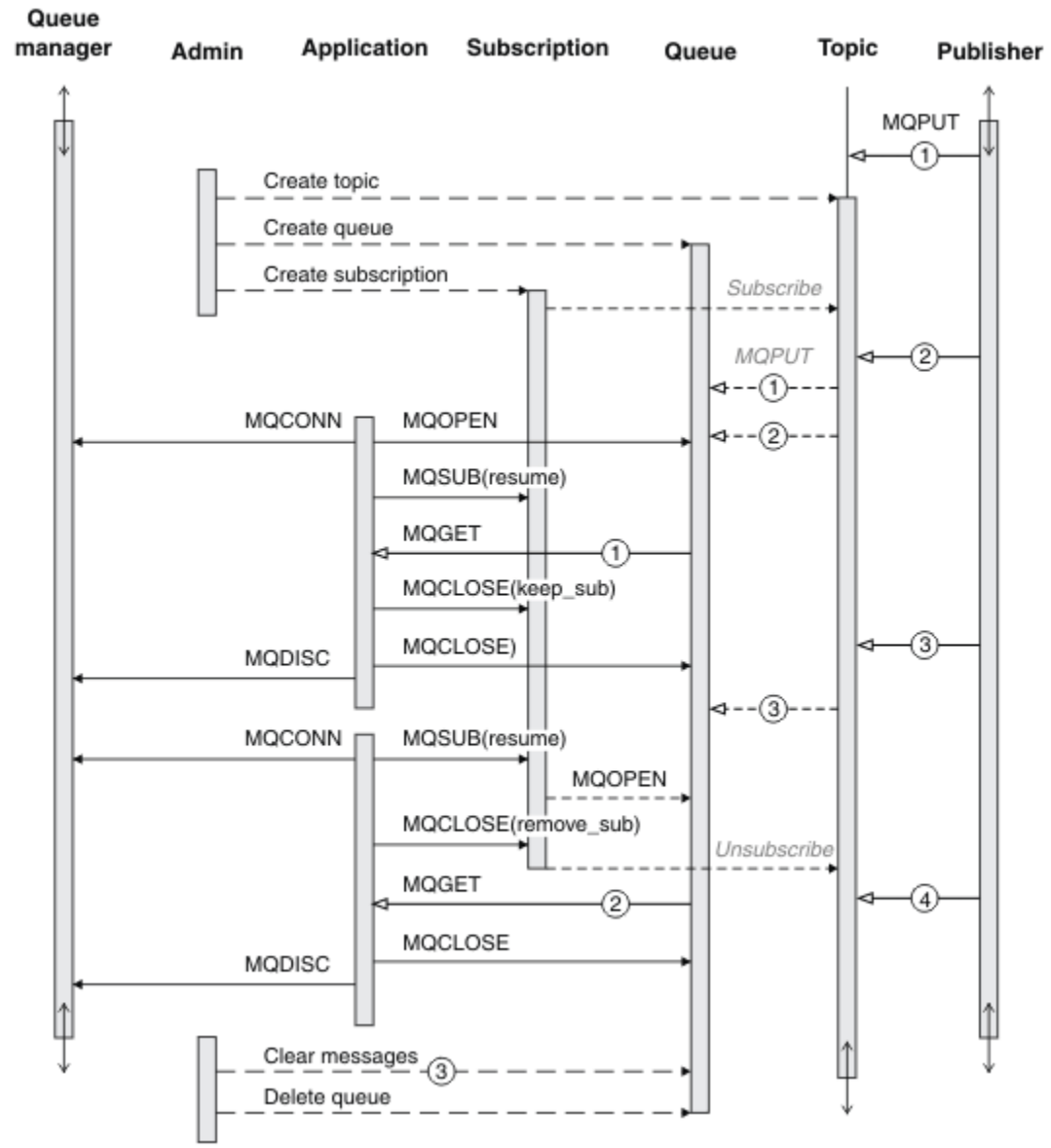
Üçüncü örneğe bir yönetici eklenir: yönetilmeyen sürekli abone. Yöneticinin bir yayınlama/abone olma uygulamasıyla nasıl etkileşimde bulunabileceğini göstermek için iyi bir örnektir.

Not alınacak noktalar listelenir.

1. Yayınlayıcı, daha sonra abonelik için kullanılan konu nesnesiyle ilişkilendirilecek bir konuya 1iletisini koyar. Konu nesnesi, genel arama karakterleri kullanılarak yayınlanan konuyla eşleşen bir konu dizgisi tanımlar.
2. Konu, korunan bir yayına sahiptir.
3. Denetimci bir konu nesnesi, kuyruk ve abonelik yaratır. Abonelikten önce konu nesnesinin ve kuyruğunun tanımlanması gerekir.
4. Uygulama, abonelikle ilişkili kuyruğu açar ve kuyruğun tanıtıcısını `MQSUB` iletir. Diğer bir seçenek olarak, aboneliği açarak `MQHO_NONE`kuyruk tanıtıcısını geçirebilir. Tersi doğru değil, bir aboneliği yalnızca kuyruk tanıtıcısını abonelik adı olmadan geçirerek sürdüremez-bir kuyruğun birden çok aboneliği olabilir.
5. Uygulama, aboneliği ilk kez açmasına rağmen `MQSO_RESUME` seçeneğini kullanarak aboneliği açar. Yönetimsel olarak oluşturulan bir aboneliğe devam ediyor.
6. Abone, alıkonan yayını (1) alır. Yayın 2, abone tarafından herhangi bir yayın alınmadan önce yayınlanmasına rağmen, abonelik başlatıldıktan sonra yayınlandı ve abonelik kuyruğundaki ikinci yayındır.

Not: Alıkonan yayın kalıcı bir ileti olarak yayınlanmazsa, kuyruk yöneticisi yeniden başlatıldıktan sonra kaybolur.

7. Bu örnekte abonelik dayanıklıdır. Bir programın yönetilmeyen, sürekli olmayan bir abonelik oluşturması mümkündür; bunun bir yöneticinin yapabileceği bir şey olmadığı açık olmalıdır.
8. MQCO_REMOVE_SUB seçeneğinin aboneliği kapatma üzerindeki etkisi, yönetici aboneliği silmiş gibi aboneliği kaldırmaktır. Bu, kuyruğa gönderilen diğer yayınları durdurur, ancak kuyruk kapatıldığında bile, *yönetilen* sürekli abonelikten farklı olarak, kuyruktaki yayınları etkilemez.
9. Yönetici daha sonra kalan 3iletisini siler ve kuyruğu siler.



Şekil 85. Yönetilmeyen sürekli abone ömür çizgisi

Yönetilmeyen bir abonelik için normal bir kalıp, yönetici tarafından gerçekleştirilecek kuyruk ve abonelik bakımıdır. Genellikle, yönetilen bir abonenin davranışını taklit etme ve uygulama kodunda kuyrukları ve abonelikleri programlı olarak toparlama girişiminde bulunulamaz. Kendinizi yönetim mantığı yazmak zorunda bulursanız, yönetilen bir kalıbı kullanarak aynı sonuçları elde edip edemeyeceğinizi sorun. Sıkıca senkronize edilmiş, tamamen güvenilir bir yönetim kodu yazmak kolay değildir. Daha sonra, iletilerin, aboneliklerin ve kuyrukların durumlarından bağımsız olarak silinebildiğinden emin olduğunuzda, el ile ya da otomatik bir yönetim programı kullanarak toparlamak daha kolay olur.

İleti özelliklerini yayınla/abone ol

IBM MQ yayınlama/abone olma ileti sistemiyle ilgili bazı ileti özellikleri vardır.

PubAccountingSimgesi

Bu değer, bu abonelikle eşleşen tüm yayın iletilerinin İleti Tanımlayıcı 'nın (MQMD) AccountingToken alanında olacak değerdir. AccountingToken , iletinin kimlik bağlamının bir parçasıdır. İleti bağlamıyla ilgili daha fazla bilgi için bkz. “İleti bağlamı” sayfa 45. MQMD ' deki AccountingToken alanıyla ilgili daha fazla bilgi için bkz. AccountingToken.

PubApplIdentityData

Bu değer, bu abonelikle eşleşen tüm yayın iletilerinin İleti Tanımlayıcı 'nın (MQMD) ApplIdentityVeri alanında yer alacak değerdir. ApplIdentityVerileri, iletinin kimlik bağlamının bir parçasıdır. İleti bağlamıyla ilgili daha fazla bilgi için bkz. “İleti bağlamı” sayfa 45. MQMD ' deki ApplIdentityData alanı hakkında daha fazla bilgi için bkz. ApplIdentityData.

MQSO_SET_IDENTITY_CONTEXT seçeneği belirtilmezse, varsayılan bağlam bilgileri olarak, bu abonelik için yayınlanan her iletide ayarlanacak ApplIdentityverileri boş olur.

MQSO_SET_IDENTITY_CONTEXT seçeneği belirtilirse, PubApplIdentityData kullanıcı tarafından üretilir ve bu alan, bu abonelik için her yayında ayarlanacak ApplIdentityverilerini içeren bir giriş alanıdır.

PubPriority

Bu değer, bu abonelikle eşleşen tüm yayın iletilerinin İleti Tanımlayıcı 'nın (MQMD) Öncelik alanında olacak değerdir. MQMD ' deki Öncelik alanıyla ilgili daha fazla bilgi için Önceliköncelikbaşlıklı konuya bakın.

Değer sıfırdan büyük ya da sıfıra eşit olmalıdır; sıfır en düşük önceliktir. Aşağıdaki özel değerler de kullanılabilir:

- MQPRI_PRIORITY_AS_Q_DEF-MQSUB çağrısında Hobj alanında bir abonelik kuyruğu sağlandığında ve yönetilen bir tanıtıcı değilse, iletinin önceliği bu kuyruğun DefProirity özniteliğinden alınır. Belirlenen kuyruk bir küme kuyruğuysa ya da kuyruk adı çözme yolunda birden çok tanımlama varsa, yayın iletisi MQMD ' de Öncelik için açıklandığı gibi kuyruğa yerleştirildiğinde öncelik belirlenir. MQSUB çağrısı yönetilen bir tanıtıcı kullanıyorsa, iletinin önceliği, abone olunan konuyla ilişkilendirilmiş model kuyruğunun DefPriority özniteliğinden alınır.
- MQPRI_PRIORITY_AS_PUBLISHED-İletinin önceliği, özgün yayının önceliğidir. Bu, bu alanın ilk değeridir.

SubCorrelTanıtıcısı

Uyarı: bir ilinti tanıtıcısı yalnızca bir yayınlama/abone olma kümesindeki kuyruk yöneticileri arasında iletilebilir, bir sıradüzen arasında değil.

Bu abonelikle eşleşmesi için gönderilen tüm yayınlar, ileti tanımlayıcısında bu ilinti tanıtıcısını içerir. Birden çok abonelik yayınlarını almak için aynı kuyruğu kullanıyorsa, MQGET ilinti tanıtıcısıyla kullanılması yalnızca belirli bir aboneliğe ilişkin yayınların elde edilmesine izin verir. Bu ilinti tanıtıcısı kuyruk yöneticisi ya da kullanıcı tarafından oluşturulabilir.

MQSO_SET_CORREL_ID seçeneği belirtilmezse, ilinti tanıtıcısı kuyruk yöneticisi tarafından üretilir ve bu alan, bu abonelik için yayınlanan her iletide ayarlanacak ilinti tanıtıcısını içeren bir çıkış alanıdır.

MQSO_SET_CORREL_ID seçeneği belirtilirse, ilinti tanıtıcısı kullanıcı tarafından üretilir ve bu alan, bu abonelik için her yayında ayarlanacak ilinti tanıtıcısını içeren bir giriş alanıdır. Bu durumda, alan MQCI_NONE içeriyorsa, bu abonelik için yayınlanan her iletide ayarlanacak ilinti tanıtıcısı, iletinin özgün konmasıyla yaratılan ilinti tanıtıcısı olur.

MQSO_GROUP_SUB seçeneği belirtilirse ve belirtilen ilinti tanıtıcısı, aynı kuyruk ve çakışan bir konu dizgisi kullanılarak var olan bir gruplanmış abonelikle aynıysa, yayının bir kopyasıyla yalnızca gruptaki en önemli abonelik sağlanır.

SubUserVerileri

Bu, abonelik kullanıcı verileridir. Bu alandaki abonelikte sağlanan veriler, bu aboneliğe gönderilen her yayının MQSubUserveri iletisi özelliği olarak içerilir.

Yayın özellikleri

Çizelge 124 sayfa 805 içinde bir yayın iletisiyle birlikte sağlanan yayın özellikleri listelenir.

Bu özelliklere doğrudan **MQRFH2** klasöründen erişebilir ya da bunları `MQINQMP`kullanarak alabilirsiniz. `MQINQMP` , soracak özelliğin adı olarak özellik adını ya da **MQRFH2** adını kabul eder.

Çizelge 124. Yayın özellikleri

Özellik adı	MQRFH2 adı	Tip	Açıklama
`MQTopicString`	`mqps.Top`	`MQTYPE_STRING`	Konu dizgisi
`MQSubUserVerileri`	`mqps.Sud`	`MQTYPE_STRING`	Abone kullanıcı verileri
`MQIsRetained`	`mqps.Ret`	`MQTYPE_BOOLEAN`	Alıkonan yayın
`MQPubOptions`	`mqps.Pub`	`MQTYPE_INT32`	Yayınlama seçenekleri
`MQPubLevel`	`mqps.Pbl`	`MQTYPE_INT32`	Yayın düzeyi
`MQPubTime`	`mqpse.Pts`	`MQTYPE_STRING`	Yayın saati
`MQPubSeqNum`	`mqpse.Seq`	`MQTYPE_INT32`	Yayın sıra numarası
`MQPubStrIntData`	`mqpse.Sid`	`MQTYPE_STRING`	Yayınlayıcı tarafından eklenen dizgi/tamsayı verileri
`MQPubFormat`	`mqpse.Pfmt`	`MQTYPE_INT32`	İleti biçimi: MQRFH1 MQRFH2 PCF

İleti sıralaması

Belirli bir konu için, iletiler kuyruk yöneticisi tarafından, yayınlama uygulamalarından alındıkları sırayla yayınlanır (ileti önceliğine dayalı olarak yeniden sıralamaya tabidir).

İleti sıralaması olağan durumda, her abonenin belirli bir konu üzerindeki belirli bir kuyruk yöneticisinden, belirli bir yayınlayıcıdan, o yayıncı tarafından yayınlandıkları sırayla ileti aldığı anlamına gelir.

Ancak, tüm IBM MQ iletilerinde olduğu gibi, zaman zaman iletilerin sırasız olarak teslim edilmesi mümkündür. Bu, aşağıdaki durumlarda gerçekleşebilir:

- Ağdaki bir bağlantı kesilir ve sonraki iletiler başka bir bağlantı üzerinden yeniden yönlendirilirse
- Bir kuyruk geçici olarak dolursa ya da engellenirse, bir ileti teslim edilmeyen iletiler kuyruğuna konur ve bu nedenle geciktirilir, sonraki iletiler doğrudan geçer.
- Denetimci, yayınlayıcılar ve aboneler çalışmaya devam ederken bir kuyruk yöneticisini silerse, kuyruğa alınan iletilerin gönderilmeyen iletiler kuyruğuna konmasına ve aboneliklerin kesintiye uğramasına neden olur.

Bu koşullar gerçekleşmezse, yayınlar her zaman sırayla teslim edilir.

Not: Yayınlama/Abone Olma ile gruplanmış ya da bölümlenmiş iletiler kullanılamaz.

Yayınları engelleme

Bir yayını durdurabilir, değiştirebilir ve başka bir aboneye ulaşmadan önce yeniden yayınlayabilirsiniz.

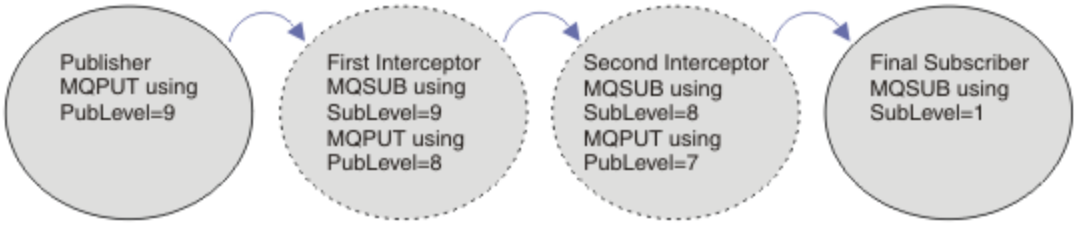
Aşağıdaki işlemlerden birini gerçekleştirmek için bir yayının bir aboneye ulaşmadan önce yayını kesmek isteyebilirsiniz:

- İletiye ek bilgi ekle
- İletiyi engelle
- İletiyi dönüştür

Her ileti üzerinde aynı işlemi gerçekleştirebilir ya da işlemi aboneliğe, iletiye ya da ileti üstbilgisine bağlı olarak değiştirebilirsiniz.

İlgili başvurular
MQ_PUBLISH_EXIT-Çıkış yayınla

Abonelik düzeyleri
Aboneliğin abonelik düzeyini, son abonelerine ulaşmadan önce bir yayını durduracak şekilde ayarlayın. Bir araya gelen abone, daha yüksek bir abonelik düzeyinde abone olur ve daha düşük bir yayın düzeyinde yeniden yayınlar. Son abonelere teslim edilmeden önce bir yayın üzerinde ileti işleme gerçekleştirmek için bir araya getirilmiş aboneler zinciri oluşturun.



Şekil 86. Kesişen abonelerin sırası

Bir yayını kesmek için **MQSD** `SubLevel` özniteliğini kullanın. Bir ileti alındıktan sonra, **MQPMO** `PubLevel` özniteliği değiştirilerek dönüştürülebilir ve daha sonra alt yayın düzeyinde yeniden yayınlanabilir. İleti daha sonra son abonelere gider ya da daha düşük bir abonelik düzeyinde bir ara abone tarafından yeniden algılanır.

Araya alan abone genellikle yeniden yayınlamadan önce bir iletiyi dönüştürür. Araya giren aboneler dizisi bir ileti akışı oluşturur. Diğer bir seçenek olarak, kesilen yayını yeniden yayınlamayabilirsiniz: Alt abonelik düzeylerindeki aboneler iletiyi almazlar.

Kesicinin diğer abonelerden önce yayınları aldığından emin olun. Kesicinin abonelik düzeyini diğer abonelerden daha yüksek bir değere ayarlayın. Varsayılan olarak, abonelerin `SubLevel` değeri 1olur. En yüksek değer 9. Bir yayın, en az en yüksek `SubLevel`kadar yüksek bir `PubLevel` ile başlamalıdır. Başlangıçta 9varsayılan `PubLevel` ile yayınlayın.

- Bir konuda tek bir araya gelen aboneniz varsa, `SubLevel` ' i 9olarak ayarlayın.
- Bir konuda birden çok araya gelen uygulama için, her art arda gelen önleme abonesi için daha düşük bir `SubLevel` ayarlayın.
- Abonelik seviyeleri 9 'den 2 ' e kadar (bu değerler de içinde olmak üzere) olmak üzere, en çok 8 önleme uygulaması uygulayabilirsiniz. İletinin son alıcısının `SubLevel` (Alt Düzey) 1.

Yayının `PubLevel` düzeyine eşit ya da bu düzeyden düşük abonelik düzeyine sahip kesici önce yayını alır. Belirli bir abonelik düzeyinde bir konu için yalnızca bir araya getirmeyi aboneyi yapılandırın. Belirli bir abonelik düzeyinde birden çok aboneye sahip olmak, yayının birden çok kopyasının abone olan uygulamaların son kümesine gönderilmesine neden olur.

`0` `SubLevel` olan bir abone, yakalama olarak kullanılır. Son abone iletiyi almazsa yayını alır. Başka hiçbir abonenin almadığı yayınları izlemek için `0` ürününün `SubLevel` düzeyine sahip bir abone kullanılabilir.

Durduran bir aboneyi programlama

Çizelge 125 sayfa 807içinde açıklanan abonelik seçeneklerini kullanın.

Çizelge 125. Aboneleri kesmeye ilişkin abonelik seçenekleri	
Abonelik seçeneği	**Notlar**
MQSO_SET_CORREL_ID ve SubCorrelTanıtıcısı MQCI_NONE olarak ayarlandı	Kesilen yayının CorrelId değerini özgün yayınla aynı tutun. **Not:** Bir sıradüzendeki bir yayının ilinti tanıtıcısını iletemezsiniz. Alan, kuyruk yöneticisi tarafından kullanılır.
PubPriority , MQPRI_PRIORITY_AS_PUBLISHED olarak ayarlandı	Kesilen yayının önceliğini, özgün yayının önceliğiyle aynı tutun.

Çizelge 125 sayfa 807 içindeki seçenekler, tüm dinlenen aboneler tarafından kullanılmalıdır. Sonuçta, ilinti tanıtıcısı ve ileti önceliği özgün yayınlayıcının ayarından değiştirilmez.

Dinleyen abone yayını işlediğinde, iletiyi kendi aboneliğinin SubLevel daha düşük bir PubLevel içinde aynı konuya yeniden yayınlar. Araya gelen abone 9 SubLevel ayarını tanımlarsa, iletiyi 8 PubLevel ile yeniden yayınlar.

İletiyi doğru şekilde yeniden yayınlamak için özgün yayından birkaç bilgi parçası gerekir. **MQMD** içindeki tüm bilgilerin sonraki aboneye iletildiğinden emin olmak için özgün iletideki **MQMD** ile aynı bilgileri yeniden kullanın ve MQPMO_PASS_ALL_CONTEXT ayarını belirleyin. Çizelge 126 sayfa 807 içinde gösterilen ileti özelliklerindeki değerleri, yeniden yayınlanan iletinin ilgili alanlarına kopyalayın. Araya gelen abone bu değerleri değiştirebilir. **MQPMO**' e ek değerler eklemek için OR işlecini kullanın. Koyma iletisi seçeneklerini birleştirmek için Seçenekler alanı.

Yönetilen yayın kuyruğunu kullanmak yerine, yayınlama kuyruğunu açık bir şekilde açmanız gerekir. Yönetilen kuyruk için MQSO_SET_CORREL_ID değerini ayarlayamazsınız. Yönetilen bir kuyrukta MQOO_SAVE_ALL_CONTEXT değerini ayarlayamazsınız. “Örnekler” sayfa 808içinde listelenen kod parçalarına bakın.

Çizelge 126. Yeniden yayınlanan iletilere ilişkin MQPUT değerleri	
MQPUT kullanarak iletiyi yeniden yayınla	**Yayın iletisindeki bilgiler**
MQOD. ObjectString	İleti Özelliği MQTopicString
MQPMO. Options	İleti Özelliği MQPubOptions

Son abone, abonelik seçeneklerini farklı ayarlama seçeneğine sahiptir. Örneğin, yayın önceliğini MQPRI_PRIORITY_AS_PUBLISHEDyerine belirtik olarak ayarlayabilir. Son abonenin ayarları yalnızca zincirdeki son yakalanan abonenin yayınını etkiler.

Yayınları alıkoyma

Alıkonan bir yayın, özgün koyma iletisi seçenekleri yeniden yayınlanan iletiye kopyalanarak engellendikten sonra korunmalıdır.

MQPMO_RETAIN seçeneği, yayınlayıcı tarafından ayarlanır. Her bir araya gelen abone, MQPubOptions ' i Çizelge 126 sayfa 807içinde gösterildiği gibi yeniden yayınlanan iletinin koyma iletisi seçeneklerine aktarmalıdır. Koyma iletisi seçeneklerinin kopyalanması, yayının alıkonup alıkonmayacağı da içinde olmak üzere, özgün yayınlayıcının belirlediği seçenekleri korur.

Bir yayın, engelleme aboneleri zincirinden aşağı geçişini bitirdiğinde ve nihai abonelere teslim edildiğinde, sonunda korunur. Alıkonan yayını isteyen SubLevel 1' deki yeni aboneler, daha fazla araya girmeden bu aboneyi alır. 1 düzeyinden büyük bir SubLevel abonelere alıkonan yayın gönderilmez. Sonuç olarak, alıkonan yayın, ikinci tur aboneler zinciri tarafından değiştirilmez.

Örnekler

Bu örnekler, kesilen bir abone oluşturmak için birleştirilebilecek kod parçalardır. Kod, üretim kalitesinden ziyade kısa olacak şekilde yazılır.

Şekil 87 sayfa 808 içindeki ön işlemci yönergeleri, MQINQMP MQI çağrısı için gerekli olan yayın iletilerinden alınacak iki özelliği tanımlar.

```
#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
```

Şekil 87. Ön işlemci yönergeleri

Şekil 88 sayfa 808 içinde kod parçalarında kullanılan bildirimler listelenir. Vurgulanan terimler dışında, bildirimler bir IBM MQ uygulaması için standarttır.

Vurgulanan Put ve Get seçenekleri, tüm bağlamı geçirmek için başlatılır. Vurgulanan MQTOPICSTRING ve MQPUBOPTIONS , önişlemci yönergelerinde tanımlanan özellik adları için MQCHARV kullanıma hazırlayıcılarıdır. Adlar MQINQMP' e iletilir.

```
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;
```

Şekil 88. Bildirimler

Bildirimlerde kolayca gerçekleştirilmeyen kullanıma hazırlama işlemleri Şekil 89 sayfa 809içinde gösterilir. Vurgulanan değerler açıklama gerektirir.

SYSTEM.NDURABLE.MODEL.QUEUE
Bu örnekte, yönetilen kalıcı olmayan bir aboneliği açmak için MQSUB kullanmak yerine, geçici bir dinamik kuyruk yaratmak için model kuyruğu (SYSTEM.NDURABLE.MODEL.QUEUE) kullanılır. Tutamacı MQSUB' e iletilir. Kuyruğu doğrudan açarak tüm ileti bağlamını kaydedebilir ve abonelik seçeneğini (MQSO_SET_CORREL_ID) ayarlayabilirsiniz.

MQGMO_CURRENT_VERSION
IBM MQ yapılarının çoğunun geçerli sürümünün kullanılması önemlidir. gmo.MsgHandle gibi alanlar yalnızca denetim yapılarının en son sürümünde kullanılabilir.

MQGMO_PROPERTIES_IN_HANDLE
Özgün yayında ayarlanan konu dizgisi ve koyma iletisi seçenekleri, ileti özellikleri kullanılarak araya alınan abone tarafından alınır. Diğer bir seçenek, iletideki **MQRFH2** yapısını doğrudan okumaktır.

MQSO_SET_CORREL_ID
MQSO_SET_CORREL_ID ile birlikte şunu kullanın:

```
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
```

Bu seçeneklerin etkisi, ilinti tanıtıcısını geçirmektir. Özgün yayınlayıcı tarafından ayarlanan ilinti tanıtıcısı, kesilen abone tarafından alınan yayının ilinti tanıtıcısı alanına yerleştirilir. Kesişen her abone, aynı ilinti tanıtıcısından geçer. Daha sonra son abone aynı ilinti tanıtıcısını alma seçeneğine sahiptir.

Not: Yayın bir yayınlama/abone olma sıradüzeninden geçirilirse, ilinti tanıtıcısı hiçbir zaman alıkonmaz.

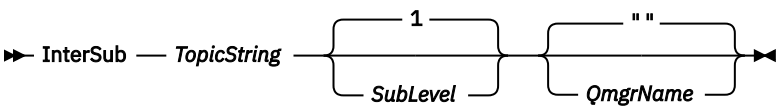
MQPRI_PRIORITY_AS_PUBLISHED
Yayın, yayınlandığı ileti önceliğiyle aynı yayın kuyruğuna yerleştirilir.

```
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;
```

Şekil 89. Kullanıma hazırlama

Şekil 90 sayfa 810 içinde komut satırı parametrelerini okumak için kod parçası gösterilir, kullanıma hazırlama işlemini tamamlanır ve durduran aboneliği yaratılır.

Programı şu komutla çalıştırın:



Hata işlemeyi mümkün olduğunca dikkat edilmez kılmak için, her MQI çağrısındaki neden kodu farklı bir dizi öğesinde saklanır. Her çağrıdan sonra tamamlama kodu sınanır ve değer MQCC_FAIL ise, denetim do { } while(0) kod öbeğinden çıkar.

Dikkate değer iki kod satırı,

pmo.PubLevel = sd.SubLevel - 1;
Yeniden yayınlanan iletiye ilişkin yayın düzeyini, durduran abonenin abonelik düzeyinden daha düşük bir düzeye ayarlar.

gmo.MsgHandle = Hmsg;
MQGET ' in ileti özelliklerini döndürmesi için bir ileti tanıtıcısı sağlar.

```
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;
```

Şekil 90. Yayınları engellemeye hazırlık

Şekil 91 sayfa 811ana kod parçası, yayınlama kuyruğundan iletileri alır. İleti özelliklerini sorgular ve konu dizesini ve özgün **MQPMO**' i kullanarak iletileri yeniden yayınlar. Yayının `seçenek` özellikleri.

Bu örnekte, yayın üzerinde dönüştürme gerçekleştirilmez. Yeniden yayınlanan yayının konu dizgisi, kesişen abonenin abone olduğu konu dizgisiyle her zaman eşleşir. Araya girme abonesi aynı yayın kuyruğuna gönderilen birden çok aboneliğe müdahale etmekten sorumluysa, farklı aboneliklerle eşleşen yayınları ayırt etmek için konu dizgisinin sorgulanması gerekebilir.

`MQINQMP` çağrıları vurgulanır. Konu dizgisi ve yayın koyma iletisi seçenekleri özellikleri doğrudan çıkış denetim yapılarına yazılır. `putOD.ObjectString` MQCHARV uzunluk alanını belirtik uzunluktan boş sonlandırılmış bir dizgiye değiştirmenin tek nedeni, dizgiyi çıkışa almak için `printf` kullanmaktır.

```
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;
}
```

Şekil 91. Yayını engelle ve yeniden yayınla

Son kod parçası Şekil 92 sayfa 811içinde gösterilir.

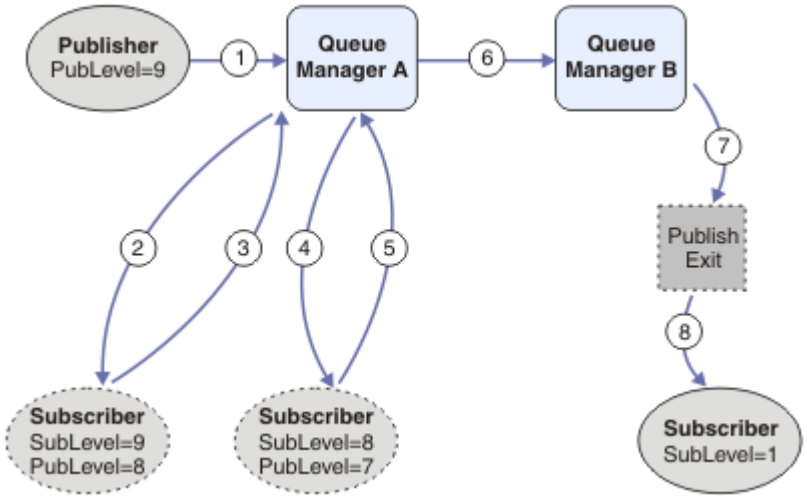
```
    } while (0);
    if (CompCode == MQCC_FAILED && Reason != MQRC_NO_MSG_AVAILABLE)
        printf("MQI Call failed with reason code %d\n", Reason);
    if (Hsub!= MQHO_NONE)
        MQCLOSE(Hcon, &Hsub, SC_Options, &CompCode, &Reason);
    if (Hcon!= MQHC_UNUSABLE_HCONN)
        MQDISC(&Hcon, &CompCode, &Reason);
}
```

Şekil 92. Tamamlama

Yayınları ve dağıtılmış yayınları engelleme/abone olma

Kesilen aboneleri dağıtırken ya da çıkışları dağıtılmış bir yayınlama/abone olma topolojisinde yayınlarken basit bir örüntüyü izleyin. Yayıncılarla aynı kuyruk yöneticilerine kesilen aboneleri konuşlandırın ve çıkışları son aboneler ile aynı kuyruk yöneticilerine yayınlayın.

Şekil 93 sayfa 812 , bir yayınlama abone olma kümesine bağlı iki kuyruk yöneticisini gösterir. Bir yayınlayıcı, 9yayın düzeyinde bir küme konusu için bir yayın yaratır. Numaralandırılmış oklar, küme konusuna abonelere akarken yayın tarafından atılan adımların sırasını gösterir. Yayın, `Alt Düzey 9` olan abone tarafından engellenir ve `Publevel` 8ile yeniden yayınlanır. Bu, `Alt düzey 8`' deki bir abone tarafından yeniden engellenir. Abone, `Publevel` 7adresinde yeniden yayınlar. Kuyruk yöneticisi tarafından sağlanan yetkili sunucu abonesi, yayını, son aboneye ek olarak bir Yayınlama çıkışının konuşlandırıldığı kuyruk yöneticisi B ' ye iletir. Yayınlama, son abone tarafından `Alt Düzey 1`' de alınmadan önce Yayınlama çıkışı tarafından işlenir. Kesilen aboneler ve yayınlama çıkışı bozuk anahatlarla gösterilir.



Şekil 93. Bir kümede önleme ve yayınlama çıkışı

Basit kalıbın amacı, bir yayın alan her abonenin aynı yayını almaktır. Yayın, abonenin bağlandığı yerden bağımsız olarak aynı dönüşüm sırasından geçer. Yayıncıların ya da son abonelerin bağlı olduğu yere bağlı olarak, dönüşümlerin dizisinin değişmesini önlemek istersiniz. Makul bir istisna, sonunda her bir aboneye teslim edilen yayının uyarlanmasıdır. Yayının son olarak teslim edildiği kuyruğa dayalı olarak yayını özelleştirmek için Yayınlama çıkışını kullanın.

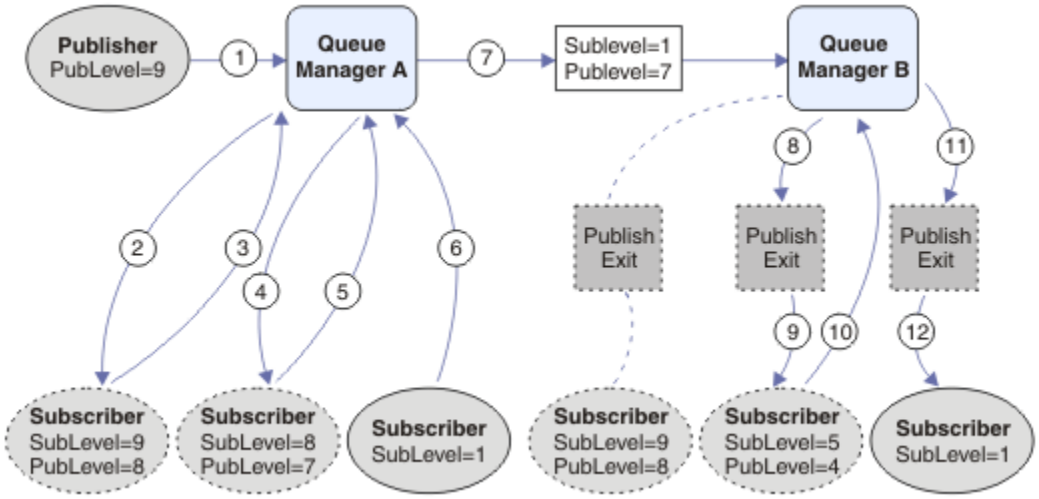
Kesilen abonelerin dağıtılmış bir yayınlama/abone olma topolojisinde nerede konuşlandırılacağını ve çıkışları yayınlamayı dikkatli bir şekilde düşünmeniz gerekir. Basit örüntü, aboneleri yayıncılarla aynı kuyruk yöneticisine konuşlandırır ve Yayınlama çıkışlarını son aboneler ile aynı kuyruk yöneticilerine yayınlar.

Kalıba karşı koruma

Şekil 94 sayfa 813 , basit bir örüntüyü izlemezseniz, işlerin nasıl ters gidebileceğini gösterir. Konuşlandırmayı karmaşıklaştırmak için, kuyruk yöneticisi A 'ya son bir abone eklenir ve kuyruk yöneticisi B' ye iki ek müdahale abonesi eklenir.

Yayın, `SubLevel` 1adresindeki son abone tarafından kullanılmadan önce `SubLevel` 5 adresindeki bir abone tarafından yakalandığı `PubLevel` 7adresindeki kuyruk yöneticisi B ' ye iletilir. Yayınlama çıkışı, yayını hem kesen tüketiciye hem de kuyruk yöneticisi B ' deki son tüketiciye iletilmeden önce yakalar. Yayın, Yayınlama çıkışı tarafından işlenmeden A kuyruk yöneticisindeki son aboneye ulaşır.

Bir yayınlama/abone olma topolojisinde, yetkili sunucu aboneleri `SubLevel` 1' e abone olur ve son önleme abonesi tarafından ayarlanan `PubLevel` öğesini geçirir. Şekil 94 sayfa 813içinde, sonuç olarak, yayının, B kuyruk yöneticisinde `SubLevel` 9 kullanılarak abone tarafından engellenmemesi ortaya çıkar.



Şekil 94. Kesişen abonelerin karmaşık devreye alımı

Yayın seçenekleri

İletilerin yayınlanma şeklini denetleyen birkaç seçenek vardır.

Yanıt bilgilerinin abonelerden saklanması

Abonelerin aldıkları yayınlara yanıt verebilmesini istemiyorsanız, MQPMO_SUPPRESS_REPLYTO koyma iletisi seçeneğini kullanarak MQMD ' nin ReplyToQ ve ReplyToQmgr alanlarındaki bilgileri gizleyebilirsiniz. Bu seçenek kullanılırsa, kuyruk yöneticisi yayını herhangi bir aboneye iletmeden önce MQMD ' den bu bilgileri kaldırır.

Bu seçenek, MQRC_MISSING_REPLY_TO_Q ile başarısız olarak çağrı girişiminde bulunulursa, ReplyToQ gerektiren bir rapor seçeneğiyle birlikte kullanılamaz.

Yayın düzeyi

Yayın düzeylerinin kullanılması, hangi abonelerin yayını alacağını denetlemenin bir yoludur. Yayın düzeyi, yayın tarafından hedeflenen abonelik düzeyini belirtir. Yalnızca en yüksek abonelik düzeyine sahip abonelikler, yayının yayın düzeyinden daha düşük ya da yayın düzeyine eşit olarak gönderilir. Bu değer sıfır ile dokuz aralığında olmalıdır; sıfır en düşük yayın düzeyidir. Bu alanın ilk değeri 9 'dur. Yayın ve abonelik düzeylerinin kullanımlarından biri, yayınları kesmektir.

Bir yayının herhangi bir aboneye teslim edilip edilmediği denetleniyor

Bir yayının herhangi bir aboneye teslim edilip edilmediğini denetlemek için, MQPUT çağrısıyla MQPMO_WARN_IF_NO_SUBS_EŞLEŞME koyma iletisi seçeneğini kullanın. Bir MQCC_WARNING tamamlanma kodu ve MQRC_NO_SUBS_ANCAK koyma işlemi tarafından döndürülürse, yayın hiçbir aboneliğe teslim edilmedi. Koyma işleminde MQPMO_RETAIN seçeneği belirtilirse, ileti alıkonur ve daha sonra tanımlanan herhangi bir eşleşen aboneliğe teslim edilir. Dağıtımlı bir yayınlama/abone olma sisteminde, MQRC_NO_SUBS_ANCAK kuyruk yöneticisinde bu konu için kayıtlı yetkili sunucu aboneliği yoksa, MQRC_NO_SUBS_ANCAK eşleştirilen neden kodu döndürülür.

Abonelik seçenekleri

İleti aboneliklerinin nasıl işleneceğini denetleyen birkaç seçenek vardır.

İleti kalıcılığı

Kuyruk yöneticileri, yayıncı tarafından belirlenen şekilde abonelere ilettikleri yayınların sürekliliğini sağlar. Yayınlayıcı kalıcılığı aşağıdaki seçeneklerden biri olarak ayarlar:

0
Kalıcı olmayan

1
Kalıcı

2
Kuyruk/konu tanımlaması olarak kalıcılık

Yayınlama/abone olma için, yayınlayıcı konu nesnesini ve **topicString** öğesini çözümlenmiş bir konu nesnesine çözer. Yayınlayıcı kuyruk/konu tanımlaması olarak Persistence belirtiyorsa, çözülen konu nesnesindeki varsayılan kalıcılık yayın için ayarlanır.

Yayınları alıkoyma

Alıkonan yayınların ne zaman alınacağını denetlemek için aboneler iki abonelik seçeneği kullanabilir:

Yalnızca istek üzerine yayınla, MQSO_PUBLICATIONS_ON_REQUEST

Bir abonenin yayınları ne zaman alacağını denetlemesine sahip olmasını istiyorsanız, MQSO_PUBLICATIONS_ON_REQUEST abonelik seçeneğini kullanabilirsiniz. Bir abone, bir konunun alıkonan yayınının gönderilmesini istemek için MQSUBRQ çağrısını (özgün MQSUB çağrısından döndürülen Hsub tanıtıcı değerini belirterek) kullanarak yayınları ne zaman alacağını denetleyebilir. MQSO_PUBLICATIONS_ON_REQUEST abonelik seçeneğini kullanan aboneler, alıkonmayan yayınlar almaz.

MQSO_PUBLICATIONS_ON_REQUEST belirtirseniz, herhangi bir yayını almak için MQSUBRQ kullanmalısınız. MQSO_PUBLICATIONS_ON_REQUEST kullanmıyorsanız, iletileri yayınlandıkça alıyorsunuz.

Bir abone MQSUBRQ aramasını kullanıyorsa ve aboneliğin konusunda genel arama karakterleri kullanıyorsa, abonelik bir konu ağacındaki birden çok konu ya da düğümle eşleşebilir; bunların tümü alıkonan iletilerle (varsa) aboneye gönderilir.

Bu seçenek, sürekli aboneliklerle kullanıldığında özellikle yararlı olabilir; çünkü bir kuyruk yöneticisi, abone uygulaması çalışmasa bile aboneye sürekli olarak abone olursa, yayınları göndermeye devam eder. Bu, abone kuyruğunda ileti birikmesine neden olabilir. Abone, MQSO_PUBLICATIONS_ON_REQUEST seçeneğini kullanarak kaydolursa bu oluşturma önlenebilir. Alternatif olarak, istenmeyen iletilerin oluşturulmasını önlemek için uygulamanız için uygunsa, kalıcı olmayan abonelikler kullanabilirsiniz.

Bir abonelik dayanıklıysa ve bir yayınlayıcı alıkonan yayınları kullanıyorsa, abone uygulaması yeniden başlatmadan sonra durum bilgilerini yenilemek için MQSUBRQ çağrısını kullanabilir. Daha sonra abone, MQSUBRQ çağrısını kullanarak durumunu düzenli olarak yenilemelidir.

Bu seçenek kullanılarak MQSUB çağrısının sonucu olarak hiçbir yayın gönderilmez. Özgün abonelik bu seçeneği kullanacak şekilde yapılandırıldıysa, bağlantıyı kesmenin ardından sürdürülebilir bir abonelik MQSO_PUBLICATIONS_ON_REQUEST seçeneğini kullanır.

Yalnızca yeni yayınlar, MQSO_NEW_PUBLICATIONS_ONLY

Bir konuda alıkonan bir yayın varsa, yayın yapıldıktan sonra abonelik yapan tüm aboneler o yayının bir kopyasını alır. Bir abone, yapılmakta olan abonelikten daha önce yapılmış herhangi bir yayını almak istemiyorsa, abone MQSO_NEW_PUBLICATIONS_ONLY abonelik seçeneğini kullanabilir.

Abonelikleri gruplama

Yayınları almak için bir kuyruk ayarladıysanız ve aynı kuyruğa çakışan yayınları besleyen abonelikleriniz varsa, abonelikleri gruplamayı düşünebilirsiniz. Bu durum, Aboneliklerin çakışmasıörneğine benzer.

Bir konuya abone olduğunuzda MQSO_GROUP_SUB seçeneğini ayarlayarak yinelenen yayınlar almayı önleyebilirsiniz. Sonuç olarak, gruptaki birden fazla abonelik bir yayının konusuyla eşleştiğinde, yayının kuyruğa yerleştirilmesinden yalnızca bir abonelik sorumlu olur. Yayın konusunun eşleştiği diğer abonelikler yoksayılır.

Yayını kuyruğa yerleştirmekten sorumlu abonelik, herhangi bir joker karakterle karşılaşmadan önce en uzun eşleşen konu dizesine sahip olduğu temelinde seçilir. En yakın eşleşen abonelik olarak düşünülebilir.

Özellikleri, MQSO_NOT_OWN_PUBS özelliğine sahip olup olmadığı da dahil olmak üzere yayına yayılır. Bu durumda, diğer eşleşen aboneliklerin MQSO_NOT_OWN_PUBS özelliği olmasa da, kuyruğa hiçbir yayın teslim edilmez.

Yinelenen yayınları ortadan kaldırmak için tüm aboneliklerinizi tek bir gruba koyamazsınız. Gruplanmış abonelikler şu koşulları karşılamalıdır:

1. Aboneliklerin hiçbiri yönetilmiyor.
2. Bir grup abonelik, yayınları aynı kuyruğa teslim ediyor.
3. Her abonelik aynı abonelik düzeyinde olmalıdır.
4. Gruptaki her abonelik için yayın iletisi aynı ilinti tanıtıcısına sahip.

   Her aboneliğin aynı ilinti tanıtıcısına sahip bir yayın iletisiyle sonuçlandığından emin olmak için, MQSO_SET_CORREL_ID değerini, yayında kendi ilinti tanıtıcınızı yaratacak şekilde ayarlayın ve her abonelikte **SubCorrelId** alanında aynı değeri ayarlayın. **SubCorrelId** değerini MQCI_NONEdeğerine ayarlamayın.

Ek bilgi için bkz. ../refdev/q100080_.dita#q100080_/mqso_group_sub .

Nesne öznitelikleri hakkında sorma ve bunları ayarlama

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

Bunlar, bir kuyruk yöneticisinin bir nesneyi işleme şeklini etkiler. Her IBM MQ nesnesi tipinin öznitelikleri, Nesnelerin öznitelikleriikonusunda ayrıntılı olarak açıklanmıştır.

Bazı öznitelikler nesne tanımlandığında ayarlanır ve yalnızca IBM MQ komutları kullanılarak değiştirilebilir; böyle bir özniteliğin örneği, kuyruğa konan iletiler için varsayılan önceliktir. Diğer öznitelikler kuyruk yöneticisinin işleminden etkilenir ve zaman içinde değişebilir; örneğin, bir kuyruğun yürürlükteki derinliği.

MQINQ çağrıyı kullanarak çoğu özniteliğin yürürlükteki değerlerini sorgulayabilirsiniz. MQI, bazı kuyruk özniteliklerini değiştirebileceğiniz bir MQSET çağrısı da sağlar. Başka bir nesne tipinin özniteliklerini değiştirmek için MQI çağrılarını kullanamazsınız. Bunun yerine aşağıdaki kaynaklardan birini kullanmalısınız:

- ALW MQSC komutlarındaaçıklanan MQSC olanağı.
- IBM i IBM i için CL komutları başvurusuya da MQSC olanağında açıklanan CHGMQMx CL komutları.
- z/OS MQSC komutlarındaaçıklanan ALTER işleci komutları ya da REPLACE seçeneğiyle DEFINE komutları.

Not: Nesnelerin özniteliklerinin adları bu belgede, bunları MQINQ ve MQSET çağrılarıyla kullandığınız formda gösterilir. Öznitelikleri tanımlamak, değiştirmek ya da görüntülemek için IBM MQ komutlarını kullandığınızda, konu bağlantılarındaki komutların tanımlamalarında gösterilen anahtar sözcükleri kullanarak öznitelikleri tanımlamanız gerekir.

Hem MQINQ hem de MQSET çağrıları, seçicilerin dizilerini kullanarak hakkında bilgi almak ya da ayarlamak istediğiniz öznitelikler. Çalışabileceğiniz her öznitelik için bir seçici vardır. Seçici adının, özniteliğin niteliğine göre belirlenen bir öneki var:

Çizelge 127. Seçici adlarına ilişkin önekler

Önek	**Açıklama**
MQCA_	Bu seçiciler, karakter verileri içeren özniteliklere (örneğin, bir kuyruğun adı) başvurur.

Çizelge 127. Seçici adlarına ilişkin önekler (devamı var)	
Önek	**Açıklama**
MQIA_	These selectors refer to attributes that contain either numeric values (such as *CurrentQueueDepth*, the number of messages on a queue) or a constant value (such as *SyncPoint*, whether the queue manager supports syncpoints).

MQINQ ya da MQSET çağrılarını kullanmadan önce, uygulamanızın kuyruk yöneticisine bağlı olması ve öznitelikleri ayarlamak ya da sorgulamak üzere nesneyi açmak için MQOPEN çağrıyı kullanmanız gerekir. Bu işlemler "Kuyruk yöneticisine bağlanma ve bağlantı kesme" sayfa 705 ve "Nesnelerin açılması ve kapatılması" sayfa 712içinde açıklanır.

Nesne öznitelikleri hakkında soru sorma ve bunları ayarlama hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- "Bir nesnenin özniteliklerini sorma" sayfa 816
- "MQINQ çağrılarının başarısız olduğu bazı durumlar" sayfa 817
- "Kuyruk özniteliklerinin ayarlanması" sayfa 818

İlgili kavramlar

"İleti Kuyruğu Arabirimine Genel Bakış" sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

"Kuyruk yöneticisine bağlanma ve bağlantı kesme" sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

"Nesnelerin açılması ve kapatılması" sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

"Kuyruktaki iletilerin konması" sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

"Kuyruktan ileti alma" sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

"İş birimlerinin kesinleştirilmesi ve geri çekilmesi" sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

"Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması" sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

"MQI ve kümelerle çalışma" sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

"IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma" sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

"IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları" sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

Bir nesnenin özniteliklerini sorma

Herhangi bir IBM MQtipinin öznitelikleri hakkında bilgi almak için MQINQ çağrısının kullanın.

Bu aramaya giriş olarak şunları sağlamanız gerekir:

- Bağlantı tanıtıcısı.

- Bir nesne tanıtıcısı.
- Seçici sayısı.
- Her seçici MQCA_ * ya da MQIA_ * biçiminde olan öznitelik seçicilerden oluşan bir dizi. Her seçici, hakkında bilgi almak istediğiniz bir değeri olan bir özniteliği temsil eder ve her seçici, nesne tanıtıcısının temsil ettiği nesne tipi için geçerli olmalıdır. Seçicileri herhangi bir sırada belirtebilirsiniz.
- Hakkında bilgi sormakta olduğunuz tamsayı özniteliklerinin sayısı. Tamsayı öznitelikleri hakkında soru sormuyorsanız sıfır belirtin.
- *CharAttrLengthi*çindeki karakter öznitelikleri arabelleğinin uzunluğu. Bu, en azından her karakter özniteliği dizgisini tutmak için gereken uzunluklar toplamı olmalıdır. Karakter öznitelikleri hakkında soru sormuyorsanız sıfır belirtin.

MQINQ çıkışı:

- Diziye kopyalanan bir tamsayı öznitelik değerleri kümesi. Değerlerin sayısı *IntAttrCount*tarafından belirlenir. *IntAttrCount* ya da *SelectorCount* sıfırsa, bu parametre kullanılmaz.
- Karakter öznitelik değerlerinin döndürüldüğü arabellek. Arabelleğin uzunluğu **CharAttrLength** değiştirgesiyle verilir. *CharAttrLength* ya da *SelectorCount* sıfırsa, bu parametre kullanılmaz.
- Bir tamamlanma kodu. Tamamlama kodu bir uyarı verirse, bu, aramanın yalnızca kısmen tamamlandığı anlamına gelir. Bu durumda, neden kodunu inceleyin.
- Bir neden kodu. Üç kısmi tamamlanma durumu vardır:
  - Seçici kuyruk tipi için geçerli değil
  - Tamsayı öznitelikleri için yeterli yer yok
  - Karakter öznitelikleri için yeterli yer yok

  Bu durumların birden fazlası ortaya çıkarsa, geçerli olan ilk durum döndürülür.

Çıkış ya da sorgu için bir kuyruk açarsanız ve bu kuyruk yerel olmayan bir küme kuyruğuna çözülüyorsa, yalnızca kuyruk adını, kuyruk tipini ve ortak öznitelikleri sorgulayabilirsiniz. MQOO_BIND_ON_OPEN kullanıldıysa, ortak özniteliklerin değerleri, seçilen kuyruğun değerleridir. Değerler, MQOO_BIND_NOT_FIXED ya da MQOO_BIND_ON_GROUP kullanıldıysa ya da MQOO_BIND_AS_Q_DEF kullanıldıysa ve **DefBind** kuyruk özniteliği MQBND_BIND_NOT_FIXED ise, olası küme kuyruklarından birinin değerleridir. Daha fazla bilgi için bkz. “MQOPEN ve kümeler” sayfa 848 ve MQOPEN .

Not: Çağrı tarafından döndürülen değerler, seçilen özniteliklerin anlık görüntüsüdür. Programınız döndürülen değerler üzerinde işlem yapmadan önce öznitelikler değişebilir.

MQINQiçinde MQINQ çağrısının bir açıklaması vardır.

MQINQ çağrılarının başarısız olduğu bazı durumlar

Özniteliklerini sorgulamak için bir diğer ad açarsanız, temel kuyruktaki öznitelikleri değil, diğer ad kuyruğunun özniteliklerini (başka bir kuyruğa erişmek için kullanılan IBM MQ nesnesi) döndürürsünüz.

Ancak, diğer adın çözüldüğü temel kuyruk tanımlaması kuyruk yöneticisi tarafından da açılır ve başka bir program MQOPEN ve MQINQ çağrılarınız arasındaki aralıkta temel kuyruk kullanımını değiştirirse, MQINQ çağrınız başarısız olur ve MQRC_OBJECT_CHANGED neden kodunu döndürür. Diğer ad kuyruğu nesnesinin öznitelikleri değiştirildiğinde de çağrı başarısız olur.

Benzer şekilde, öznitelikleri hakkında bilgi almak üzere bir uzak kuyruk açtığınızda, yalnızca uzak kuyruğun yerel tanımının öznitelikleri döndürülür.

Sormakta olduğunuz kuyruk öznitelikleri tipi için geçerli olmayan bir ya da daha çok seçici belirtirseniz, MQINQ çağrısı bir uyarıyla tamamlanır ve çıkışı aşağıdaki gibi ayarlar:

- Tamsayı öznitelikleri için, *IntAttrs* ile ilgili öğeler MQIAV_NOT_UYGULANABILIR olarak ayarlanır.
- Karakter öznitelikleri için, *CharAttrs* dizgisinin karşılık gelen bölümleri yıldız işaretlerine ayarlanır.

Sorguladığınız nesne öznitelikleri tipi için geçerli olmayan bir ya da daha çok seçici belirtirseniz, MQINQ çağrısı başarısız olur ve MQRC_SELECTOR_ERROR neden kodunu döndürür.

Bir model kuyruğuna bakmak için MQINQ ' yu çağıramazsınız; MQSC olanağını ya da altyapınızda var olan komutları kullanın.

Kuyruk özniteliklerinin ayarlanması

MQSET çağrısı kullanılarak kuyruk özniteliklerinin nasıl ayarlanacağını öğrenmek için bu bilgileri kullanın.

MQSET çağrıyı kullanarak yalnızca aşağıdaki kuyruk özniteliklerini ayarlayabilirsiniz:

- *InhibitGet* (uzak kuyruklar için değil)
- *DistList* (z/OS üzerinde değil)
- *InhibitPut*
- *TriggerControl*
- *TriggerType*
- *TriggerDepth*
- *TriggerMsgPriority*
- *TriggerData*

MQSET çağrısı, MQINQ çağrısıyla aynı parametrelere sahip. Ancak, MQSET için, tamamlanma kodu ve neden kodu dışındaki tüm değiştirgeler giriş değiştirgeleridir. Kısmi tamamlanma durumu yok.

Not: Yerel olarak tanımlanan kuyruklar dışındaki IBM MQ nesnelerinin özniteliklerini ayarlamak için MQI ' ı kullanamazsınız.

MQSET çağrısına ilişkin daha fazla ayrıntı için bkz. MQSET.

İş birimlerinin kesinleştirilmesi ve geri çekilmesi

Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

Bu konuda aşağıdaki terimler kullanılır:

- Kesinleştir
- İptal Et
- Eşitleme noktası koordinasyonu
- Eşitleme Noktası
- İş birimi
- Tek aşamalı kesinleştirme
- İki aşamalı kesinleştirme

Bu işlem işleme terimlerini biliyorsanız, “IBM MQ uygulamalarında eşitleme noktası ile ilgili önemli noktalar” sayfa 820başlıklı konuya atlayabilirsiniz.

Kesinleştir ve geri çekil
Bir program bir iş birimi içindeki bir kuyruğa ileti koyduğunda, bu ileti yalnızca program iş birimini kesinleştirdiğinde diğer programlar tarafından görülebilir. Bir iş birimini kesinleştirmek için, veri bütünlüğünü korumak için tüm güncellemelerin başarılı olması gerekir. Program bir hata saptar ve koyma işleminin kalıcı olmadığına karar verirse, iş birimini geri alabilir. Bir program geriletme işlemi gerçekleştirdiğinde, IBM MQ o iş birimi tarafından kuyruğa konan iletileri kaldırarak kuyruğu geri yükler. Programın kesinleştirme ve geri alma işlemlerini nasıl gerçekleştireceği, programın çalıştığı ortama bağlıdır.

Benzer şekilde, bir program iş birimi içindeki bir kuyruktan ileti aldığında, program iş birimini kesinleştirinceye kadar bu ileti kuyrukta kalır, ancak ileti başka programlar tarafından alınamaz. Program iş birimini kesinleştirdiğinde ileti kuyruktan kalıcı olarak silinir. Program iş birimini geri çevirirse, IBM MQ diğer programlar tarafından alınabilecek iletileri kullanarak kuyruğu geri yükler.

Syncpoint koordinasyonu, syncpoint, iş birimi
Syncpoint koordinasyonu , iş birimlerinin kesinleştirildiği ya da veri bütünlüğüyle geriletildiği süreçtir.

Değişiklikleri kesinleştirme ya da geri alma kararı, en basit durumda, bir işlemin sonunda alınır. Ancak, bir uygulamanın bir hareket içindeki diğer mantıksal noktalarda veri değişikliklerini eşitlemesi daha yararlı olabilir. Bu mantıksal noktalara *syncpoints* (ya da *eşitleme noktaları*) denir ve iki eşitleme noktası arasındaki bir güncelleme kümesini işleme süresi *iş birimi*olarak adlandırılır. Birden çok MQGET çağrısı ve MQPUT çağrısı tek bir iş biriminin parçası olabilir.

Bir iş birimindeki ileti sayısı üst sınırı, ALTER QMGR komutunun MAXUMSGS özniteliği tarafından denetlenebilir.

Tek aşamalı kesinleştirme
Tek fazlı kesinleştirme işlemi, bir programın diğer kaynak yöneticileriyle değişiklikleri koordine etmeden bir kuyrukta güncellemeleri kesinleştirebileceği bir işlemdir.

İki aşamalı kesinleştirme
İki aşamalı kesinleştirme işlemi, bir programın IBM MQ kuyruklarında yaptığı güncellemelerin diğer kaynaklarda yapılan güncellemelerle (örneğin, Db2 denetimi altındaki veritabanları) eşgüdümlenebildiği bir işlemdir. Böyle bir süreç altında, tüm kaynaklara ilişkin güncellemeler kesinleştirilebilir ya da birlikte geriletilebilir.

IBM MQ , iş birimlerini işlemeye yardımcı olmak için **BackoutCount** özniteliğini sağlar. Bu, bir iş birimi içindeki bir iletinin her geriletilmesinde artırılır. İleti yinelenen bir şekilde iş biriminin olağandışı bitmesine neden olursa, *BackoutCount* nihayet değeri *BackoutThreshold*değerini aşar. Bu değer, kuyruk tanımlandığında belirlenir. Bu durumda, uygulama iletiyi iş biriminden kaldırabilir ve *BackoutRequeueQName*içinde tanımlandığı şekilde başka bir kuyruğa yerleştirebilir. İleti taşındığında, iş birimi kesinleştirilebilir.

İş birimlerinin kesinleştirilmesine ve geri çekilmesine ilişkin ek bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “IBM MQ uygulamalarında eşitleme noktası ile ilgili önemli noktalar” sayfa 820
- z/OS “IBM MQ for z/OS uygulamalarında eşitleme noktaları” sayfa 821
- IBM i “IBM i uygulamaları için CICS içindeki eşitleme noktaları” sayfa 823
- “IBM MQ for Multiplatforms içinde eşitleme noktaları” sayfa 823
- IBM i “IBM i dış eşitleme noktası yöneticisine ilişkin arabirimler” sayfa 828

İlgili kavramlar
“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

“Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

“Kuyruktaki iletilerin konması” sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

“Kuyruktan ileti alma” sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

“Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

“MQI ve kümelerle çalışma” sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

“IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

“IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları” sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

IBM MQ uygulamalarında eşitleme noktası ile ilgili önemli noktalar

IBM MQ uygulamalarında eşitleme noktalarını kullanma hakkında bilgi edinmek için bu bilgileri kullanın.

İki aşamalı kesinleştirme aşağıdaki ortamlar tarafından desteklenir:

- Multi IBM MQ for Multiplatforms
- z/OS CICS Hareket Sunucusu- z/OS
- z/OS TXSeries
- z/OS IMS/ESA
- z/OS RRS ile z/OS toplu iş
- X/Open XA arabirimini kullanan diğer dış eşgüdümcüler

Tek fazlı kesinleştirme aşağıdaki ortamlar tarafından desteklenir:

- Multi IBM MQ for Multiplatforms
- z/OS z/OS toplu

Dış arabirimler hakkında daha fazla bilgi için bkz. “Multiplatforms üzerinde dış syncpoint yöneticilerine yönelik arabirimler” sayfa 826ve The Open Group tarafından yayınlanan XA belgeleri *CAE Specification Distributed Transaction Processing: The XA Specification*. İşlem yöneticileri (CICS, IMS, Encina ve Tuxedo gibi), kurtarılabilir diğer kaynaklarla koordine edilen iki aşamalı kesinleştirmede yer alabilir. Bu, IBM MQ tarafından sağlanan kuyruğa alma işlevlerinin, hareket yöneticisi tarafından yönetilen bir iş birimi kapsamında getirilebileceği anlamına gelir.

IBM MQ ile verilen örnekler, XA uyumlu veritabanlarını koordine eden IBM MQ ' i gösterir. Bu örneklerle ilgili daha fazla bilgi için bkz. “IBM MQ örnek yordam programlarının kullanılması” sayfa 1013.

IBM MQ uygulamanızda, her koymada ve alma çağrısında, aramanın syncpoint denetimi altında olmasını isteyip istemediğinizi belirtebilirsiniz. Bir koyma işleminin syncpoint denetimi altında çalışmasını sağlamak için, MQPMO yapısını çağırdığınızda MQPMO yapısının *Options* alanında MQPMO_SYNCPOINT değerini kullanın. Bir alma işlemi için, MQGMO yapısının *Options* alanında MQGMO_SYNCPOINT değerini kullanın. Belirtik olarak bir seçenek belirlemezseniz, varsayılan işlem altyapıya bağlıdır:

- Multi Eşitleme noktası denetimi varsayılan değeri NO değeridir.
- z/OS Eşitleme noktası denetimi varsayılan değeri YES değeridir.

MQPUT1 çağrısı MQPMO_SYNCPOINT ile yayınlandığında, koyma işleminin zamanuyumsuz olarak tamamlanması için varsayılan davranış değişir. Bu, döndürülen MQOD ve MQMD yapılarındaki belirli alanlara dayalı olan, ancak şimdi tanımlanmamış değerler içeren bazı uygulamaların davranışında bir değişikliğe neden olabilir. Bir uygulama, koyma işleminin zamanuyumlu olarak gerçekleştirildiğinden ve tüm uygun alan değerlerinin tamamlandığından emin olmak için MQPMO_SYNC_RESPONSE belirtilebilir.

Uygulamanız, eşitleme noktası altındaki bir MQPUT ya da MQGET yanıt olarak bir MQRC_BACKED_OUT neden kodu aldığında, uygulama olağan durumda MQBACK kullanarak yürürlükteki hareketi geri çevirmeli ve uygunsa, tüm hareketi yeniden denemelidir. Uygulama bir MQCMIT ya da MQDISC çağrısına yanıt olarak MQRC_BACKED_OUT alırsa, MQBACK çağrılması gerekmez.

Bir MQGET çağrısının her geriletilmesinde, etkilenen iletinin MQMD yapısının *BackoutCount* alanı artırılır. Yüksek bir *BackoutCount* , arka arkaya yedeklenen bir iletiyi gösterir. Bu, araştırmanız gereken bu iletiyle ilgili bir sorunu gösterebilir. *BackoutCount*ile ilgili ayrıntılar için bkz. BackoutCount .

RRS ile z/OS toplu işi dışında, bir program kesinleştirilmemiş istekler varken MQDISC çağrısıyla karşılaşırsa, örtük bir uyumlulaştırma noktası oluşur. Program olağandışı bir şekilde sona ererse, örtük bir geriletme oluşur.

z/OS z/OSüzerinde, program önce MQDISC çağrılmadan olağan bir şekilde sona ererse de örtük bir eşitleme noktası oluşur. MQ ' ya bağlı TCB normal şekilde sona ererse, bu programın olağan şekilde sona erdiği varsayılır. z/OS UNIX System Services ve Dil Ortamı (LE) altında çalışırken, olağandışı sonlanma ya da sinyaller için varsayılan koşul işleme çağrılır. LE koşul işleyicileri hata koşulunu işlerler ve TCB olağan şekilde sona erer. Bu koşullar altında MQ iş birimini kesinleştirir. Daha fazla bilgi için Introduction to Language Environment Condition Handlingbaşlıklı konuya bakın.

z/OS IBM MQ for z/OS programları için, geriletme oluştuğunda (*MQGET-error-backout* döngüsünü önlemek için) bir iletinin geriletilmemesi gerektiğini belirtmek üzere MQGMO_MARK_SKIP_BACKOUT seçeneğini kullanabilirsiniz. Bu seçeneği kullanma hakkında bilgi için bkz. “Geriletme atlanıyor” sayfa 766.

Kuyruk özniteliklerinde yapılan değişiklikler (MQSET çağrısıyla ya da komutlarla), iş birimlerinin kesinleştirilmesinden ya da geri çekilmesinden etkilenmez.

z/OS *IBM MQ for z/OS uygulamalarında eşitleme noktaları*

Bu konuda, hareket yöneticisinde (CICS ve IMS) eşitleme noktalarının nasıl kullanılacağı açıklanmaktadır ve toplu iş uygulamaları.

z/OS *CICS Transaction Server for z/OS uygulamalarında eşitleme noktaları*

Bir CICS uygulamasında EXEC CICS SYNCPOINT komutunu kullanarak bir eşitleme noktası oluşturabilirsiniz.

Önceki eşitleme noktasında yapılan tüm değişiklikleri geri almak için EXEC CICS SYNCPOINT ROLLBACK komutunu kullanabilirsiniz. Daha fazla bilgi için bkz. *CICS Application Programming Reference*.

İş biriminde kurtarılabilir başka kaynaklar varsa, kuyruk yöneticisi (CICS syncpoint yöneticisiyle birlikte) iki aşamalı bir kesinleştirme protokolüne katılır; tersi durumda, kuyruk yöneticisi tek aşamalı bir kesinleştirme işlemi gerçekleştirir.

Bir CICS uygulaması MQDISC çağrısını yaparsa, örtük bir eşitleme noktası alınmaz. Uygulama olağan bir şekilde kapanırsa, açık kuyruklar kapanır ve örtük kesinleştirme gerçekleşir. Uygulama olağandışı bir şekilde kapanırsa, açık kuyruklar kapanır ve örtük bir geriletme oluşur.

z/OS *IMS uygulamalarında eşitleme noktaları*

Bir IMS uygulamasında, IOPCB ve CHKP (denetim noktası) için GU (benzersiz olsun) gibi IMS çağrılarını kullanarak bir eşitleme noktası oluşturun.

Önceki denetim noktasından bu yana yapılan tüm değişiklikleri geri almak için IMS ROLB (rollback) çağrıyı kullanabilirsiniz. Daha fazla bilgi için IMS belgelerine bakın.

Kuyruk yöneticisi (IMS syncpoint yöneticisiyle birlikte), iş biriminde kurtarılabilir başka kaynaklar da varsa, iki aşamalı bir kesinleştirme protokolüne katılır.

Tüm açık tutamaçlar, IMS bağdaştırıcısı tarafından bir eşitleme noktasında (toplu ya da ileti odaklı olmayan BMP ortamı dışında) kapatılır. Bunun nedeni, MQPUT ya da MQGET çağrıları yapıldığında değil, MQCONN, MQCONNX ve MQOPEN çağrıları yapıldığında farklı bir kullanıcı sonraki iş birimini başlatabilir ve IBM MQ güvenlik denetimi gerçekleştirilir.

Ancak, bir WFI (Wait-for-Input) ya da pseudo Wait-for-Input (PWFI) ortamında, bir sonraki ileti gelinceye ya da uygulamaya bir QC durum kodu dönünceye kadar tanıtıcıları kapatmaları IMS bildirilmez IBM MQ . Uygulama IMS bölgesinde bekliyorsa ve bu tanıtıcılardan herhangi biri tetiklenen kuyruklara aitse, kuyruklar açık olduğu için tetikleme gerçekleşmez. Bu nedenle, WFI ya da PWFI ortamında çalışan uygulamalar, sonraki ileti için GU 'yu IOPCB' ye yapmadan önce kuyruk tanıtıcılarını belirtik olarak MQCLOSE olarak belirtmelidir.

Bir IMS uygulaması (BMP ya da MPP) MQDISC çağrısını yaparsa, açık kuyruklar kapanır, ancak örtük eşitleme noktası alınmaz. Uygulama olağan bir şekilde kapanırsa, açık kuyruklar kapanır ve örtük kesinleştirme gerçekleşir. Uygulama olağandışı bir şekilde kapanırsa, açık kuyruklar kapanır ve örtük bir geriletme oluşur.

z/OS *z/OS toplu iş uygulamalarında eşitleme noktaları*

Toplu iş uygulamaları için IBM MQ syncpoint yönetim çağrılarını kullanabilirsiniz: MQCMIT ve MQBACK. Önceki sürümlerle uyumluluk için, CSQBCMT ve CSQBBAK eşanlamlı olarak kullanılabilir.

Not: IBM MQ ve Db2gibi farklı kaynak yöneticileri tarafından yönetilen kaynaklara ilişkin güncellemeleri tek bir iş birimi içinde kesinleştirmeniz ya da geri çıkarmanız gerekiyorsa, RRS ' yi kullanabilirsiniz. Daha fazla bilgi için bkz. “Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri” sayfa 822.

MQCMIT çağrısı kullanılarak değişiklikler kesinleştiriliyor

Giriş olarak, MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını (*Hconn*) sağlamanız gerekir.

MQCMIT çıkışı bir tamamlanma kodu ve neden kodudur. Eşitleme noktası tamamlandıysa, ancak kuyruk yöneticisi önceki eşitleme noktasından bu yana koyma ve alma işlemlerini geri çektiyse, çağrı bir uyarıyla tamamlanır.

MQCMIT çağrısının başarıyla tamamlanması, kuyruk yöneticisine uygulamanın bir eşitleme noktasına ulaştığını ve önceki eşitleme noktasından bu yana yapılan tüm koyma ve alma işlemlerinin kalıcı hale getirildiğini gösterir.

Tüm başarısızlık yanıtları MQCMIT ' in tamamlanmadığı anlamına gelmez. Örneğin, uygulama MQRC_CONNECTION_BROKEN değerini alabilir.

MQCMITiçinde MQCMIT çağrısının bir açıklaması vardır.

MQBACK çağrısı kullanılarak değişiklikler geri alınıyor

Giriş olarak bir bağlantı tanıtıcısı sağlamanız gerekir (*Hconn*). MQCONN ya da MQCONNX çağrısıyla döndürülen tanıtıcıyı kullanın.

MQBACK çıkışı bir tamamlanma kodu ve neden kodudur.

Çıkış, kuyruk yöneticisine uygulamanın bir eşitleme noktasına ulaştığını ve son eşitleme noktasından bu yana yapılan tüm alma ve alma işleminin geriletildiğini gösterir.

MQBACKiçinde MQBACK çağrısının bir açıklaması var.

Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri

Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri (RRS), katılan kaynak yöneticileri arasında iki aşamalı eşitleme noktası desteği sağlayan bir z/OS olanağıdır.

Bir uygulama, IBM MQ ve Db2gibi çeşitli z/OS kaynak yöneticileri tarafından yönetilen kurtarılabilir kaynakları güncelleyebilir ve daha sonra, bu güncellemeleri tek bir iş birimi olarak kesinleştirebilir ya da geri alabilir. RRS, normal yürütme sırasında gerekli iş birimi durum günlüğünü sağlar, eşitleme noktası işlemesini koordine eder ve altsistemin yeniden başlatılması sırasında uygun iş birimi durumu bilgilerini sağlar.

IBM MQ for z/OS RRS katılımcı desteği, toplu iş, TSO ve Db2 saklanmış yordam ortamlarındaki IBM MQ uygulamalarının hem IBM MQ hem deIBM MQ dışı kaynakları güncellemesini sağlar (örneğin, Db2) tek bir mantıksal iş birimi içinde. RRS katılımcı desteği hakkında bilgi için bkz. *z/OS MVS Programming: Resource Recovery*.

IBM MQ uygulamanız MQCMIT ve MQBACK ya da eşdeğer RRS çağrılarını, SRRCMIT ve SRRBACK kullanabilir. Ek bilgi için bkz. “RRS toplu iş bağdaştırıcısı” sayfa 854 .

RRS kullanılabilirliği

z/OS sisteminizde RRS etkin değilse, RRS sınırlı kod öbeğine (CSQBRSTB ya da CSQBRRSI) bağlı bir programdan verilen herhangi bir IBM MQ çağrısı MQRC_ENVIRONMENT_ERROR değerini döndürür.

Db2 saklı yordamlar

Db2 saklanmış yordamlarını RRS ile birlikte kullanıyorsanız, aşağıdakilere dikkat edin:

- RRS kullanan Db2 saklanmış yordamları, iş yükü yöneticisi (WLM yönetimli) tarafından yönetilmelidir.
- Db2tarafından yönetilen bir saklanmış yordam IBM MQ çağrıları içeriyorsa ve RRS sınırlı kod öbeğine (CSQBRSTB ya da CSQBRRSI) bağlıysa, MQCONN ya da MQCONNX çağrısı MQRC_ENVIRONMENT_ERROR değerini döndürür.
- WLM tarafından yönetilen bir saklanmış yordam IBM MQ çağrıları içeriyorsa ve RRS olmayan bir sınırlı kod öbeğiyle bağlıysa, MQCONN ya da MQCONNX çağrısı, saklanmış yordam adresi alanı başlatıldığından bu yana yürütülen ilk IBM MQ çağrısı değilse, MQRC_ENVIRONMENT_ERROR değerini döndürür.
- Db2 saklanmış yordamınız IBM MQ çağrıları içeriyorsa ve RRS olmayan bir sınırlı kod öbeğiyle bağlıysa, saklanmış yordam adres alanı sona erinceye ya da sonraki bir saklanmış yordam bir MQCMIT (IBM MQ Batch/TSO sınırlı kod öbeği kullanılarak) gerçekleştirinceye kadar, o saklanmış yordamda güncellenen IBM MQ kaynakları kesinleştirilmez.
- Aynı saklanmış yordamın birden çok kopyası aynı adres alanında eşzamanlı olarak yürütülebilir. Db2 ' in saklanmış yordamınızın tek bir kopyasını kullanmasını istiyorsanız, programınızın yeniden girişli olarak kodlandığından emin olun. Tersi durumda, programınızdaki IBM MQ çağrılarında MQRC_HCONN_ERROR alabilirsiniz.
- WLM tarafından yönetilen Db2 saklanmış yordamında MQCMIT ya da MQBACK kodlamayın.
- Dil Ortamında (LE) çalıştırılacak tüm programları tasarlayın.

IBM i *IBM i uygulamaları için CICS içindeki eşitleme noktaları*

IBM MQ for IBM i , IBM i iş birimleri için CICS içine katılır. İletileri yürürlükteki iş birimine koymak ve almak için CICS for IBM i uygulaması içindeki MQI ' ı kullanabilirsiniz.

IBM MQ for IBM i işlemlerini içeren bir eşitleme noktası oluşturmak için EXEC CICS SYNCPOINT komutunu kullanabilirsiniz. Önceki eşitleme noktasına kadar olan tüm değişiklikleri geri yüklemek için EXEC CICS SYNCPOINT ROLLBACK komutunu kullanabilirsiniz.

Bir CICS for IBM i uygulamasında MQPMO_SYNCPOINT ya da MQGMO_SYNCPOINT ile bir MQPUT, MQPUT1ya da MQGET kullanıyorsanız, IBM MQ for IBM i , kaydını bir API kesinleştirme kaynağı olarak kaldırıncaya kadar CICS for IBM i oturumunu kapatamazsınız. Kuyruk yöneticisiyle bağlantınızı kesmeden önce, bekleyen koyma ya da alma işlemlerini kesinleştirin ya da geri çekin. Bu, IBM iiçin CICS oturumunu kapatmanızı sağlar.

Multi *IBM MQ for Multiplatforms içinde eşitleme noktaları*

Syncpoint desteği iki tip iş birimi üzerinde çalışır: yerel ve genel.

Yerel iş birimi, güncellenen kaynakların yalnızca IBM MQ kuyruk yöneticisinininkiler olduğu bir iş birimidir. Burada, kuyruk yöneticisinin kendisi tarafından tek aşamalı bir kesinleştirme yordamı kullanılarak eşitleme noktası eşgüdümü sağlanır.

Genel iş birimi, veritabanları gibi diğer kaynak yöneticilerine ait kaynakların da güncellendiği bir birimdir. IBM MQ , bu tür iş birimlerini koordine edebilir. Bunlar, bir dış kesinleştirme denetleyicisi tarafından da koordine edilebilir. Örneğin:

- Başka bir hareket yöneticisi
- IBM i IBM i kesinleştirme denetleyicisi

Tam bütünlük için iki aşamalı bir kesinleştirme yordamı kullanın. İki aşamalı kesinleştirme, XA uyumlu hareket yöneticileri ve veritabanları tarafından sağlanabilir. Örneğin:

- TXSeries
- UDB
- IBM i IBM i kesinleştirme denetleyicisi

ALW IBM MQ ürünleri, iki aşamalı kesinleştirme işlemini kullanarak genel iş birimlerini koordine edebilir.

IBM i IBM MQ for IBM i , bir WebSphere Application Server ortamındaki genel iş birimleri için kaynak yöneticisi olarak işlev görür, ancak hareket yöneticisi olarak işlev görmez.

Örtük eşitleme noktası

Kalıcı iletiler konurken IBM MQ , sürekli iletileri eşitleme noktası altına koymak için eniyilenir. Aynı kuyruğa kalıcı ileti yerleştirecek birden çok uygulama, bu uygulamalar eşitleme noktası kullanıyorsa daha iyi performans sağlar. Bunun nedeni, kalıcı iletileri koymak için syncpoint kullanılıyorsa, kuyruk için daha az çekişme olmasıdır.

ImplSyncOpenOutput , uygulamalar eşitleme noktasının dışına kalıcı iletiler koyduğunda örtük bir eşitleme noktası ekler. Bu, uygulamalar örtülü eşitleme noktasının farkında olmadan bir performans artışı sağlar.

Örtük uyumlulaştırma noktası, kuyruk için çekişmeyi azalttığı için, yalnızca kuyruğa birden çok uygulama konduğunda başarım artışı sağlar. Bu nedenle **ImplSyncOpenOutput** , örtük bir eşitleme noktası eklenmeden önce çıkış için bir kuyruğu açık olan uygulama sayısı alt sınırını belirtir. Varsayılan değer 2 'dir. Bu, **ImplSyncOpenOutput**belirtmezseniz, örtük eşitleme noktasının yalnızca birden çok uygulama kuyruğa yerleştiriliyorsa ekleneceği anlamına gelir.

Ek bilgi için Ayarlama Parametreleri konusuna bakın.

Çoklu Platformlar üzerinde yerel iş birimleri
Yalnızca kuyruk yöneticisini içeren iş birimlerine *yerel* iş birimi adı verilir. Uyumlulaştırma noktası eşgüdümü, kuyruk yöneticisinin kendisi (iç eşgüdüm) tarafından tek fazlı bir kesinleştirme işlemi kullanılarak sağlanır.

Yerel bir iş birimini başlatmak için, uygulama uygun uyumlulaştırma noktası seçeneğini belirterek MQGET, MQPUT ya da MQPUT1 isteklerini yayınlar. İş birimi MQCMIT kullanılarak kesinleştirildi ya da MQBACK kullanılarak geriye işlendi. Ancak, uygulama ile kuyruk yöneticisi arasındaki bağlantı bilerek ya da istemeyerek kesildiğinde de iş birimi sona erer.

IBM MQ tarafından eşgüdümlü bir genel iş birimi hala etkinken bir uygulama kuyruk yöneticisiyle bağlantısını keserse (MQDISC), iş birimini kesinleştirme girişiminde bulunur. Ancak, uygulama bağlantısı kesilmeden sona ererse, uygulamanın olağandışı bittiği varsayıldığı için iş birimi geriye işlenir.

Çoklu Platformlar üzerinde genel iş birimleri
Diğer kaynak yöneticilerine ait kaynaklara ilişkin güncellemeleri de eklemeniz gerekiyorsa, genel iş birimlerini kullanın.

Burada eşgüdüm, kuyruk yöneticisinin içinde ya da dışında olabilir:

İç eşitleme noktası koordinasyonu

Genel iş birimlerinin kuyruk yöneticisi eşgüdümü IBM MQ for IBM i ya da IBM MQ for z/OStarafından desteklenmez. Bir IBM MQ MQI client ortamdadesteklenmez.

Burada, IBM MQ koordinasyonu yapar. Genel bir iş birimini başlatmak için uygulama MQBEGIN çağrısına neden olur.

MQBEGIN çağrısına giriş olarak, MQCONN ya da MQCONNX çağrısıyla döndürülen bağlantı tanıtıcısını (*Hconn*) sağlamanız gerekir. Bu tanıtıcı, IBM MQ kuyruk yöneticisine yönelik bağlantıyı gösterir.

Uygulama, uygun eşitleme noktası seçeneğini belirten MQGET, MQPUT ya da MQPUT1 istekleri oluşturur. Bu, yerel kaynakları, diğer kaynak yöneticilerine ait kaynakları ya da her ikisini güncelleyen bir genel iş birimi başlatmak için MQBEGIN ' i kullanabileceğiniz anlamına gelir. Diğer kaynak yöneticilerine ait kaynaklarda yapılan güncellemeler, o kaynak yöneticisinin API 'si kullanılarak yapılır. Ancak, diğer kuyruk yöneticilerine ait kuyrukları güncellemek için MQI ' ı kullanamazsınız. Ek iş birimlerine (yerel ya da genel) başlamadan önce MQCMIT ya da MQBACK komutunu verin.

Genel iş birimi MQCMIT kullanılarak kesinleştirilir; bu, iş birimine dahil olan tüm kaynak yöneticilerinin iki aşamalı kesinleştirmeyi başlatır. Kaynak yöneticilerinin (örneğin, Db2, Oracleve Sybasegibi XA uyumlu veritabanı yöneticileri) kesinleştirmeye hazırlanmaları istendiği iki aşamalı bir kesinleştirme işlemi kullanılır. Ancak hepsi hazırsa, kesinleştirmeleri isteniyor. Herhangi bir kaynak yöneticisi kesinleştirilemeyeceğini belirtiyorsa, bunun yerine her birinden geri çekilmesi isteniyor. Diğer bir seçenek olarak, tüm kaynak yöneticilerinin güncellemelerini geri almak için MQBACK ' i kullanabilirsiniz.

Genel iş birimi etkinken bir uygulama bağlantıyı keserse (MQDISC), iş birimi kesinleştirilir. Ancak, uygulama bağlantısı kesilmeden sona ererse, uygulamanın olağandışı bittiği varsayıldığı için iş birimi geriye işlenir.

MQBEGIN çıkışı bir tamamlanma kodu ve neden kodudur.

Genel bir iş birimini başlatmak için MQBEGIN kullandığınızda, kuyruk yöneticisiyle yapılandırılmış tüm dış kaynak yöneticileri içerilir. Ancak, arama bir iş birimini başlatır, ancak aşağıdaki bir uyarıyla tamamlanır:

- Katılan kaynak yöneticisi yok (kuyruk yöneticisiyle kaynak yöneticisi yapılandırılmadı)

veya

- Bir ya da daha fazla kaynak yöneticisi kullanılamıyor.

Bu durumlarda, iş birimi, yalnızca iş birimi başlatıldığında kullanılabilir olan kaynak yöneticilerine ilişkin güncellemeleri içermelidir.

Kaynak yöneticilerinden biri güncellemelerini kesinleştiremezse, tüm kaynak yöneticilerine güncellemelerini geri alma talimatı verilir ve MQCMIT bir uyarıyla tamamlanır. Olağan dışı durumlarda (genellikle işletmen müdahalesi), bazı kaynak yöneticileri güncellemelerini kesinleştirirken diğerleri geri alırsa bir MQCMIT çağrısı başarısız olabilir; işin *karışık* bir sonuçla tamamlandığı varsayılır. Bu tür oluşumlar, düzelme işleminin yapılabilmesi için kuyruk yöneticisinin hata günlüğünde tanılanabilir.

Bir genel iş biriminin MQCMIT 'si, ilgili tüm kaynak yöneticileri güncellemelerini kesinleştirirse başarılı olur.

MQBEGIN çağrısının açıklaması için bkz. MQBEGIN.

Dış eşitleme noktası koordinasyonu

Bu, IBM MQ dışında bir eşitleme noktası eşgüdümcüsü seçildiğinde oluşur; örneğin, CICS, Encina ya da Tuxedo.

Bu durumda IBM MQ for AIX, Linux, and Windows sistemleri, kesinleştirilmemiş alma ya da koyma işlemlerini gerektiği şekilde kesinleştirebilmeleri ya da geriye işleyebilmeleri için syncpoint eşgüdümcüsüyle iş biriminin sonucuna olan ilgilerini kaydeder. Dış eşitleme noktası koordinatörü, bir ya da iki aşamalı kesinleştirme protokollerinin sağlanıp sağlanmadığını belirler.

Dış eşgüdümcü kullandığınızda, MQCMIT, MQBACK ve MQBEGIN komutu verilemez. Bu işlevlere yapılan çağrılar, MQRC_ENVIRONMENT_ERROR neden koduyla başarısız olur.

Dışarıdan eşgüdümlü bir iş biriminin nasıl başlatılacağı, syncpoint koordinatörü tarafından sağlanan programlama arabirimine bağlıdır. Belirtik bir çağrı gerekebilir. Belirtik bir çağrı gerekiyorsa ve bir iş birimi başlatılmadığında MQPMO_SYNCPOINT seçeneğini belirterek bir MQPUT çağrısı yaparsanız, MQRC_SYNCPOINT_NOT_ALLOWED tamamlanma kodu döndürülür.

İş biriminin kapsamı, syncpoint koordinatörü tarafından belirlenir. Uygulama ile kuyruk yöneticisi arasındaki bağlantının durumu, iş biriminin durumunu değil, bir uygulamanın sorun çıkardığı MQI çağrılarının başarılı ya da başarısız olduğunu etkiler. Bir uygulama, örneğin, etkin bir iş birimi sırasında

bir kuyruk yöneticisiyle bağlantı kesebilir ve yeniden bağlantı kurabilir ve aynı iş birimi içinde ek MQGET ve MQPUT işlemleri gerçekleştirebilir. Bu, beklemedeki bağlantı kesilmesi olarak bilinir.

CICSXA yeteneklerini kullanmayı seçseniz de, CICS programlarında IBM MQ API çağrılarını kullanabilirsiniz. XA kullanmıyorsanız, kuyruklara/kuyruklardan gelen iletilerin girişleri ve alınmaları CICS atomik iş birimleri içinde yönetilmez. Bu yöntemi seçmenin bir nedeni, iş biriminin genel tutarlılığının sizin için önemli olmamasıdır.

İş birimlerinizin bütünlüğü sizin için önemliyse, XA kullanmalısınız. XA kullandığınızda, CICS iş birimindeki tüm kaynakların birlikte güncellenmesini sağlamak için iki aşamalı bir kesinleştirme protokolü kullanır.

Hareket desteği ayarlama hakkında daha fazla bilgi için bkz. İşlemsel destek senaryolarıve TXSeries CICS belgeleri; örneğin, *TXSeries for Multiplatforms CICS Administration Guide for Open Systems*.

Multi *Çoklu platformlarda örtük eşitleme noktası*
Örtük eşitleme noktası desteği, eşitleme noktasının dışına kalıcı ileti konmasına olanak sağlar.

Kalıcı iletiler konurken IBM MQ , sürekli iletileri eşitleme noktası altına koymak için eniyilenir. Aynı kuyruğa eşzamanlı olarak kalıcı iletiler kosan birden çok uygulama genellikle bu uygulamalar eşitleme noktası kullanıyorsa daha iyi performans gösterir. Bunun nedeni, kalıcı iletiler konurken eşitleme noktası kullanılırsa IBM MQ kilitleme stratejisinin daha verimli olmasıdır.

`qm.ini` dosyasındaki **ImplSyncOpenOutput** parametresi, uygulamalar kalıcı iletileri eşitleme noktasının dışına koyduğunda örtük bir eşitleme noktasının eklenip eklenemeyeceğini denetler. Bu, uygulamalar örtük eşitleme noktasının farkında olmadan bir performans artışı sağlayabilir.

Örtük uyumlulaştırma noktası, kilit çekişmesini azalttığı için, yalnızca kuyruğa koşutzamanlı olarak birden çok uygulama yerleştirildiğinde başarım artışı sağlar. **ImplSyncOpenOutput** , örtük bir eşitleme noktası eklenmeden önce çıkış için kuyruğu açık olan uygulama sayısı alt sınırını belirtir. Varsayılan değer *2*' dir. Bu, **ImplSyncOpenOutput**öğesini belirtik olarak belirtmezseniz, örtük eşitleme noktasının yalnızca birden çok uygulama kuyruğa yerleştirildiğinde ekleneceği anlamına gelir.

Örtük bir eşitleme noktası eklerseniz, istatistikler bu oluşumu yansıtır ve **runmqsc display conn**' dan bir hareket çıkışı görebilirsiniz.

Hiçbir zaman örtük bir eşitleme noktası eklenmesini istemiyorsanız **ImplSyncOpenOutput**=*OFF* değerini belirleyin.

Ek bilgi için Ayarlama Parametreleri konusuna bakın.

Multiplatforms üzerinde dış syncpoint yöneticilerine yönelik arabirimler
IBM MQ for Multiplatforms , X/Open XA arabirimini kullanan dış syncpoint yöneticileri tarafından işlemlerin koordinasyonunu destekler.

Bazı XA hareket yöneticileri (TXSeries), her XA kaynak yöneticisinin kendi adını sağlamasını gerektirir. Bu, XA anahtar yapısında `name` olarak adlandırılan dizedir.

- ALW AIX, Linux, and Windows üzerinde IBM MQ kaynak yöneticisi MQSeries_XA_RMI adını taşır.
- IBM i IBM iiçin kaynak yöneticisi adı MQSeries XA RMI 'dir.

XA arabirimleriyle ilgili daha fazla ayrıntı için, The Open Group tarafından yayınlanan *CAE Specification Distributed Transaction Processing: The XA Specification*adlı XA belgelerine bakın.

Bir XA yapılandırmasında IBM MQ for Multiplatforms , bir XA kaynak yöneticisi rolünü yerine getirir. Bir XA uyumlulaştırma noktası eşgüdümcüsü, bir XA kaynak yöneticisi kümesini yönetebilir ve her iki kaynak yöneticisinde de hareketlerin kesinleştirilmesini ya da gerilemesini uyumlulaştırabilir. Statik olarak kayıtlı bir kaynak yöneticisi için çalışma şekli şöyledir:

1. Bir uygulama, syncpoint koordinatörüne bir hareket başlatmak istediğini bildirir.

2. Syncpoint eşgüdümcüsü, bildiği tüm kaynak yöneticilerine, yürürlükteki hareketi bildirmeleri için bir çağrı yayınlar.
3. Uygulama, yürürlükteki işlemle ilişkilendirilmiş kaynak yöneticileri tarafından yönetilen kaynakları güncellemek için çağrılar yayınlar.
4. Uygulama, syncpoint eşgüdümleyicisinden hareketi kesinleştirmesini ya da geriye işlemesini ister.
5. Syncpoint eşgüdümcüsü, hareketi istenen şekilde tamamlamak için iki aşamalı kesinleştirme protokollerini kullanarak her kaynak yöneticisine çağrı verir.

XA belirtimi, her kaynak yöneticisinin XA Anahtarı adlı bir yapı sağlamasını gerektirir. Bu yapı, kaynak yöneticisinin yeteneklerini ve syncpoint eşgüdümcüsü tarafından çağrılacak işlevleri bildirir.

Bu yapının iki sürümü vardır:

Çizelge 128. XA Anahtarı Sürümleri

Sürüm	**Açıklama**
MQRMIXASwitch	Statik XA kaynak yönetimi
MQRMIXASwitchDynamic	Dinamik XA kaynak yönetimi

Bu yapıyı içeren kitaplıkların listesi için bkz. IBM MQ XA anahtar yapısı.

Bunları bir XA syncpoint koordinatörüne bağlamak için kullanılması gereken yöntem, eşgüdümcü tarafından tanımlanır; IBM MQ ' in XA syncpoint koordinatörünüzle nasıl işbirliği yapacağını belirlemek için ilgili eşgüdümcü tarafından sağlanan belgelere bakın.

Syncpoint eşgüdümcüsü tarafından herhangi bir *xa_open* çağrısına geçirilen *xa_info* yapısı, denetlenecek kuyruk yöneticisinin adı olabilir. Bu, MQCONN ya da MQCONNX 'e geçirilen kuyruk yöneticisi adıyla aynı formu alır ve varsayılan kuyruk yöneticisi kullanılacaksa boş olabilir. Ancak, fazladan iki TPM ve AXLIB parametresini kullanabilirsiniz.

TPM, IBM MQ işlem yöneticisi adını belirtmenizi sağlar; örneğin, CICS. AXLIB, hareket yöneticisinde XA AX giriş noktalarının bulunduğu gerçek kitaplık adını belirtmenizi sağlar.

Bu parametrelerden birini ya da varsayılan olmayan bir kuyruk yöneticisini kullanırsanız, QMNAME parametresini kullanarak kuyruk yöneticisi adını belirtmeniz gerekir. Daha fazla bilgi için bkz. xa_open dizgisinin CHANNEL, TRPTYPE, CONNAME ve QMNAME parametreleri.

Sınırlamalar

1. Paylaşılan bir Hconn ile genel iş birimlerine izin verilmez (açıklamalar için bkz. “MQCONNX ile paylaşılan (iş parçacığından bağımsız) bağlantılar” sayfa 709.
2. IBM i IBM MQ for IBM i , XA kaynak yöneticilerinin dinamik kaydını desteklemez.

   Desteklenen tek hareket yöneticisi WebSphere Application Server.
3. Windows Windows sistemlerinde, XA anahtarında bildirilen tüm işlevler _cdecl işlevleri olarak bildirilir.
4. Bir dış uyumlulaştırma noktası eşgüdümcüsü, bir kerede tek bir kuyruk yöneticisini denetleyebilir. Bunun nedeni, eşgüdümcünün her kuyruk yöneticisiyle etkin bir bağlantısının olması ve bu nedenle aynı anda yalnızca bir bağlantıya izin verilmesi kuralına tabi olmasıdır.

   Not: Not: JEE sunucusunda çalışan bir JMS istemci uygulaması (CLIENT JEE uygulaması) bu kısıtlamaya sahip değildir, bu nedenle tek bir JEE sunucusu tarafından yönetilen hareket, aynı işlemde birden çok kuyruk yöneticisini koordine edebilir. Ancak, bağ tanımlama kipinde çalışan bir JMS sunucu uygulaması, aynı anda yalnızca bir bağlantıya izin verilen kurala tabi olmaya devam etmektedir.
5. Syncpoint eşgüdümcüsü kullanılarak çalıştırılan tüm uygulamalar, eşgüdümcü tarafından denetlenen kuyruk yöneticisine bağlanabilir; bu kuyruk yöneticisine zaten etkin bir şekilde bağlanmış durumdadırlar. Bağlantı tanıtıcısı elde etmek için MQCONN ya da MQCONNX yayınlamalı ve

çıkmadan önce MQDISC vermelidir. Diğer bir yöntem olarak, TXSeries CICSiçin UE014015 çıkışını da kullanabilirler.

IBM i *IBM i dış eşitleme noktası yöneticisine ilişkin arabirimler*

IBM MQ for IBM i , yerel IBM i kesinleştirme denetimini dış uyumlulaştırma noktası eşgüdümcüsü olarak kullanabilir.

Kesinleştirme denetimiyle iş parçacığından bağımsız (paylaşılan) bağlantılara izin verilmez. IBM i' in kesinleştirme denetimi yeteneklerine ilişkin ek bilgi için *IBM i Programming: Backup and Recovery Guide, SC21-8079* belgesine bakın.

IBM i kesinleştirme denetimi olanaklarını başlatmak için STRCMTCTL sistem komutunu kullanın. Kesinleştirme denetimini sona erdirmek için ENDCMTCTL sistem komutunu kullanın.

Not: *Kesinleştirme tanımı kapsamı* varsayılan değeri *ACTGRP ' dir. Bu, IBM iiçin IBM MQ için *JOB olarak tanımlanmalıdır. Örneğin:

```
STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
```

IBM MQ for IBM i , yalnızca IBM MQ kaynaklarına ilişkin güncellemeleri içeren yerel iş birimlerini de gerçekleştirebilir. Uygulama, MQPMO_SYNCPOINT ya da MQGMO_SYNCPOINT ya da MQGMO_SYNCPOINT ya da MQBEGIN belirtilerek MQPUT, MQPUT1ya da MQGET öğesini çağırdığında, IBM i ile eşgüdümlenen genel iş birimlerine yerel iş birimleri ile katılım arasındaki seçim her uygulamada yapılır. Bu tür ilk çağrı yapıldığında kesinleştirme denetimi etkin değilse, IBM MQ yerel bir iş birimini başlatır ve bu bağlantıya ilişkin diğer tüm iş birimleri IBM MQ , kesinleştirme denetiminin başlatılıp başlatılmadığından bağımsız olarak yerel iş birimlerini de kullanır. Yerel bir iş birimini kesinleştirmek için MQCMIT kullanın. Yerel bir iş birimini geri almak için MQBACK kullanın. COMMIT Denetim dili (CL) komutu gibi IBM i kesinleştirme ve geriye işleme çağrılarının IBM MQ yerel iş birimleri üzerinde bir etkisi yoktur.

Dış eşitleme noktası koordinatörü olarak IBM MQ for IBM i yerel IBM i kesinleştirme denetimiyle kullanmak istiyorsanız, kesinleştirme denetimine sahip her işin etkin olduğundan ve IBM MQ ' yi tek iş parçacıklı bir işte kullandığınızdan emin olun. Kesinleştirme denetiminin başlatıldığı çok iş parçacıklı bir işte MQPMO_SYNCPOINT ya da MQGMO_SYNCPOINT belirterek MQPUT, MQPUT1ya da MQGET çağrılırsa, çağrı MQRC_SYNCPOINT_NOT_ALLOWED neden koduyla başarısız olur.

Çok iş parçacıklı bir işte yerel iş birimleri ve MQCMIT ve MQBACK çağrıları kullanılabilir.

Kesinleştirme denetimini başlattıktan sonra MQPMO_SYNCPOINT ya da MQGMO_SYNCPOINT belirterek MQPUT, MQPUT1ya da MQGET öğesini çağırırsanız, IBM MQ for IBM i kendisini kesinleştirme tanımlamasına API kesinleştirme kaynağı olarak ekler. Bu genellikle bir işte bu tür ilk çağrıdır. Belirli bir kesinleştirme tanımı altında kayıtlı API kesinleştirme kaynakları olsa da, bu tanım için kesinleştirme denetimini sona erdiremezsiniz.

IBM MQ for IBM i , yürürlükteki iş biriminde bekleyen MQI işlemleri yoksa, kuyruk yöneticisiyle bağlantınızı kestiğinizde API kesinleştirme kaynağı olarak kaydını kaldırır.

Yürürlükteki iş biriminde bekleyen MQPUT, MQPUT1ya da MQGET işlemleri varken kuyruk yöneticisiyle bağlantınızı keserseniz, IBM MQ for IBM i sonraki kesinleştirme ya da geriye işleme işleminin bildirilmesi için bir API kesinleştirme kaynağı olarak kayıtlı kalır. Sonraki eşitleme noktasına ulaşıldığında, IBM MQ for IBM i değişiklikleri gerektiği gibi kesinleştirir ya da geri alır. Bir uygulama, etkin bir iş birimi sırasında kuyruk yöneticisinin bağlantısını kesebilir ve kuyruk yöneticisine yeniden bağlanabilir ve aynı iş birimi içinde başka MQGET ve MQPUT işlemleri gerçekleştirebilir (bu, bekleyen bir bağlantıdır).

Bu kesinleştirme tanımı için bir ENDCMTCTL sistem komutu verme girişiminde bulunursanız, bekleyen değişikliklerin etkin olduğunu belirten CPF8355 iletisi görüntülenir. Bu ileti, iş sona erdiğinde iş günlüğünde de görüntülenir. Bunu önlemek için, bekleyen tüm IBM MQ for IBM i işlemlerini kesinleştirin ya da geri döndürün ve kuyruk yöneticisiyle bağlantıyı kesin. Bu nedenle, ENDCMTCTL ' den önce COMMIT ya da ROLLBACK komutlarının kullanılması, kesinleştirme sonu denetiminin başarıyla tamamlanmasını sağlar.

IBM i kesinleştirme denetimini dış uyumlulaştırma noktası eşgüdümcüsü olarak kullandığınızda, MQCMIT, MQBACK ve MQBEGIN çağrılarını yayınlayamazsınız. Bu işlevlere yapılan çağrılar, MQRC_ENVIRONMENT_ERROR neden koduyla başarısız olur.

İş biriminizi kesinleştirmek ya da geri yüklemek (yani, geri almak) için, kesinleştirme denetimini destekleyen programlama dillerinden birini kullanın. Örneğin:

- CL komutları: COMMIT ve ROLLBACK
- ILE C Programlama İşlevleri: _Rcommit ve _Rrollback
- ILE RPG: COMMIT ve ROLBK
- COBOL/400: COMMIT ve ROLLBACK

IBM i kesinleştirme denetimini IBM MQ for IBM iile dış uyumlulaştırma noktası eşgüdümcüsü olarak kullandığınızda, IBM i IBM MQ ' in katıldığı iki aşamalı bir kesinleştirme protokolü gerçekleştirir. Her iş birimi iki aşamada kesinleştirildiği için, kuyruk yöneticisi birinci aşamada kesinleştirme oylamasından sonra ikinci aşamada kullanılamaz duruma gelebilir. Örneğin, kuyruk yöneticisinin iç işleri sona erdirilirse bu durum oluşabilir. Bu durumda, kesinleştirmeyi gerçekleştiren iş günlüğü CPF835F iletisini içerir; bu ileti, bir kesinleştirme ya da geriye işleme işleminin başarısız olduğunu gösterir. Bundan önceki iletiler, sorunun nedenini, kesinleştirme ya da geriye işleme işlemi sırasında oluşup oluşmadığını ve başarısız olan iş birimine ilişkin mantıksal iş birimi tanıtıcısını (LUWID) gösterir.

Sorunun nedeni, hazırlanmış bir iş biriminin kesinleştirilmesi ya da geriye işlenmesi sırasında IBM MQ API kesinleştirme kaynağının başarısız olması ise, işlemi tamamlamak ve hareketin bütünlüğünü geri yüklemek için WRKMQMTRN komutunu kullanabilirsiniz. Komut, kesinleştirilecek ve geri çekilecek iş biriminin LUWID değerini bilmenizi gerektirir.

Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması

Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

Kuyruklara hizmet veren bazı IBM MQ uygulamaları sürekli olarak çalışır, bu nedenle kuyruklara gelen iletileri almak için her zaman kullanılabilir. Ancak, kuyruklara gelen iletilerin sayısı önceden kestirilemediğinde bunu istemeyebilirsiniz. Bu durumda, alınacak ileti olmasa bile, uygulamalar sistem kaynaklarını tüketiyor olabilir.

IBM MQ , bir uygulamanın alınabilecek iletiler olduğunda otomatik olarak başlatılmasını sağlayan bir olanak sağlar. Bu olanak *triggering*olarak bilinir.

Kanalları tetikleme hakkında bilgi için bkz. Kanalları tetikleme.

Tetikleyen nedir?

Kuyruk yöneticisi, belirli koşulları *tetikleyici olayları*olarak tanımlar.

Bir kuyruk için tetikleme etkinleştirilirse ve bir tetikleme olayı ortaya çıkarsa, kuyruk yöneticisi *başlatma kuyruğu*adı verilen bir kuyruğa *tetikleme iletisi* gönderir. Tetikleyici iletisinin başlatma kuyruğunda bulunması, bir tetikleyici olayının oluştuğunu gösterir.

Kuyruk yöneticisi tarafından oluşturulan tetikleyici iletileri kalıcı değil. Bu, günlüğe kaydetmeyi azaltır (başarımı artırır) ve yeniden başlatma sırasında yinelemeleri en aza indirir, böylece yeniden başlatma süresini kısaltır.

Başlatma kuyruğunu işleyen programa *tetikleyici izleme uygulaması*adı verilir ve bunun işlevi tetikleyici iletisini okumak ve tetikleyici iletide bulunan bilgilere dayalı olarak uygun işlemi gerçekleştirmektir. Genellikle bu işlem, tetikleyici iletisini oluşturan kuyruğu işlemek için başka bir uygulama başlatmaktır. Kuyruk yöneticisinin bakış açısından, tetikleyici izleme uygulamasıyla ilgili özel bir şey yoktur; bu, kuyruktaki (başlatma kuyruğu) iletileri okuyan başka bir uygulamadır.

Bir kuyruk için tetikleme etkinleştirildiyse, ilişkili bir *süreç tanımlaması nesnesi* yaratabilirsiniz. Bu nesne, tetikleme olayına neden olan iletiyi işleyen uygulamayla ilgili bilgileri içerir. Süreç tanımlaması nesnesi yaratılırsa, kuyruk yöneticisi bu bilgileri alır ve tetikleyici izleyici uygulaması tarafından kullanılmak üzere tetikleyici iletisine yerleştirir. Bir kuyrukla ilişkilendirilmiş süreç tanımlamasının adı, *ProcessName* yerel

kuyruk özniteliği tarafından verilir. Her kuyruk farklı bir süreç tanımlaması belirtebilir ya da birden çok kuyruk aynı süreç tanımlamasını paylaşabilir.

Bir kanalın başlangıcını tetiklemek istiyorsanız, bir süreç tanımlaması nesnesi tanımlamanız gerekmez. Bunun yerine iletim kuyruğu tanımı kullanılır.

Tetikleme, AIX, Linux, and Windowsüzerinde çalışan IBM MQ istemcileri tarafından desteklenir. İstemci ortamında çalışan bir uygulama, istemci kitaplıklarına bağlamanız dışında, tam IBM MQ ortamında çalışan bir uygulamayla aynıdır. Ancak, tetikleyici izleme programının ve başlatılacak uygulamanın aynı ortamda olması gerekir.

Tetikleme şunları içerir:

Uygulama kuyruğu
Uygulama kuyruğu , tetikleme açık olduğunda ve koşullar karşılandığında tetikleyici iletilerin yazılmasını gerektiren yerel bir kuyruktur.

Süreç tanımlaması
Bir uygulama kuyruğuyla ilişkilendirilmiş bir *süreç tanımlaması nesnesi* olabilir; bu nesne, uygulama kuyruğundan ileti alacak uygulamanın ayrıntılarını içerir. (Özniteliklerin listesi için Süreç tanımlamaları için öznitelikler konusuna bakın.)

Bir tetikleyicinin kanal başlatmasını istiyorsanız, süreç tanımlaması nesnesi tanımlamanıza gerek olmadığını unutmayın.

İletim kuyruğu
Bir tetikleyicinin kanal başlatmasını istiyorsanız bir iletim kuyruğuna gereksinim duyarsınız.

Linuxdışındaki bir platformda bulunan bir iletim kuyruğu için, iletim kuyruğunun `TriggerData` özniteliği başlatılacak kanalın adını belirleyebilir. Bu, tetikleyici kanallar için süreç tanımlamasının yerine geçebilir, ancak yalnızca bir süreç tanımlaması yaratılmadığında kullanılır.

Tetikleme olayı
Tetikleme olayı , kuyruk yöneticisi tarafından bir tetikleme iletisi oluşturulmasına neden olan bir olaydır. Bu genellikle bir uygulama kuyruğuna gelen bir iletidir, ancak başka zamanlarda da oluşabilir. Örneğin, bkz. “Tetikleme olayına ilişkin koşullar” sayfa 835.

IBM MQ , bir tetikleme olayına neden olan koşulları denetlemenizi sağlayacak bir seçenek aralığına sahiptir (bkz. “Tetikleyici olaylarının denetlenmesi” sayfa 839).

Tetikleyici ileti
Kuyruk yöneticisi, bir tetikleyici olayını tanıdığında bir *tetikleyici iletisi* oluşturur. Başlatılacak uygulamaya ilişkin tetikleyici ileti bilgilerine kopyalar. Bu bilgiler uygulama kuyruğundan ve uygulama kuyruğuyla ilişkilendirilmiş süreç tanımlaması nesnesinden gelir.

Tetikleyici iletilerin sabit bir biçimi vardır (bkz. “Tetikleyici iletilerinin biçimi” sayfa 846).

Başlatma kuyruğu
Başlatma kuyruğu , kuyruk yöneticisinin tetikleyici iletileri koyduğu yerel bir kuyruktur. Başlatma kuyruğunun bir diğer ad kuyruğu ya da model kuyruğu olamayacağını unutmayın.

Bir kuyruk yöneticisi birden çok başlatma kuyruğuna sahip olabilir ve her biri bir ya da daha çok uygulama kuyruğuyla ilişkilendirilir.

z/OS Bir kuyruk paylaşım grubundaki kuyruk yöneticileri tarafından erişilebilen paylaşılan bir kuyruk, IBM MQ for z/OSüzerindeki bir başlatma kuyruğu olabilir.

Tetikleyici
Tetikleyici , bir ya da daha çok başlatma kuyruğuna hizmet veren, sürekli çalışan bir programdır. Başlatma kuyruğuna bir tetikleyici iletisi geldiğinde, tetikleyici izleme programı iletiyi alır. Tetikleyici izleme programı, tetikleyici iletideki bilgileri kullanır. Uygulama kuyruğuna gelen iletileri almak ve uygulama kuyruğunun adını içeren tetikleyici ileti üstbilgisinde bulunan bilgileri iletmek üzere uygulamayı başlatmak için bir komut gönderir.

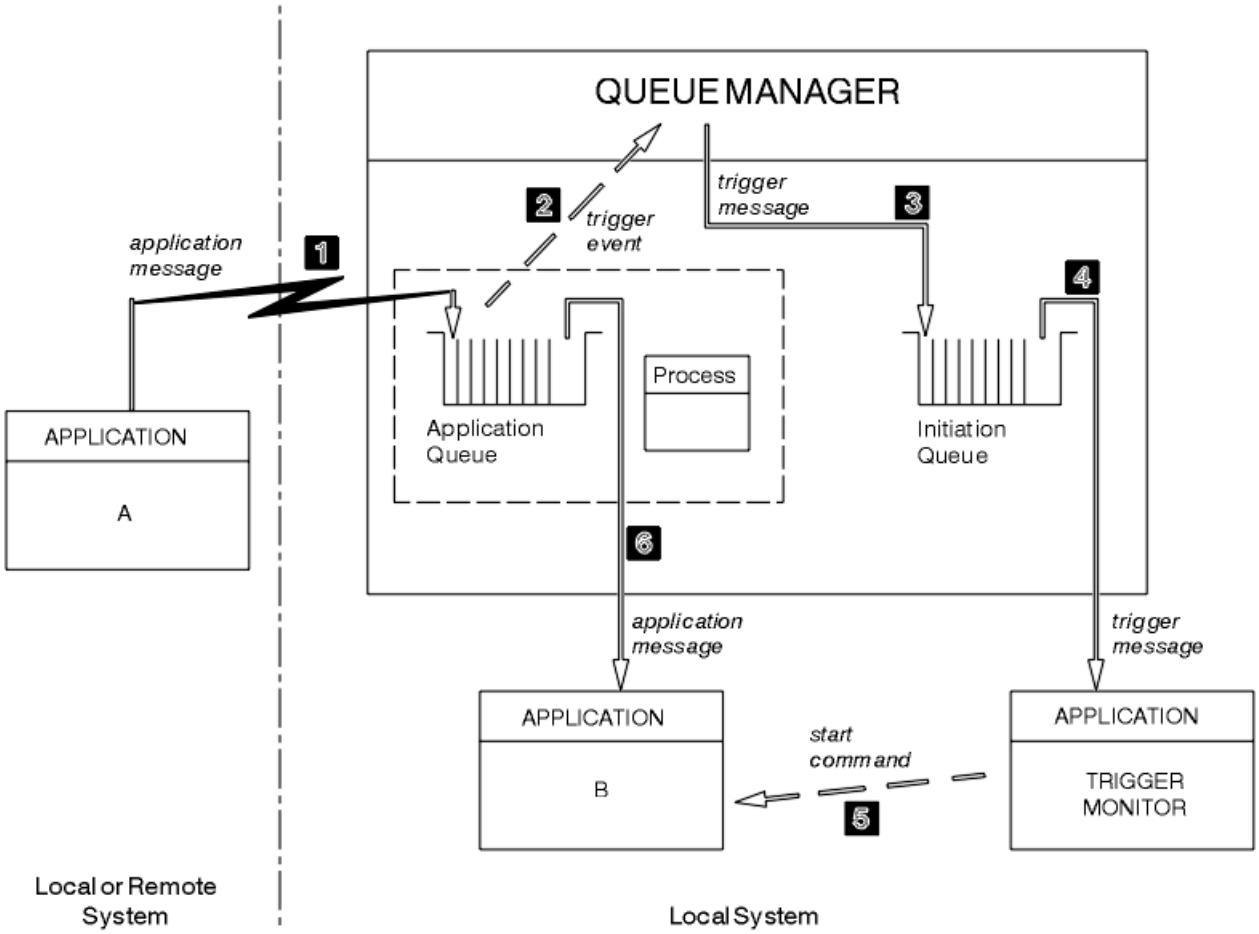
Tüm platformlarda, kanal başlatıcısı olarak bilinen özel bir tetikleyici, başlangıç kanallarından sorumludur.

z/OS z/OSsistemlerinde, kanal başlatıcı genellikle el ile başlatılır ya da kuyruk yöneticisi başlatma JCL 'sinde CSQINP2 değiştirilerek kuyruk yöneticisi başlatıldığında otomatik olarak başlatılabilir.

Multi Çoklu platformlarişletim sistemi üzerinde, kuyruk yöneticisi başlatıldığında kanal başlatıcı otomatik olarak başlatılır ya da **runmqchi** komutuyla el ile başlatılabilir.

Daha fazla bilgi için bkz "Tetikleyici izleme programları tarafından başlatma kuyruğunun işlenmesi" sayfa 842.

Tetikleyicinin nasıl çalıştığını anlamak için, FIRST (MQTT_FIRST) tetikleyici tipinin bir örneği olan Şekil 95 sayfa 831değerini göz önünde bulundurun.



Şekil 95. Uygulama akışı ve tetikleyici iletileri

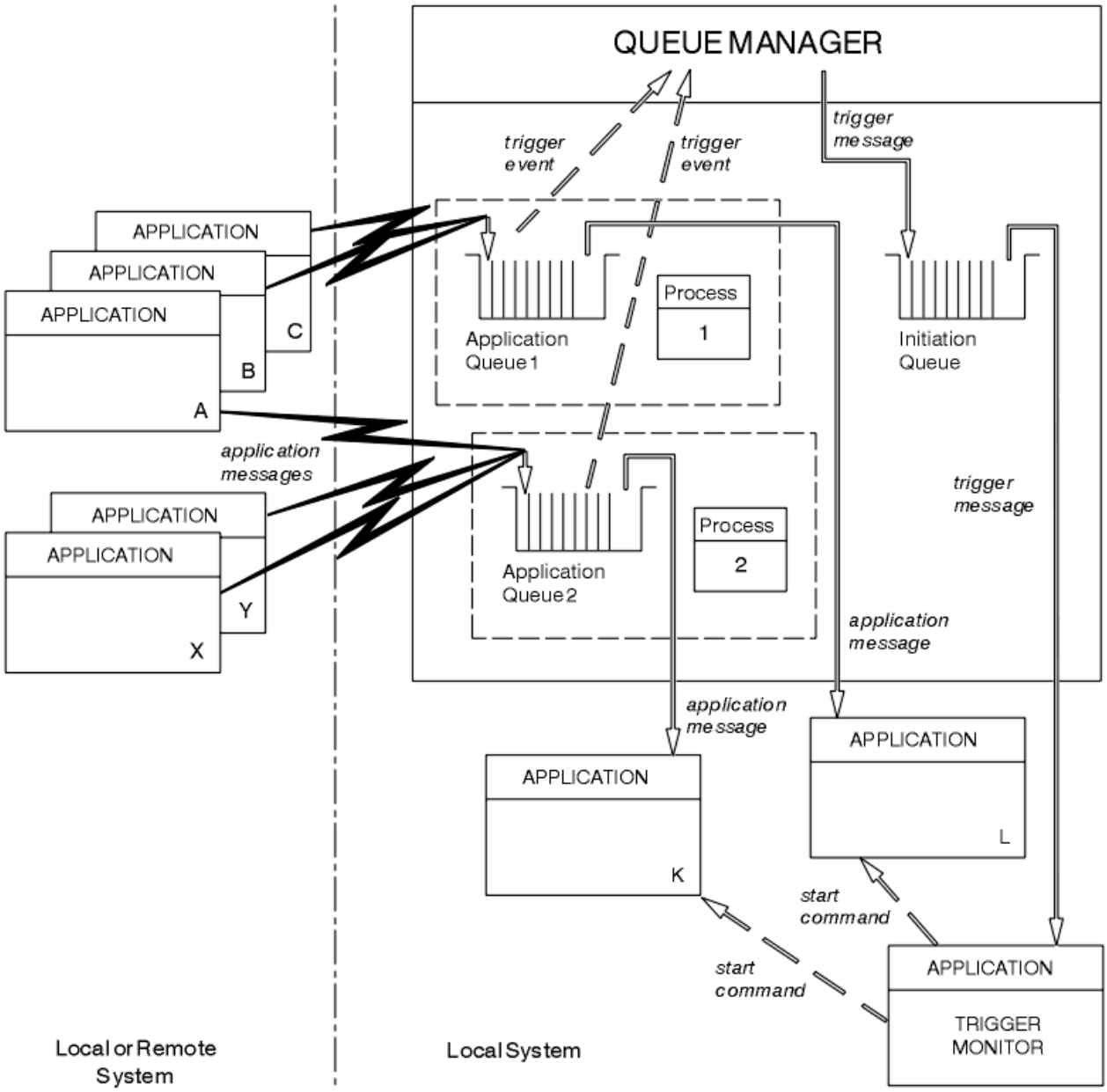
Şekil 95 sayfa 831içinde olayların sırası şöyledir:

1. Kuyruk yöneticisi için yerel ya da uzak olabilen Uygulama A, uygulama kuyruğuna bir ileti koyar. Giriş için bu kuyruk açık bir uygulama yok. Ancak, bu olgu yalnızca FIRST ve DEPTH tiplerini tetiklemek için geçerlidir.
2. Kuyruk yöneticisi, bir tetikleme olayı oluşturmak için gereken koşulların karşılanması durumunda olup olmadığını denetler. Bunlar ve bir tetikleyici olayı oluşturulur. Tetikleyici iletisi yaratılırken, ilişkili süreç tanımlaması nesnesi içinde tutulan bilgiler kullanılır.
3. Kuyruk yöneticisi bir tetikleyici iletisi yaratır ve bunu bu uygulama kuyruğuyla ilişkilendirilmiş başlatma kuyruğuna koyar; ancak, bir uygulamanın (tetikleyici izleyicisi) giriş için başlatma kuyruğu açıksa.
4. Tetikleyici izleme programı, tetikleyici iletisini başlatma kuyruğundan alır.
5. Tetikleyici, B uygulamasını (sunucu uygulaması) başlatmak için bir komut verir.
6. Uygulama B, uygulama kuyruğunu açar ve iletiyi alır.

Not:

1. Uygulama kuyruğu herhangi bir program tarafından giriş için açıksa ve FIRST ya da DEPTH için tetikleme ayarlıysa, kuyruk zaten sunulduğundan tetikleme olayı gerçekleşmez.
2. Başlatma kuyruğu giriş için açık değilse, kuyruk yöneticisi herhangi bir tetikleyici ileti oluşturmaz; bir uygulama giriş için başlatma kuyruğunu açıncaya kadar bekler.
3. Kanallar için tetikleme kullanırken, FIRST ya da DEPTH tetikleyici tipini kullanın.
4. Tetiklenen uygulamalar, tetikleyici izleme programını başlatan kullanıcının, CICS kullanıcısının ya da kuyruk yöneticisini başlatan kullanıcının kullanıcı kimliği ve grubu altında çalışır.

Şu ana kadar, tetikleme içindeki kuyruklar arasındaki ilişki sadece bire bir olarak gerçekleşmiştir. Şekil 96 sayfa 832başlıklı konuya dikkat edin.



Şekil 96. Tetikleme içindeki kuyrukların ilişkisi

Bir uygulama kuyruğuyla ilişkilendirilmiş, iletiyi işleyecek uygulamanın ayrıntılarını içeren bir süreç tanımlaması nesnesi var. Kuyruk yöneticisi, bilgileri tetikleyici iletisine yerleştirdiği için yalnızca bir

başlatma kuyruğu gereklidir. Tetikleyici, tetikleyici iletiden bu bilgileri alır ve her uygulama kuyruğundaki iletiyle ilgilenmek için ilgili uygulamayı başlatır.

Bir kanalın başlatılmasını tetiklemek istiyorsanız, bir süreç tanımlaması nesnesi tanımlamanıza gerek olmadığını unutmayın. İletim kuyruğu tanımı, tetiklenecek kanalı belirleyebilir.

Tetikleyicileri kullanarak IBM MQ uygulamalarını başlatmaya ilişkin ek bilgi edinmek için aşağıdaki bağlantıları kullanın:

- "Tetikleme için önkoşullar" sayfa 833
- "Tetikleme olayına ilişkin koşullar" sayfa 835
- "Tetikleyici olaylarının denetlenmesi" sayfa 839
- "Tetiklenen kuyrukları kullanan bir uygulama tasarlanması" sayfa 841
- "Tetikleyici izleme programları tarafından başlatma kuyruğunun işlenmesi" sayfa 842
- "Tetikleyici iletilerinin özellikleri" sayfa 845
- "Tetikleme çalışmadığında" sayfa 847

İlgili kavramlar

"İleti Kuyruğu Arabirimine Genel Bakış" sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

"Kuyruk yöneticisine bağlanma ve bağlantı kesme" sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

"Nesnelerin açılması ve kapatılması" sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

"Kuyruktaki iletilerin konması" sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

"Kuyruktan ileti alma" sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

"Nesne öznitelikleri hakkında sorma ve bunları ayarlama" sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

"İş birimlerinin kesinleştirilmesi ve geri çekilmesi" sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

"MQI ve kümelerle çalışma" sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

"IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma" sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

"IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları" sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

Tetikleme için önkoşullar

Tetikleme işlemini kullanmadan önce atılacak adımlar hakkında bilgi edinmek için bu bilgileri kullanın.

Uygulamanız tetiklemeden yararlanmadan önce aşağıdaki adımları tamamlayın:

1. Aşağıdakilerden birini yapın:
   a. Uygulama kuyruğunuz için bir başlatma kuyruğu yaratın. Örneğin:

```
DEFINE QLOCAL (initiation.queue) REPLACE      +
       LIKE (SYSTEM.DEFAULT.INITIATION.QUEUE)      +
       DESCR ('initiation queue description')
```

veya

b. Var olan ve uygulamanız tarafından kullanılabilen bir yerel kuyruğun adını belirleyin (genellikle bu ad SYSTEM.DEFAULT.INITIATION.QUEUE ya da kanalları tetikleyicilerle başlatıyorsanız, SYSTEM.CHANNEL.INITQ) ve uygulama kuyruğunun *InitiationQName* alanında adını belirtin.

2. Başlatma kuyruğunu uygulama kuyruğuyla ilişkilendirin. Bir kuyruk yöneticisi birden çok başlatma kuyruğuna sahip olabilir. Bazı uygulama kuyruklarınızın farklı programlar tarafından sunulmasını isteyebilirsiniz; bu durumda, her bir hizmet programı için bir başlatma kuyruğu kullanabilirsiniz, ancak buna gerek yoktur. Aşağıda bir uygulama kuyruğunun nasıl yaratılacağını gösteren bir örnek verilmiştir:

```
DEFINE QLOCAL (application.queue) REPLACE   +
LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE)           +
DESCR ('appl queue description')    +
INITQ (initiation.queue)          +
PROCESS (process.name)            +
TRIGGER                                     +
TRIGTYPE (FIRST)
```

IBM i IBM MQ for IBM i Denetim dili (CL) programından, başlatma kuyruğu yaratan bir alma işlemi:

```
/*   Queue used by AMQSINQA                                      */
            CRTMQMQ     QNAME('SYSTEM.SAMPLE.INQ')                 +
                        QTYPE(*LCL)  REPLACE(*YES)                 +
                        MQMNAME                                    +
                        TEXT('queue for AMQSINQA')                 +
                        SHARE(*YES)                /* Shareable   */+
                        DFTMSGPST(*YES)/* Persistent messages OK  */+
                                                                   +
                        TRGENBL(*YES)  /* Trigger control on      */+
                        TRGTYPE(*FIRST)/* Trigger on first message*/+
                        PRCNAME('SYSTEM.SAMPLE.INQPROCESS')        +
                        INITQNAME('SYSTEM.SAMPLE.TRIGGER')
```

3. Bir uygulamayı tetikliyorsanız, uygulama kuyruğunuza hizmet edecek uygulamayla ilgili bilgileri içerecek bir süreç tanımlaması nesnesi yaratın. Örneğin, PAYR adlı bir CICS bordro işlemini tetiklemek için:

```
DEFINE PROCESS (process.name) +
       REPLACE +
       DESCR ('process description') +
       APPLICID ('PAYR') +
       APPLTYPE (CICS) +
       USERDATA ('Payroll data')
```

IBM i IBM MQ for IBM i için bir CL programından, süreç tanımlaması nesnesi yaratan bir alma işlemi:

```
/*   Process definition                                          */
            CRTMQMPRC   PRCNAME('SYSTEM.SAMPLE.INQPROCESS')      +
                        REPLACE(*YES)                            +
                        MQMNAME                                  +
                        TEXT('trigger process for AMQSINQA')     +
                        ENVDATA('JOBPTY(3)') /* Submit parameter */+
                        APPID('AMQSINQA')    /* Program name    */
```

Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, süreç tanımlaması nesnesinin özniteliklerinden gelen bilgileri tetikleyici iletisine kopyalar.

Hizmet olarak sunulan	Süreç tanımlaması nesnesi yaratmak için
AIX, Linux, and Windows sistemleri	DEFINE PROCESS ya da SYSTEM.DEFAULT.PROCESS ve ALTER PROCESS komutunu kullanarak değiştirin
z/OS z/OS z/OS	DEFINE PROCESS kullanın (adım “3” sayfa 834 içindeki örnek koda bakın) ya da işlemleri ve denetim panolarını kullanın.
IBM i IBM i IBM i	“3” sayfa 834. adımda olduğu gibi kod içeren bir CL programı kullanın.

4. İsteğe bağlı: Bir iletim kuyruğu tanımı yaratın ve **ProcessName** özniteliği için boşluklar kullanın.

   TrigData özniteliği, tetiklenecek kanalın adını içerebilir ya da boş bırakılabilir. IBM MQ for z/OSdışında, boş bırakılırsa, kanal başlatıcısı, adı belirtilen iletim kuyruğuyla ilişkili bir kanal buluncaya kadar kanal tanımlama dosyalarını arar. Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, iletim kuyruğu tanımının **TrigData** özniteliğindeki bilgileri tetikleyici iletiye kopyalar.

5. Uygulama kuyruğunuza hizmet edecek uygulamanın özelliklerini belirtmek için bir süreç tanımlaması nesnesi yarattıysanız, süreç nesnesini kuyruğun **ProcessName** özniteliğinde adlandırarak uygulama kuyruğunuzla ilişkilendirin.

Hizmet olarak sunulan	Komutları kullan
AIX, Linux, and Windows sistemleri	ALTER QLOCAL (YEREL)
z/OS z/OS z/OS	ALTER QLOCAL (YEREL)
IBM i IBM i IBM i	CHGMQMQ

6. Tanımladığınız başlatma kuyruklarına hizmet verecek tetikleyici izleme programlarının IBM i (ya da IBM MQ for IBM i içindeki tetikleyici sunucularının) eşgörünümlerini başlatın. Ek bilgi için bkz. “Tetikleyici izleme programları tarafından başlatma kuyruğunun işlenmesi” sayfa 842 .

Teslim edilmemiş tetikleyici iletilerini bilmek istiyorsanız, kuyruk yöneticinizin tanımlı bir teslim edilmemiş (teslim edilmemiş) kuyruğu olduğundan emin olun. *DeadLetterQName* kuyruk yöneticisi alanında kuyruğun adını belirtin.

Daha sonra, uygulama kuyruğunuzu tanımlayan kuyruk nesnesinin özniteliklerini kullanarak, gerek duyduğunuz tetikleyici koşullarını ayarlayabilirsiniz. Daha fazla bilgi için “Tetikleyici olaylarının denetlenmesi” sayfa 839 başlıklı konuya bakın.

Tetikleme olayına ilişkin koşullar

Kuyruk yöneticisi, bu konuda ayrıntılı olarak açıklanan koşullar karşılandığında bir tetikleyici iletisi yaratır.

Bu konuda paylaşılan kuyruklara yapılan başvurular, bir kuyruk paylaşım grubundaki paylaşılan kuyruklar anlamına gelir; yalnızca IBM MQ for z/OSüzerinde kullanılabilir.

Aşağıdaki koşullar kuyruk yöneticisinin bir tetikleyici iletisi yaratmasına neden olur:

1. Bir ileti, kuyruğa *konan* bir iletidir.
2. İleti, kuyruğun eşik tetikleme önceliğine eşit ya da daha büyük bir önceliğe sahip. Bu öncelik **TriggerMsgPriority** yerel kuyruk özniteliğinde ayarlanır; sıfır olarak ayarlanırsa, her ileti nitelidir.
3. *TriggerType*' e bağlı olarak, öncelikleri *TriggerMsgPriority* değerinden büyük ya da bu değere eşit olan kuyruktaki iletilerin sayısı önceden belirlenmişti:

- Sıfır (MQTT_FIRST tetikleyici tipi için)
- Herhangi bir sayı (MQTT_EVERY tetikleyici tipi için)
- *TriggerDepth* eksi 1 (MQTT_DEPTH tetikleyici tipi için)

Not:

a. Paylaşılmayan yerel kuyruklar için, kuyruk yöneticisi bir tetikleyici olayına ilişkin koşulların var olup olmadığını değerlendirdiğinde hem kesinleştirilmiş hem de kesinleştirilmemiş iletileri sayar. Sonuç olarak, kuyruktaki iletiler kesinleştirilmediği için, alınacak ileti olmadığında bir uygulama başlatılabilir. Bu durumda, uygulamanın iletilerinin gelmesini beklemesi için bekleme seçeneğini uygun bir *WaitInterval*ile kullanmayı düşünün.

b. Yerel paylaşılan kuyruklar için, kuyruk yöneticisi yalnızca kesinleştirilmiş iletileri sayar.

4. FIRST ya da DEPTH tipinde tetikleme için, hiçbir programda iletileri kaldırmak üzere uygulama kuyruğu açık değildir (yani, **OpenInputCount** yerel kuyruk özniteliği sıfırdır).

Not:

a. Paylaşılan kuyruklar için, birden çok kuyruk yöneticisinin bir kuyruğa karşı çalışan tetikleyici izleme programları olduğunda özel koşullar geçerli olur. Bu durumda, bir ya da daha çok kuyruk yöneticisinin kuyruğu paylaşılan giriş için açıksa, diğer kuyruk yöneticilerindeki tetikleyici ölçütleri *TriggerType* MQTT_FIRST ve *TriggerMsgPriority* sıfır olarak değerlendirilir. Tüm kuyruk yöneticileri giriş için kuyruğu kapattığında, tetikleyici koşulları kuyruk tanımlamasında belirtilen koşullara geri döner.

Bu koşuldan etkilenen örnek bir senaryo QM1, QM2ve QM3 kuyruk yöneticileridir ve bir uygulama kuyruğu A için çalışan bir tetikleyici izleme programı vardır. A ' da tetikleme koşullarını karşılayan bir ileti gelir ve başlatma kuyruğunda bir tetikleyici iletisi üretilir. QM1 üzerindeki tetikleyici izleme programı, tetikleyici iletisini alır ve bir uygulamayı tetikler. Tetiklenen uygulama, paylaşılan giriş için uygulama kuyruğunu açar. Bu noktadan itibaren A uygulama kuyruğuna ilişkin tetikleme koşulları, QM1 uygulama kuyruğunu kapatıncaya kadar *TriggerType* MQTT_FIRST ve *TriggerMsgPriority* kuyruk yöneticilerindeki sıfır QM2 ve QM3olarak değerlendirilir.

b. Paylaşılan kuyruklar için, bu koşul her kuyruk yöneticisi için uygulanır. Yani, kuyruk yöneticisi tarafından kuyruk için bir tetikleyici iletisinin oluşturulabilmesi için kuyruk yöneticisinin *OpenInputCount* değeri sıfır olmalıdır. Ancak, kuyruk paylaşım grubundaki herhangi bir kuyruk yöneticisinde MQOO_INPUT_EXCLUSIVE seçeneği kullanılarak kuyruk açıksa, kuyruk paylaşım grubundaki kuyruk yöneticileri tarafından o kuyruk için tetikleyici ileti oluşturulmaz.

Tetiklenen uygulama giriş için kuyruğu açtığında, tetikleyici koşullarının değerlendiriliş şeklindeki değişiklik oluşur. Yalnızca bir tetikleyici izleme programının çalıştığı senaryolarda, uygulama kuyruğunu benzer şekilde giriş için açmaları nedeniyle diğer uygulamalar da aynı etkiye sahip olabilir. Uygulama kuyruğunun bir tetikleyici izleme programı tarafından başlatılan bir uygulama tarafından mı, yoksa başka bir uygulama tarafından mı açıldığının önemi yoktur; kuyruğun, tetikleyici ölçütlerinde değişikliğe neden olan başka bir kuyruk yöneticisinde giriş için açık olması önemlidir.

5. IBM MQ for z/OSsistemlerinde, uygulama kuyruğu MQUS_NORMAL **Usage** özniteliğine sahip bir kuyruksa, alma istekleri engellenmez (yani, **InhibitGet** kuyruk özniteliği MQQA_GET_ALLOWED). Ayrıca, tetiklenen uygulama kuyruğu MQUS_XMITQ ' nun **Usage** özniteliğine sahip bir kuyruksa, bunun için alma istekleri engellenmez.
6. Aşağıdakilerden birini yapın:
   - Kuyruğa ilişkin **ProcessName** yerel kuyruk özniteliği boş değil ve bu öznitelikle tanıtılan süreç tanımlaması nesnesi yaratıldı ya da
   - Kuyruğa ilişkin **ProcessName** yerel kuyruk özniteliği boş, ancak kuyruk bir iletim kuyruğu. Süreç tanımlaması isteğe bağlı olduğundan, **TriggerData** özniteliği başlatılacak kanalın adını da içerebilir. Bu durumda, tetikleyici iletisi aşağıdaki değerleri içeren öznitelikleri içerir:
     - **QName**: kuyruk adı
     - **ProcessName**: boşluklar

- **TriggerData**: tetikleyici verileri
- **ApplType**: MQAT_UNKNOWN
- **ApplId**: boşluklar
- **EnvData**: boşluklar
- **UserData**: boşluklar

7. Bir başlatma kuyruğu yaratıldı ve **InitiationQName** yerel kuyruk özniteliğinde belirtildi. Ayrıca:
   - Başlatma kuyruğu için alma istekleri engellenmez (**InhibitGet** kuyruk özniteliğinin değeri MQQA_GET_ALLOWED).
   - Başlangıç kuyruğu için koyma istekleri engellenmemelidir (**InhibitPut** kuyruk özniteliğinin değeri MQQA_PUT_ALLOWED olmalıdır).
   - Başlatma kuyruğunun **Usage** özniteliğinin değeri MQUS_NORMAL olmalıdır.
   - Dinamik kuyrukların desteklendiği ortamlarda, başlatma kuyruğu mantıksal olarak silinmiş olarak işaretlenmiş dinamik bir kuyruk olmamalıdır.
8. Bir tetikleyici izleme programında, iletileri kaldırmak için başlatma kuyruğu açık (yani, **OpenInputCount** yerel kuyruk özniteliği sıfırdan büyük).
9. Uygulama kuyruğuna ilişkin tetikleyici denetimi (**TriggerControl** yerel kuyruk özniteliği) MQTC_ON olarak ayarlandı. Bunu yapmak için, kuyruğunuzu tanımlarken **trigger** özniteliğini ayarlayın ya da ALTER QLOCAL komutunu kullanın.
10. Tetikleyici tipi (**TriggerType** yerel kuyruk özniteliği) MQTT_NONE değil.

    Tüm gerekli koşullar karşılanırsa ve tetikleyici koşuluna neden olan ileti bir iş biriminin parçası olarak konursa, tetikleyici ileti, iş birimi tamamlanıncaya kadar (iş birimi kesinleştirilinceye ya da MQTT_FIRST ya da MQTT_DEPTH tetikleyici tipi için) geri çekilinceye kadar tetikleyici izleyici uygulaması tarafından alınamaz.
11. A suitable message is placed on the queue, for a **TriggerType** of MQTT_FIRST or MQTT_DEPTH, and the queue:
    - Daha önce boş değildi (MQTT_FIRST) ya da
    - **TriggerDepth** ya da daha fazla ileti vardı (MQTT_DEPTH)

    “2” sayfa 835 - “10” sayfa 837 arasındaki koşullar (“3” sayfa 835hariç), MQTT_FIRST durumunda, bu kuyruk için son tetikleyici iletisinin yazılmasından bu yana yeterli bir aralık (**TriggerInterval** kuyruk yöneticisi özniteliği) geçtiyse karşılanır.

    Bu, kuyruktaki tüm iletileri işlemeden önce sona eren bir kuyruk sunucusuna izin verilmesini sağlar. Tetikleme aralığının amacı, oluşturulan yinelenen tetikleyici iletilerinin sayısını azaltmaktır.

    Not: Kuyruk yöneticisini durdurup yeniden başlatır ve **TriggerInterval** süreölçeri ilk durumuna getirilir. İki tetikleyici ileti üretilebilen küçük bir pencere vardır. Kuyruğun tetikleyici özniteliği bir ileti geldiğinde ve kuyruk daha önce boş olmadığında (MQTT_FIRST) ya da **TriggerDepth** ya da daha fazla ileti içermediğinde (MQTT_DEPTH) pencere var olur.
12. Bir kuyruğa hizmet veren tek uygulama, MQTT_FIRST ya da MQTT_DEPTH **TriggerType** için bir MQCLOSE çağrısı oluşturur ve en az:
    - Bir (MQTT_FIRST) ya da
    - **TriggerDepth** (MQTT_DEPTH)

    kuyruktaki yeterli önceliğe sahip iletiler (koşul “2” sayfa 835) ve “6” sayfa 836 ile “10” sayfa 837 arasındaki koşullar da karşılanır.

    Bu, MQGET çağrısı veren, kuyruğu boş bulan ve bu nedenle sona eren bir kuyruk sunucusuna izin vermek içindir; ancak, MQGET ile MQCLOSE çağrıları arasındaki aralıkta bir ya da daha fazla ileti gelir.

    Not:

a. Uygulama kuyruğuna hizmet veren program tüm iletileri almazsa, bu kapalı bir döngüye neden olabilir. Program kuyruğu her kapattığında, kuyruk yöneticisi tetikleyici izleme programının sunucu programını yeniden başlatmasına neden olacak başka bir tetikleyici iletisi yaratır.

b. Uygulama kuyruğuna hizmet veren program, kuyruğu kapatmadan önce alma isteğini geri çevirirse (ya da program olağandışı sona ererse), aynı durum oluşur. Ancak, program alma isteğini geri çekmeden önce kuyruğu kapatırsa ve kuyruk boşsa, tetikleyici iletisi yaratılmaz.

c. Böyle bir döngünün oluşmasını önlemek için, MQMD ' nin *BackoutCount* alanını kullanarak arka arkaya geriletilmiş iletileri saptayın. Daha fazla bilgi için bkz “Geriletilmeyen iletiler” sayfa 44.

13. MQSET ya da bir komut kullanılarak aşağıdaki koşullar karşılanır:

    a. • **TriggerControl** , MQTC_ON olarak değiştirildi ya da
    • **TriggerControl** zaten MQTC_ON ve **TriggerType**, **TriggerMsgPriority**ya da **TriggerDepth** (ilgili ise) değeri değiştirilmiştir,

    Ve en azından:

    • Bir (MQTT_FIRST ya da MQTT_EVERY) ya da
    • **TriggerDepth** (MQTT_DEPTH)

    yeterli önceliğe sahip kuyruktaki iletiler (koşul “2” sayfa 835) ve koşullar “4” sayfa 836 - “10” sayfa 837 (“8” sayfa 837 hariç) Onlar da memnun.

    Bu, tetikleme ölçütlerini değiştiren bir uygulamaya ya da işletmene, tetikleme koşullarının karşılanması durumunda izin verilmesini sağlar.

    b. Bir başlatma kuyruğunun **InhibitPut** kuyruk özniteliğinin değeri MQQA_PUT_INALLOWED değerinden MQQA_PUT_ALLOWED değerine değişir ve en az:

    • Bir (MQTT_FIRST ya da MQTT_EVERY) ya da
    • **TriggerDepth** (MQTT_DEPTH)

    yeterli önceliğe sahip iletiler (koşul “2” sayfa 835) Bunun başlangıç kuyruğu olduğu herhangi bir kuyrukta ve “4” sayfa 836 - “10” sayfa 837 arasındaki koşullar da karşılanır. (Koşulları karşılayan bu tür her kuyruk için bir tetikleyici ileti oluşturulur.)

    Bu, başlatma kuyruğundaki MQQA_PUT_INENGELLEME koşulu nedeniyle tetikleyici iletilerin oluşturulmamasına izin vermek içindir, ancak şimdi bu koşul değiştirilmiştir.

    c. Bir uygulama kuyruğunun **InhibitGet** kuyruk özniteliğinin değeri MQQA_GET_INALLOWED değerinden MQQA_GET_ALLOWED değerine değişiyor ve en azından:

    • Bir (MQTT_FIRST ya da MQTT_EVERY) ya da
    • **TriggerDepth** (MQTT_DEPTH)

    yeterli önceliğe sahip iletiler (koşul “2” sayfa 835) kuyrukta ve “4” sayfa 836 ile “10” sayfa 837arasındaki koşullar (“5” sayfa 836hariç) da karşılanır.

    Bu, uygulamaların yalnızca uygulama kuyruğundan ileti alabildiklerinde tetiklenmesini sağlar.

    d. Tetikleyici-izleyici uygulaması, başlatma kuyruğundan giriş için bir MQOPEN çağrısı yayınlar ve en az:

    • Bir (MQTT_FIRST ya da MQTT_EVERY) ya da
    • **TriggerDepth** (MQTT_DEPTH)

    yeterli önceliğe sahip iletiler (koşul “2” sayfa 835) Bunun başlatma kuyruğu olduğu herhangi bir uygulama kuyruğunda ve koşullar “4” sayfa 836 - “10” sayfa 837 (“8” sayfa 837 hariç) Bunlar da yerine getirilir ve başka bir uygulama giriş için başlatma kuyruğunu açmaz (koşulları karşılayan her kuyruk için bir tetikleyici ileti üretilir).

    Bu, tetikleyici izleme programı çalışmadığında kuyruklara gelen iletilere ve kaybolan kuyruk yöneticisi yeniden başlatma ve tetikleme iletilerine (kalıcı olmayan) izin verilmesini sağlar.

14. MSGDLVSQ doğru ayarlandı. MSGDLVSQ=FIFO değerini ayarlarsanız, iletiler kuyruğa First In First Out (İlk İlk Çıkış) temelinde teslim edilir. İletinin önceliği yoksayılır ve kuyruğun varsayılan önceliği

iletiye atanır. **TriggerMsgPriority** değeri, kuyruğun varsayılan önceliklerinden daha yüksek bir değere ayarlanırsa, hiçbir ileti tetiklenmez. **TriggerMsgPriority** , kuyruğun varsayılan önceliğine eşit ya da daha düşük bir değere ayarlanırsa, FIRST, EVERY ve DEPTH tipi için tetikleme gerçekleşir. Bu tiplerle ilgili bilgi için "Tetikleyici olaylarının denetlenmesi" sayfa 839altındaki **TriggerType** alanının açıklamasına bakın.

MSGDLVSQ=PRIORITY değerini ayarlarsanız ve ileti önceliği *TriggerMsgPriority* alanına eşit ya da ondan büyükse, iletiler yalnızca bir tetikleyici olayına doğru sayılır. Bu durumda, FIRST, EVERY ve DEPTH tipi için tetikleme gerçekleşir. Örneğin, **TriggerMsgPriority**' dan daha düşük önceliğe sahip 100 ileti koyarsanız, tetikleme amacıyla etkin kuyruk derinliği yine de sıfırdır. Daha sonra kuyruğa başka bir ileti koyarsanız, ancak bu kez öncelik **TriggerMsgPriority**değerinden büyük ya da bu değere eşitse, etkin kuyruk derinliği sıfırdan bire yükselir ve **TriggerType** FIRST koşulu karşılanır.

Notlar:

1. Adım "12" sayfa 837 ' den (uygulama kuyruğuna gelen bir ileti dışında bir olayın sonucu olarak tetikleyici iletilerin oluşturulduğu durumlarda), tetikleyici ileti bir iş biriminin parçası olarak yerleştirilmez. Ayrıca, **TriggerType** MQTT_EVERY ise ve uygulama kuyruğunda bir ya da daha fazla ileti varsa, yalnızca bir tetikleyici iletisi üretilir.
2. IBM MQ , MQPUT sırasında bir iletiyi bölümlerse, tüm kesimler kuyruğa başarıyla yerleştirilinceye kadar bir tetikleyici olayı işlenmez. Ancak, ileti bölümleri kuyruğa girdikten sonra, IBM MQ bunları tetikleme amacıyla tek tek iletiler olarak işler. Örneğin, üç parçaya bölünmüş tek bir mantıksal ileti, ilk MQPUT ve kesimlere ayrıldığında yalnızca bir tetikleyici olayın işlenmesine neden olur. Ancak bu üç bölümden her biri, IBM MQ ağı üzerinden taşınırken kendi tetikleyici olaylarının işlenmesine neden olur.
3. IBM MQ for z/OSiçin, paylaşılan kuyruk tetikleme ve paylaşılan kuyruğu barındıran Coupling Facility bağlantısı için ayarlandıysa, bir tetikleme olayı oluşturulabilir ve başlatma kuyruğuna bir ileti konabilir. Tetikleme için özgün paylaşılan kuyruk ayarlarına hiçbir ileti konmadığında da bu durum oluşabilir. Bunun nedeni, Liste Bildirim Vektörü' nde belgelendiği gibi, IXLVECTR makrosunun bitlerin aşırı gösteriminden kaynaklanır.

Tetikleyici olaylarının denetlenmesi

Tetikleyici olaylarını, uygulama kuyruğunuzu tanımlayan bazı öznitelikleri kullanarak denetleyebilirsiniz. Bu bilgiler, tetikleyici tiplerinin kullanılmasına ilişkin örnekler de verir: EVERY, FIRST ve DEPTH.

Tetiklemeyi etkinleştirebilir ve devre dışı bırakabilir ve bir tetikleme olayına doğru sayılacak iletilerin sayısını ya da önceliğini seçebilirsiniz. Nesnelerin öznitelikleriiçinde bu özniteliklerin tam açıklaması vardır.

İlgili öznitelikler şunlardır:

TriggerControl
Bir uygulama kuyruğuna ilişkin tetiklemeyi etkinleştirmek ve geçersiz kılmak için bu özniteliği kullanın.

TriggerMsgPriority
Bir iletinin bir tetikleme olayına doğru sayılması için sahip olması gereken öncelik alt sınırı. Uygulama kuyruğuna *TriggerMsgPriority* değerinden küçük bir öncelik iletisi gelirse, kuyruk yöneticisi tetikleyici ileti yaratılıp yaratılmayacağını belirlediğinde iletiyi yoksayar. *TriggerMsgPriority* sıfır olarak ayarlanırsa, tüm iletiler bir tetikleme olayına doğru sayılır.

TriggerType
Tetikleyici tipi NONE 'a ek olarak (*TriggerControl* ' nin OFF olarak ayarlanması gibi tetiklemeyi devre dışı bırakır), bir kuyruğun duyarlılığını olayları tetikleyecek şekilde ayarlamak için aşağıdaki tetikleyici tiplerini kullanabilirsiniz:

Her
Uygulama kuyruğuna her ileti geldiğinde bir tetikleyici olayı oluşur. Bir uygulamanın birden çok eşgörünümünün başlatılmasını istiyorsanız bu tetikleyici tipini kullanın.

Birinci
Tetikleme olayı yalnızca, uygulama kuyruğundaki iletilerin sayısı sıfırdan bire değiştiğinde oluşur. Bir hizmet programının kuyruğa ilk ileti geldiğinde başlamasını istiyorsanız, işlenecek başka ileti kalmayıncaya kadar devam edin ve sona erdirin. Boş oluncaya kadar kuyruğu her zaman işlemeniz gerekir. Ayrıca bkz. “FIRST tetikleyici tipinin özel durumu” sayfa 840.

Derinlik
Tetikleme olayı yalnızca, uygulama kuyruğundaki iletilerin sayısı **TriggerDepth** özniteliğinin değerine ulaştığında oluşur. Bu tip bir tetikleyicinin tipik kullanımı, bir istek kümesine tüm yanıtlar alındığında bir programı başlatmaktır.

Derinliğe göre tetikleme: Derinlik temelinde tetikleme ile kuyruk yöneticisi, bir tetikleyici iletisi oluşturduktan sonra tetiklemeyi (*TriggerControl* özniteliğini kullanarak) devre dışı bırakır. Bu gerçekleştikten sonra uygulamanız tetiklemeyi yeniden etkinleştirmelidir (MQSET çağrısıyla).

Tetikleyiciyi devre dışı bırakma işlemi eşitleme noktası denetimi altında olmadığından, bir iş birimi yedeklenerek tetikleme yeniden etkinleştirilemez. Bir program tetikleme olayına neden olan bir koyma isteğini geri çevirirse ya da program olağandışı biterse, MQSET çağrısını ya da ALTER QLOCAL komutunu kullanarak tetiklemeyi yeniden etkinleştirmeniz gerekir.

TriggerDepth
Derinlik temelinde tetikleme kullanılırken tetikleme olayına neden olan kuyruktaki iletilerin sayısı.

Bir kuyruk yöneticisinin tetikleyici ileti yaratması için karşılanması gereken koşullar “Tetikleme olayına ilişkin koşullar” sayfa 835içinde açıklanmıştır.

EVERY tetikleyici tipi kullanımı örneği

Motor sigortası için talepler oluşturan bir uygulamayı göz önünde bulundurun. Uygulama, her seferinde aynı yanıt kuyruğunu belirten bir dizi sigorta şirketine istek iletileri gönderebilir. Bu yanıt kuyruğunda EVERY tipinde bir tetikleyici ayarlayabilir; böylece, yanıt her geldiğinde, yanıtın sunucunun bir eşgörünümünün yanıtı işlemesini tetikleyebilir.

FIRST tetikleyici tipi kullanımı örneği

Her biri iş günlerinin ayrıntılarını merkez ofise ileten bir dizi şubeye sahip bir kuruluş düşünün. Hepsi aynı anda, çalışma gününün sonunda ve merkez ofiste tüm şubelerdeki detayları işleyen bir uygulama var. Merkez ofise gelen ilk mesaj, bu uygulamayı başlatan bir tetikleme olayına neden olabilir. Bu uygulama, kuyruğunda başka ileti kalmaıncaya kadar işlemeye devam edecek.

DEPTH tetikleyici tipi kullanımı örneği

Uçuş rezervasyonunu onaylamak, otel odası rezervasyonunu onaylamak, araba kiralamak ve bazı yolcuların çeklerini sipariş etmek için tek bir istek oluşturan bir seyahat acentesi uygulamasını göz önünde bulundurun. Uygulama bu öğeleri dört istek iletisine ayırarak her birini ayrı bir hedefe gönderebilir. Yalnızca dört yanıtın tümü geldiğinde yeniden başlatılabilmesi için, yanıt kuyruğunda DEPTH tipinde bir tetikleyici ayarlayabilir (derinlik 4 değerine ayarlanmış olarak).

Dört yanıtın sonundan önce yanıt kuyruğuna başka bir ileti (büyük olasılıkla farklı bir istekten) gelirse, istekte bulunan uygulama erken tetiklenir. Bunu önlemek için, bir isteğe birden çok yanıt toplamak üzere DEPTH tetikleme kullanılırken, her istek için her zaman yeni bir yanıt kuyruğu kullanın.

FIRST tetikleyici tipinin özel durumu

Tetikleyici tipi FIRST olduğunda, başka bir ileti geldiğinde uygulama kuyruğunda zaten bir ileti varsa, kuyruk yöneticisi genellikle başka bir tetikleyici iletisi yaratmaz.

Ancak, kuyruğa hizmet veren uygulama kuyruğu gerçekten açmayabilir (örneğin, uygulama bir sistem sorunu nedeniyle sona erebilir). Süreç tanımlaması nesnesine yanlış bir uygulama adı konduysa, kuyruğa hizmet veren uygulama iletilerin hiçbirini almayacaktır. Bu durumlarda, uygulama kuyruğuna başka bir ileti gelirse, bu iletiyi (ve kuyruktaki diğer iletileri) işlemek için çalışan bir sunucu yoktur.

Bununla başa çıkmak için, kuyruk yöneticisi aşağıdaki koşullar altında daha fazla tetikleyici ileti yaratır:

- Uygulama kuyruğuna başka bir ileti gelirse, ancak kuyruk yöneticisi o kuyruk için son tetikleyici iletiyi yarattığından bu yana önceden tanımlanmış bir zaman aralığı geçtiyse. Bu zaman aralığı, *TriggerInterval*kuyruk yöneticisi özniteliğinde tanımlanır. Varsayılan değeri 999 999 999 milisaniyedir.
- IBM MQ for z/OSişletim sistemi üzerinde, açık başlatma kuyruğuna ad veren uygulama kuyrukları düzenli aralıklarla taranır. Son tetikleyici iletisinin gönderilmesinden bu yana *TRIGINT* milisaniye geçildiyse ve kuyruk bir tetikleyici olayına ilişkin koşulları karşılıyorsa ve CURDEPTH sıfırdan büyükse, bir tetikleyici iletisi üretilir. Bu işleme backstop triggering adı verilir.

Uygulamanızda kullanılacak tetikleme aralığı için bir değere karar verirken aşağıdaki noktaları göz önünde bulundurun:

- *TriggerInterval* değerini düşük bir değere ayarlarsanız ve uygulama kuyruğuna hizmet veren bir uygulama yoksa, FIRST tetikleyici tipi EVERY tetikleyici tipi gibi davranabilir. Bu, iletilerin uygulama kuyruğuna yerleştirilme hızına bağlıdır ve bu da diğer sistem etkinliğine bağlı olabilir. Bunun nedeni, tetikleme aralığı çok küçükse, tetikleyici tipi EVERY değil FIRST olsa da, uygulama kuyruğuna her ileti yerleştirildiğinde başka bir tetikleyici iletisi oluşturulmasıdır. (Tetikleme aralığı sıfır olan FIRST tipi, tetikleme tipi EVERY ile eşdeğerdir.)
- IBM MQ for z/OS ' ta *TRIGINT* değerini düşük bir değere ayarlarsanız ve tetikleyici tipi FIRST uygulama kuyruğuna hizmet veren bir uygulama yoksa, arka plan tetikleme işlemi, adı açık başlatma kuyrukları olan uygulama kuyruklarının periyodik taraması her gerçekleşişinde bir tetikleyici iletisi oluşturur.
- Bir iş birimi geriletildiyse (bkz. İletileri ve iş birimlerini tetikle) ve tetikleme aralığı yüksek bir değere (ya da varsayılan değere) ayarlandığında, iş birimi geriletildiğinde bir tetikleyici iletisi oluşturulur. Ancak, tetikleme aralığını düşük bir değere ya da sıfıra ayarladıysanız (FIRST tetikleyici tipinin EVERY tetikleyici tipi gibi davranmasına neden oluyorsa), birçok tetikleyici iletisi oluşturulabilir. İş birimi geriletildiyse, tüm tetikleyici iletileri kullanılır kılmıştır. Oluşturulan tetikleyici iletilerinin sayısı, tetikleme aralığına bağlıdır. Tetikleme aralığı sıfır olarak ayarlanırsa, ileti sayısı üst sınırı oluşturulur.

Tetiklenen kuyrukları kullanan bir uygulama tasarlanması

Uygulamalarınızı nasıl kuracağınızı, denetleyeceğinizi ve tetikleyeceğinizi gördünüz. Burada, uygulamanızı tasarlarken göz önünde bulundurmanız gereken bazı ipuçlarını bulabilirsiniz.

Tetikleyici iletileri ve iş birimleri

Bir iş biriminin parçası olmayan tetikleyici olayları nedeniyle yaratılan tetikleyici iletiler, başlatma kuyruğuna, herhangi bir iş biriminin dışına, başka iletilere bağımlı olmadan konur ve tetikleyici izleme programı tarafından hemen alınabilir.

Bir iş biriminin parçası olan tetikleyici olaylar nedeniyle yaratılan tetikleyici iletiler, UOW çözüldüğünde, iş biriminin kesinleştirilip kesinleştirilmemesinden ya da geriletilmesinden bağımsız olarak, başlatma kuyruğunda kullanılır kıllır.

Kuyruk yöneticisi bir başlatma kuyruğuna tetikleyici ileti yerleştiremezse, bu ileti teslim edilmeyen ileti kuyruğuna konur.

Not:

1. Kuyruk yöneticisi, bir tetikleme olayına ilişkin koşulların var olup olmadığını değerlendirdiğinde hem kesinleştirilmiş hem de kesinleştirilmemiş iletileri sayar.

   FIRST ya da DEPTH tipinde tetikleme ile, gerekli koşullar karşılandığında tetikleyici iletisi her zaman kullanılabilir olacak şekilde, iş birimi geriletilse bile tetikleyici iletiler kullanılabilir duruma getirilebilir. Örneğin, FIRST tetikleyici tipiyle tetiklenen bir kuyruğa ilişkin iş birimi içindeki bir koyma isteğini göz önünde bulundurun. Bu, kuyruk yöneticisinin bir tetikleyici iletisi yaratmasına neden olur. Başka bir iş biriminden başka bir koyma isteği oluşursa, uygulama kuyruğundaki iletilerin sayısı birden ikiye değiştiği için bu başka bir tetikleme olayına neden olmaz; bu durum, bir tetikleme olayına ilişkin koşulları karşılamaz. Şimdi ilk iş birimi geri çekilirse, ancak ikincisi kesinleştirilirse, yine de bir tetikleyici iletisi yaratılır.

Ancak bu, tetikleyici iletilerin bazen bir tetikleyici olayına ilişkin koşullar karşılanmadığında yaratıldığı anlamına gelir. Tetikleme kullanan uygulamalar her zaman bu durumla başa çıkmak için hazırlanmalıdır. *WaitInterval* değerini uygun bir değere ayarlayarak MQGET çağrısıyla bekleme seçeneğini kullanmanız önerilir.

Yaratılan tetikleyici iletileri, iş biriminin geriletilmesi ya da kesinleştirilmesi gibi her zaman kullanılabilir kılmıştır.

2. Yerel paylaşılan kuyruklar (yani, bir kuyruk paylaşım grubundaki paylaşılan kuyruklar) için, kuyruk yöneticisi yalnızca kesinleştirilmiş iletileri sayar.

Tetiklenen kuyruktan ileti alma

Tetikleme kullanan uygulamalar tasarladığınızda, tetikleyici izleme programının başlatılması ile uygulama kuyruğunda kullanılabilecek diğer iletiler arasında bir gecikme olabileceğini unutmayın. Tetikleme olayına neden olan ileti diğerlerinden önce kesinleştirildiğinde bu oluşabilir.

İletilerin gelmesine izin vermek için, tetikleyici koşullarının ayarlandığı bir kuyruktan iletileri kaldırmak için MQGET çağrısını kullandığınızda her zaman bekleme seçeneğini kullanın. *WaitInterval* , verilmekte olan bir iletiyle, çağrıyı kesinleştirilmekte olan ileti arasındaki en uzun makul süreyi sağlamak için yeterli olmalıdır. İleti uzak bir kuyruk yöneticisinden geliyorsa, bu sefer aşağıdakilerden etkilenir:

- Kesinleştirilmeden önce konan iletilerin sayısı
- İletişim bağlantısının hızı ve kullanılabilirliği
- İletilerin boyutları

Bekleme seçeneğiyle MQGET çağrısının kullanılması gereken bir durum örneği için, iş birimlerini tanımlarken kullandığımız örnekle aynı örneği göz önünde bulundurun. Bu, FIRST tetikleyici tipiyle tetiklenen bir kuyruğa ilişkin iş birimi içindeki bir koyma isteğiydi. Bu olay, kuyruk yöneticisinin bir tetikleyici iletisi yaratmasına neden olur. Başka bir iş biriminden başka bir koyma isteği oluşursa, uygulama kuyruğundaki iletilerin sayısı sıfırdan bire değişmediği için bu başka bir tetikleme olayına neden olmaz. Şimdi ilk iş birimi geri çekilirse, ancak ikincisi kesinleştirilirse, yine de bir tetikleyici iletisi yaratılır. Tetikleyici ileti, ilk iş biriminin geriletildiği sırada yaratılır. İkinci ileti kesinleştirilmeden önce önemli bir gecikme olursa, tetiklenen uygulamanın bunu beklemesi gerekebilir.

DEPTH tipinde tetiklemelerle, tüm ilgili iletiler kesinleştirilse bile bir gecikme oluşabilir. **TriggerDepth** kuyruk özniteliğinin 2 değerine sahip olduğunu varsayın. Kuyruğa iki ileti geldiğinde, ikinci ileti bir tetikleyici iletisinin yaratılmasına neden olur. Ancak, ikinci ileti kesinleştirilecek ilk iletiyse, tetikleyici ileti o zaman kullanılabilir olur. Tetikleyici izleme programı sunucu programını başlatır, ancak program, ilk ileti kesinleştirilinceye kadar yalnızca ikinci iletiyi alabilir. Bu nedenle, programın ilk iletinin kullanıma sunulmasını beklemesi gerekebilir.

Bekleme aralığınızın süresi dolduğunda, alma işlemi için kullanılabilecek ileti yoksa, uygulamanızı sonlandırılacak şekilde tasarlayın. Bir ya da daha fazla ileti daha sonra gelirse, bunları işlemek için uygulamanızın alınmasına güvenin. Bu yöntem, uygulamaların boşta durmasını ve gereksiz yere kaynak kullanmasını önler.

Tetikleyici izleme programları tarafından başlatma kuyruğunun işlenmesi

Bir kuyruk yöneticisinde, tetikleyici izleme programı, kuyruğa hizmet veren diğer uygulamalar gibidir. Ancak, tetikleyici izleme programı başlatma kuyruklarına hizmet verir.

Tetikleyici genellikle sürekli çalışan bir programdır. Başlatma kuyruğuna bir tetikleyici iletisi geldiğinde, tetikleyici izleme programı o iletiyi alır. Uygulama kuyruğundaki iletileri işlemek üzere uygulamayı başlatmak için bir komut yayınlamak üzere iletideki bilgileri kullanır.

Programın doğru uygulama kuyruğunda doğru işlemleri gerçekleştirebilmesi için, tetikleyici izleme programının başlatmış olduğu programa yeterli bilgi aktarması gerekir.

Kanal başlatıcı, ileti kanalı aracıları için özel bir tetikleyici izleyicisi tipi örneğidir. Ancak bu durumda, tetikleyici tipi FIRST ya da DEPTH olmalıdır.

ALW *AIX, Linux, and Windows sistemlerinde izleme programlarını tetikle*

Bu konu, AIX, Linux, and Windows sistemlerinde sağlanan tetikleyici izleme programlarına ilişkin bilgileri içerir.

Sunucu ortamı için aşağıdaki tetikleyici izleme programları sağlanır:

amqstrg0

Bu, **runmqtrm**tarafından sağlanan işlevin bir alt kümesini sağlayan örnek bir tetikleyici izleyicidir. amqstrg0hakkında daha fazla bilgi için bkz. "Multiplatforms üzerinde örnek programların kullanılması" sayfa 1014 .

mqtrm çalıştırma

Bu komutun sözdizimi şöyledir: **runmqtrm** [*-m QMgrName*] [*-q InitQ*]; burada QMgrName kuyruk yöneticisidir ve InitQ başlatma kuyruğudur. Varsayılan kuyruk SYSTEM.DEFAULT.INITIATION.QUEUE QUEUE değerini belirleyin. Uygun tetikleyici iletileri için programları çağırır. Bu tetikleyici, varsayılan uygulama tipini destekler.

Tetikleyici izleme programı tarafından işletim sistemine geçirilen komut dizgisi aşağıdaki gibi oluşturulur:

1. İlgili PROCESS tanımlamasından *ApplId* (yaratıldıysa)
2. Çift tırnak içine alınmış MQTMC2 yapısı
3. İlgili PROCESS tanımlamasından *EnvData* (yaratıldıysa)

Burada *ApplId* , komut satırına girileceği şekilde çalıştırılacak programın adıdır.

Geçirilen değiştirge MQTMC2 karakter yapısıdır. Sistem komutunun bu dizgiyi tek bir değiştirge olarak kabul etmesi için, tam olarak belirtildiği gibi, çift tırnak işareti içinde bu dizgiyi içeren bir komut dizgisi çağrılır.

Tetikleyici izleme programı, yeni başlattığı uygulama tamamlanıncaya kadar, başlatma kuyruğunda başka bir ileti olup olmadığını görmek için bakmaz. Uygulamanın yapacak çok işi varsa, tetikleyici izleme programı gelen tetikleyici iletilerinin sayısını takip edemeyebilir. İki seçeneğiniz var:

- Çalışan daha fazla tetikleyici izleme programı var
- Başlatılan uygulamaları arka planda çalıştır

Çalışmakta olan tetikleyici izleme programınız varsa, herhangi bir zamanda çalışabilecek uygulama sayısı üst sınırını denetleyebilirsiniz. Uygulamaları artalanda çalıştırırsanız, çalışabilecek uygulamaların sayısı için IBM MQ tarafından uygulanan bir kısıtlama yoktur.

Linux AIX Başlatılan uygulamayı AIX and Linuxarka planda çalıştırmak için, PROCESS tanımlamasının *EnvData* sonuna bir & koyun.

Başlatılan uygulamayı Windows sistemlerinde artalanda çalıştırmak için, *ApplId* alanında uygulamanızın adına bir START komutuyla önek ekleyin. Örneğin:

```
START ?B AMQSECHA
```

Not: Windows Bir Windows yolunda yol adının bir parçası olarak boşluklar varsa, bunlar tırnak işareti (") içine alınmalıdır tek bir bağımsız değişken olarak işlenmesini sağlamak için. Örneğin, "C:\Program Files\Application Directory\Application.exe".

Aşağıda, dosya adının yolun bir parçası olarak boşluk içerdiği bir APPLICID dizgisi örneği verilmiştir:

```
START "" /B "C:\Program Files\Application Directory\Application.exe"
```

Örnekteki Windows START komutunun sözdizimi, çift tırnak içine alınmış boş bir dizgi içerir. START, tırnak işaretlerindeki ilk bağımsız değişkenin yeni komutun başlığı olarak kabul edileceğini belirtir. Windows 'in uygulama yolunu bir' title ' bağımsız değişkeniyle karıştırmadığından emin olmak için, uygulama adından önce komuta çift tırnak işareti içine alınmış bir başlık dizgisi ekleyin.

IBM MQ istemcisi için aşağıdaki tetikleyici izleme programları sağlanır:

mqtmc
Bu, IBM MQ MQI client kitaplıklarına bağlandığı dışında runmqtrm ile aynıdır.

ALW *CICS için tetikleyici*

CICSiçin amqltmc0 tetikleyici izleyicisi sağlanır. Standart tetikleyici izleyicisi runmqtrm ile aynı şekilde çalışır, ancak bunu farklı bir şekilde çalıştırır ve CICS hareketlerini tetikler.

Bu konu yalnızca Windows, AIX and Linux x86-64 sistemleri için geçerlidir.

Tetikleyici izleyicisi CICS programı olarak verilir; 4 karakterlik bir hareket adıyla tanımlayın. Tetikleyici izleme programını başlatmak için 4 karakterlik bir ad girin. Varsayılan kuyruk yöneticisini (qm.ini dosyasında ya da IBM MQ for Windowskaydında adlandırıldığı gibi) ve SYSTEM.CICS.INITIATION.QUEUE.

Farklı bir kuyruk yöneticisi ya da kuyruk kullanmak istiyorsanız, tetikleyici izleyicisi MQTMC2 yapısını oluşturun: Yapı, parametre olarak eklenemeyecek kadar uzun olduğundan EXEC CICS START çağrısıyla bir program yazmanızı gerektirir. Daha sonra, MQTMC2 yapısını tetikleyici izleme programına ilişkin START isteğine veri olarak geçirin.

MQTMC2 yapısını kullandığınızda, tetikleyici izleyiciye yalnızca *StrucId*, *Version*, *QName*ve **QMgrName** değiştirgelerini sağlamanız gerekir; bu değiştirgeler başka alanlara gönderme yapmaz.

İletiler başlatma kuyruğundan okunur ve CICS hareketlerini başlatmak için kullanılır (EXEC CICS START kullanılarak), tetikleyici iletideki APPL_TYPE değerinin MQAT_CICS olduğu varsayılarak. İletilerin başlatma kuyruğundan okunması CICS eşitleme noktası denetimi altında gerçekleştirilir.

İletiler, izleme programı başlatıldığında ve durduğunda ve bir hata oluştuğunda oluşturulur. Bu iletiler CSMT geçici veri kuyruğuna gönderilir.

Çizelge 129. Tetikleyici izleme programının kullanılabilir sürümleri.

İki sütunlu bir tablo. İlk kolon, tetikleyici izleme programının kullanılabilir sürümlerini listeler ve ikinci kolon, her sürümün hangi platformlar için kullanılacağını gösterir.

Sürüm	**Kullanım**
amqltmc0	TXSeries için: • AIX AIX • Linux Linux x86-64 sistemleri
amqltmc4	Windows TXSeries for Windows 5.1
amqltmcc	CICS tetikleyici izleyicisinin istemciye bağlı sürümü

Diğer ortamlar için tetikleyici izleyici gerekiyorsa, kuyruk yöneticisinin başlatma kuyruklarına koyduğu tetikleyici iletilerini işleyebilecek bir program yazın. Bu tür bir program aşağıdaki işlemleri gerçekleştirmelidir:

1. Bir iletinin başlatma kuyruğuna gelmesini beklemek için MQGET çağrıyı kullanın.
2. Başlatılacak uygulamanın adını ve içinde çalıştığı ortamı bulmak için, tetikleyici iletisinin MQTM yapısının alanlarını inceleyin.
3. Ortama özgü bir başlatma komutu verin.

   z/OS Örneğin, z/OS toplu işte, iç okuyucuya bir iş gönderin.
4. Gerekiyorsa, MQTM yapısını MQTMC2 yapısına dönüştürün.
5. MQTMC2 ya da MQTM yapısını başlatılan uygulamaya geçirin. Bu, kullanıcı verilerini içerebilir.

6. Bu kuyruğa hizmet verecek uygulamayı uygulama kuyruğunuzla ilişkilendirin. Bunu, kuyruğun **ProcessName** özniteliğinde süreç tanımlaması nesnesini (yaratıldıysa) adlandırarak yaparsınız. Süreç tanımlaması nesnesini adlamak için **DEFINE QLOCAL** ya da **ALTER QLOCAL** komutunu kullanabilirsiniz.

   IBM i IBM iüzerinde CRTMQMQ ya da CHGMQMQ da kullanılabilir.

Tetikleyici izleyicisi arabirimine ilişkin ek bilgi için MQTMC2başlıklı konuya bakın.

IBM i *IBM i üzerinde izleme programlarını tetikle*

IBM iüzerinde, **runmqtrm** denetim komutu yerine IBM MQ for IBM i CL komutunu **STRMQMTRM**kullanın.

STRMQMTRM komutunu aşağıdaki gibi kullanın:

```
STRMQMTRM INITQNAME(InitQ) MQMNAME(QMgrName)
```

Ayrıntılar runmqtrm ile ilgili.

Ayrıca, kendi tetikleyici izleme programlarınızı yazmak için model olarak kullanabileceğiniz aşağıdaki örnek programlar da sağlanır:

AMQSTRG4
Bu, başlatılacak işlem için bir IBM i işi gönderen bir tetikleyici izleyicidir; ancak, her tetikleyici iletisiyle ilişkili ek işlemler olduğu anlamına gelir.

AMQSERV4
Bu bir tetikleyici sunucusudur. Her tetikleyici ileti için bu sunucu, işleme ilişkin komutu kendi işinde çalıştırır ve CICS hareketlerini çağırabilir.

Tetikleyici izleyicisi ve tetikleyici sunucusu, başlatdıkları programlara MQTMC2 yapısı geçirir. Bu yapının tanımı için bkz. MQTMC2. Bu örneklerin her ikisi de hem kaynak hem de yürütülebilir formlarda teslim edilir.

Bu tetikleyici izleme programları yalnızca yerli IBM i programlarını çağırabildiğinden, Java sınıfları IFS ' de bulunduğundan Java programlarını doğrudan tetikleyemezler. Ancak Java programları, Java programını çağıran ve TMC2 yapısından geçen bir CL programını tetikleyerek dolaylı olarak tetiklenebilir. TMC2 yapısının büyüklük alt sınırı 732 bayttır.

Örnek bir CLP ' nin kaynağı:

```
PGM PARM(&TMC2)
  DCL &TMC2 *CHAR LEN(800)
    ADDENVVAR ENVVAR(TM) VALUE(&TMC2)
    QSH  CMD('java_pgmname $TM')
    RMVENVVAR ENVVAR(TM)
ENDPGM
```

IBM MQ MQI clientiçin aşağıdaki tetikleyici izleme programı sağlanır: RUNMQTMC

RUNMQTMC ' yi aşağıdaki gibi çağırın:

```
CALL PGM(QMQM/RUNMQTMC) PARM('-m' QMgrName '-q' InitQ)
```

Tetikleyici iletilerinin özellikleri

Aşağıdaki konularda tetikleyici iletilerin diğer bazı özellikleri açıklanmaktadır.

- “Tetikleyici iletilerin kalıcılığı ve önceliği” sayfa 846
- “Kuyruk yöneticisi yeniden başlatma ve tetikleme iletileri” sayfa 846
- “İletileri ve nesne özniteliklerinde yapılan değişiklikleri tetikle” sayfa 846
- “Tetikleyici iletilerinin biçimi” sayfa 846

Tetikleyici iletilerin kalıcılığı ve önceliği

Tetikleyici iletileri, bu iletilere gerek olmadığı için kalıcı değildir.

Ancak, tetikleyici olayların oluşturulmasına ilişkin koşullar devam ediyor; bu nedenle, bu koşullar karşılandığında tetikleyici iletiler üretiliyor. Bir tetikleyici iletisi kaybolursa, uygulama kuyruğunda uygulama iletisinin var olmaya devam etmesi, kuyruk yöneticisinin tüm koşullar yerine gelir gelmez bir tetikleyici iletisi oluşturmasını garanti eder.

Bir iş birimi geriye işlenirse, oluşturduğu tetikleyici iletiler her zaman teslim edilir.

Tetikleyici iletiler, başlatma kuyruğunun varsayılan önceliğini alır.

Kuyruk yöneticisi yeniden başlatma ve tetikleme iletileri

Bir kuyruk yöneticisinin yeniden başlatılmasının ardından, giriş için bir başlatma kuyruğu açıldığında, kendisiyle ilişkilendirilmiş bir uygulama kuyruğunda iletiler varsa ve tetikleme için tanımlanmışsa, bu başlatma kuyruğuna bir tetikleyici ileti yerleştirilebilir.

İletileri ve nesne özniteliklerinde yapılan değişiklikleri tetikle

Tetikleyici iletiler, tetikleyici olayı sırasında yürürlükte olan tetikleyici özniteliklerinin değerlerine göre yaratılır.

Tetikleyici iletisi, daha sonra (oluşturulmasına neden olan ileti bir iş birimi içine konduğu için) tetikleyici izleme programı tarafından kullanılabilir kılınmazsa, bu sırada tetikleyici özniteliklerinde yapılan değişikliklerin tetikleyici ileti üzerinde bir etkisi olmaz. Özellikle, tetikleyicinin devre dışı bırakılması, bir tetikleyici iletisinin yaratıldıktan sonra kullanılabilir kılınmasını önlemez. Ayrıca, tetikleyici iletisinin kullanılabilir kılındığı sırada uygulama kuyruğu artık var olmayabilir.

Tetikleyici iletilerinin biçimi

Bir tetikleyici iletisinin biçimi MQTM yapısı tarafından tanımlanır.

Bu alan, kuyruk yöneticisinin tetikleyici iletisini yarattığında doldurduğu, uygulama kuyruğunun nesne tanımlamalarındaki ve o kuyrukla ilişkili işlemdeki bilgileri kullanarak aşağıdaki alanları içerir:

StrucId
Yapı tanıtıcısı.

Version
Yapının sürümü.

QName
Tetikleme olayının oluştuğu uygulama kuyruğunun adı. Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, uygulama kuyruğunun **QName** özniteliğini kullanarak bu alanı doldurur.

ProcessName
Uygulama kuyruğuyla ilişkili süreç tanımlaması nesnesinin adı. Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, uygulama kuyruğunun **ProcessName** özniteliğini kullanarak bu alanı doldurur.

TriggerData
Tetikleyici izleme programı tarafından kullanılacak serbest biçimli alan. Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, uygulama kuyruğunun **TriggerData** özniteliğini kullanarak bu alanı doldurur. IBM MQ for z/OSdışındaki herhangi bir IBM MQ ürününde, bu alan tetiklenecek kanalın adını belirtmek için kullanılabilir.

ApplType
Tetikleyici izleme programının başlatılacak uygulamanın tipi. Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, *ProcessName*içinde tanımlanan süreç tanımlaması nesnesinin **ApplType** özniteliğini kullanarak bu alanı doldurur.

ApplId
Tetikleyici izleme programının başlatılacak uygulamayı tanıtan bir karakter dizilimi. Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, *ProcessName*içinde tanımlanan süreç tanımlaması nesnesinin **ApplId** özniteliğini kullanarak bu alanı doldurur.

CICStarafından sağlanan tetikleyici izleyicisi CKTI ' yı kullandığınızda, süreç tanımlaması nesnesinin **ApplId** özniteliği bir CICS işlem tanıtıcısıdır.

IBM MQ for z/OStarafından sağlanan CSQQTRMN ' yi kullandığınızda, süreç tanımlaması nesnesinin **ApplId** özniteliği bir IMS hareket tanıtıcısıdır.

EnvData
Tetikleyici izleme programı tarafından kullanılacak ortamla ilgili verileri içeren karakter alanı. Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, *ProcessName*içinde tanımlanan süreç tanımlaması nesnesinin **EnvData** özniteliğini kullanarak bu alanı doldurur. CICStarafından sağlanan tetikleyici izleme programı (CKTI) ya da IBM MQ for z/OStarafından sağlanan tetikleyici izleme programı (CSQQTRMN) bu alanı kullanmaz, ancak diğer tetikleyici izleme programları bu alanı kullanmayı seçebilir.

UserData
Tetikleyici izleme programı tarafından kullanılacak kullanıcı verilerini içeren karakter alanı. Kuyruk yöneticisi bir tetikleyici iletisi yarattığında, *ProcessName*içinde tanımlanan süreç tanımlaması nesnesinin **UserData** özniteliğini kullanarak bu alanı doldurur. Bu alan, tetiklenecek kanalın adını belirtmek için kullanılabilir.

MQTM' de tetikleyici ileti yapısının tam açıklaması vardır.

Tetikleme çalışmadığında

Tetikleme programı programı başlatamadıysa ya da kuyruk yöneticisi tetikleme iletisini teslim edemezse, program tetiklenmez. Örneğin, süreç nesnesindeki applid değeri, programın artalanda başlatılacağını belirtmelidir; tersi durumda, tetikleyici programı başlatamaz.

Bir tetikleyici iletisi yaratılır, ancak başlatma kuyruğuna konulamazsa (örneğin, kuyruk dolu olduğu için ya da tetikleyici iletisinin uzunluğu, başlatma kuyruğu için belirlenen ileti uzunluğu üst sınırından fazlaysa), tetikleyici ileti, teslim edilmeyen ileti kuyruğuna yerleştirilir.

Teslim edilmeyen ileti kuyruğuna yerleştirme işlemi başarıyla tamamlanamazsa, tetikleyici iletisi atılır ve z/OS z/OS konsoluna ya da sistem işletmenine bir uyarı iletisi gönderilir ya da hata günlüğüne yazılır.

Tetikleyici iletisinin gönderilmeyen ileti kuyruğuna konması, o kuyruk için bir tetikleyici iletisi oluşturabilir. Bu ikinci tetikleyici ileti, teslim edilmeyen ileti kuyruğuna bir ileti eklerse atılır.

Program başarıyla tetiklenirse, ancak kuyruktan iletiyi almadan önce olağandışı biterse, bir izleme yardımcı programı kullanın (örneğin, CICS AUXTRACE (program CICS altında çalışıyorsa)). hata nedenini bulmak için.

MQI ve kümelerle çalışma

Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

Kümelerle kullanılmak üzere çağrılarda ve dönüş kodlarında kullanılabilir seçenekler hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “MQOPEN ve kümeler” sayfa 848
- “MQPUT, MQPUT1 ve kümeler” sayfa 849
- “MQINQ ve kümeler” sayfa 850
- “MQSET ve kümeler” sayfa 850
- “Dönüş kodları” sayfa 850

İlgili kavramlar
“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692

İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

"Kuyruk yöneticisine bağlanma ve bağlantı kesme" sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

"Nesnelerin açılması ve kapatılması" sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

"Kuyruktaki iletilerin konması" sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

"Kuyruktan ileti alma" sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

"Nesne öznitelikleri hakkında sorma ve bunları ayarlama" sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

"İş birimlerinin kesinleştirilmesi ve geri çekilmesi" sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

"Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması" sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

"IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma" sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

"IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları" sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

MQOPEN ve kümeler
Bir küme kuyruğu açıldığında iletinin konduğu ya da okunduğu kuyruk, MQOPEN çağrısına bağlıdır.

Hedef kuyruğun seçilmesi

MQODnesne tanımlayıcısında bir kuyruk yöneticisi adı sağlamazsanız, kuyruk yöneticisi iletinin gönderileceği kuyruk yöneticisini seçer. Nesne tanımlayıcıda bir kuyruk yöneticisi adı belirttiyseniz, iletiler her zaman seçtiğiniz kuyruk yöneticisine gönderilir.

Kuyruk yöneticisi hedef kuyruk yöneticisini seçiyorsa, seçim bağ tanımlama seçeneklerine (MQOO_BIND_*) ve yerel bir kuyruk olup bulunmadığına bağlıdır. Kuyruğun yerel bir eşgörünümü varsa, CLWLUSEQ özniteliği ANYolarak ayarlanmadıkça, her zaman uzak bir yönetim ortamı için tercih edilir. Tersi durumda, seçim bağ tanımlama seçeneklerine bağlıdır. Gruptaki tüm iletilerin aynı hedefte işlendiğinden emin olmak için kümelerle ileti grupları kullanılırken MQOO_BIND_ON_OPEN ya da MQOO_BIND_ON_GROUP belirtilmelidir.

Kuyruk yöneticisi hedef kuyruk yöneticisini seçiyorsa, iş yükü yönetimi algoritmasını kullanarak bunu çevrimsel sıralı olarak yapar; bkz. Kümeler için iş yükü dengeleme.

İş yükü dengeleme algoritması kullanıldığında, küme kuyruğunun nasıl açıldığı bağlıdır:

- MQOO_BIND_ON_OPEN -algoritma, kuyruk uygulama tarafından açıldığında bir kez kullanılır.
- MQOO_BIND_NOT_FIXED -algoritma, kuyruğa konan her ileti için kullanılır.
- MQOO_BIND_ON_GROUP -algoritma, her ileti grubunun başında bir kez kullanılır.

MQOO_BIND_ON_OPEN
MQOPEN çağrısında MQOO_BIND_ON_OPEN seçeneği, hedef kuyruk yöneticisinin düzeltilececeğini belirtir. Bir küme içinde aynı kuyruğun birden çok eşgörünümü varsa, MQOO_BIND_ON_OPEN seçeneğini kullanın. MQOPEN çağrısından döndürülen nesne tanıtıcısını belirten kuyruğa konan tüm iletiler aynı kuyruk yöneticisine yönlendirilir.

- İletilerin yakınlıkları varsa MQOO_BIND_ON_OPEN seçeneğini kullanın. Örneğin, bir ileti kümesinin tümü aynı kuyruk yöneticisi tarafından işleniyorsa, kuyruğu açtığınızda MQOO_BIND_ON_OPEN değerini belirtin. IBM MQ , kuyruk yöneticisini ve o kuyruğa konan tüm iletiler tarafından alınacak rotayı düzeltir.
- MQOO_BIND_ON_OPEN seçeneği belirtilirse, kuyruğun yeni bir eşgörünümünün seçilmesi için kuyruğun yeniden açılması gerekir.

MQOO_BIND_NOT_FIXED
MQOPEN çağrısında MQOO_BIND_NOT_FIXED seçeneği, hedef kuyruk yöneticisinin düzeltilmediğini belirtir. MQOPEN çağrısından döndürülen nesne tanıtıcısını belirterek kuyruğa yazılan iletiler, ileti temelinde MQPUT saatinde bir kuyruk yöneticisine yöneltilir. Tüm iletilerinizi aynı hedefe yazmaya zorlamak istemiyorsanız MQOO_BIND_NOT_FIXED seçeneğini kullanın.

- Aynı anda MQOO_BIND_NOT_FIXED ve MQMF_SEGMENTATION_ALLOWED belirtmeyin. Bunu yaparsanız, iletinizin bölümleri kümeye dağılmış olarak farklı kuyruk yöneticilerine teslim edilebilir.

MQOO_BIND_ON_GROUP
Bir uygulamanın bir grup iletinin aynı hedef örneğe ayrılmasını istemesini sağlar. Bu seçenek yalnızca kuyruklar için geçerlidir ve yalnızca küme kuyruklarını etkiler. Küme kuyruğu olmayan bir kuyruk için belirtildiyse, seçenek yoksayılır.

- MQPUT üzerinde MQPMO_LOGICAL_ORDER belirtildiğinde gruplar tek bir hedefe yöneltilir. MQOO_BIND_ON_GROUP belirtildiyse, ancak bir ileti bir mantıksal grubun parçası değilse, bunun yerine BIND_NOT_FIXED davranışı kullanılır.

MQOO_BIND_AS_Q_DEF
MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED ya da MQOO_BIND_ON_GROUPbelirtmezseniz, varsayılan seçenek MQOO_BIND_AS_Q_DEFolur. MQOO_BIND_AS_Q_DEF değerinin kullanılması, kuyruk tanıtıcısı için kullanılan bağ tanımının DefBind kuyruk özniteliğinden alınmasına neden olur.

MQOPEN seçeneklerinin ilgi düzeyi

MQOPEN Seçenekler MQOO_BROWSE, MQOO_INPUT_*ya da MQOO_SET , MQOPEN ' in başarılı olması için küme kuyruğunun yerel bir eşgörünümünü gerektirir.

MQOPEN seçenekler MQOO_OUTPUT, MQOO_BIND_*ya da MQOO_INQUIRE , küme kuyruğunun yerel bir eşgörünümünün başarılı olmasını gerektirmez.

Çözülmüş kuyruk yöneticisi adı

MQOPEN sırasında bir kuyruk yöneticisi adı çözüldüğünde, çözülen ad uygulamaya döndürülür. Uygulama sonraki bir MQOPEN çağrısında bu adı kullanmayı denerse, ada erişim yetkisi olmadığını bulabilir.

MQPUT, MQPUT1 ve kümeler

MQOO_BIND_NOT_FIXED bir MQOPEN üzerinde belirtilirse, iş yükü yönetimi yordamları hangi hedefi MQPUT ya da MQPUT1 seçtiğini seçer.

Bir MQOPEN çağrısında MQOO_BIND_NOT_FIXED belirtilirse, sonraki her MQPUT çağrısı, iletinin hangi kuyruk yöneticisine gönderileceğini belirlemek için iş yükü yönetimi yordamını çağırır. Alınacak hedef ve rota, ileti temelinde seçilir. Ağ üzerindeki koşullar değişirse, ileti gönderildikten sonra hedef ve rota değişebilir. MQPUT1 çağrısı her zaman MQOO_BIND_NOT_FIXED etkinmiş gibi çalışır, yani her zaman iş yükü yönetimi yordamını çağırır.

İş yükü yönetimi yordamı bir kuyruk yöneticisi seçtiğinde, yerel kuyruk yöneticisi koyma işlemini tamamlar. İleti farklı kuyruklara yerleştirilebilir:

1. Hedef, kuyruğun yerel örneğiyse, ileti yerel kuyruğa yerleştirilir.
2. Hedef bir kümedeki kuyruk yöneticisiyse, ileti bir küme iletim kuyruğuna yerleştirilir.
3. Hedef bir kümenin dışındaki bir kuyruk yöneticisiyse, ileti, hedef kuyruk yöneticisiyle aynı adı taşıyan bir iletim kuyruğuna yerleştirilir.

MQOPEN çağrısında MQOO_BIND_ON_OPEN belirtilirse, hedef ve rota önceden seçildiğinden, MQPUT çağrıları iş yükü yönetimi yordamını çağırmaz.

MQINQ ve kümeler

Hangi küme kuyruğunun sorulduğu, MQOO_INQUIREile birleştirdiğiniz seçeneklere bağlıdır.

Bir kuyrukta sormadan önce, MQOPEN çağrıyı kullanarak açın ve MQOO_INQUIREseçeneğini belirleyin.

Bir küme kuyruğunda soru sormak için MQOPEN çağrıyı kullanın ve diğer seçenekleri MQOO_INQUIREile birleştirin. Sorulabilecek öznitelikler, küme kuyruğunun yerel bir örneğinin olup olmadığına ve kuyruğun nasıl açılacağına bağlıdır:

- MQOO_BROWSE, MQOO_INPUT_*ya da MQOO_SET ile MQOO_INQUIRE birleştirilirse, açmanın başarılı olması için küme kuyruğunun yerel bir eşgörünümü gerekir. Bu durumda, yerel kuyruklar için geçerli olan tüm öznitelikleri sorgulayabilirsiniz.
- MQOO_OUTPUT , MQOO_INQUIREile birleştiriliyor ve önceki seçeneklerin hiçbirini belirtmiyor; açılan örnek aşağıdakilerden biri olabilir:
  - Varsa, yerel kuyruk yöneticisindeki yönetim ortamı. Bu durumda, yerel kuyruklar için geçerli olan tüm öznitelikleri sorgulayabilirsiniz.
  - Yerel kuyruk yöneticisi yönetim ortamı yoksa, kümenin başka bir yerindeki bir yönetim ortamı. Bu durumda yalnızca aşağıdaki öznitelikler sorulacaktır. QType özniteliği bu durumda MQQT_CLUSTER değerine sahiptir.
    - DefBind
    - DefPersistence
    - DefPriority
    - InhibitPut
    - QDesc
    - QName
    - QType

Bir küme kuyruğunun DefBind özniteliğini sorgulamak için MQINQ seçiciyle birlikte çağrıyı MQIA_DEF_BINDkullanın. Döndürülen değer MQBND_BIND_ON_OPEN ya da MQBND_BIND_NOT_FIXEDya da MQBND_BIND_ON_GROUP. Kümeler içeren gruplar kullanılırken MQBND_BIND_ON_OPEN ya da MQBND_BIND_ON_GROUP belirtilmelidir.

Bir kuyruğun yerel yönetim ortamının CLUSTER ve CLUSNL özniteliklerini sorgulamak için MQINQ seçici ile çağır MQCA_CLUSTER_NAME ya da seçici MQCA_CLUSTER_NAMELISTseçeneğini kullanın.

Not: MQOPEN ' in bağlı olduğu kuyruğu düzeltmeden bir küme kuyruğunu açarsanız, ardışık MQINQ çağrıları küme kuyruğunun farklı eşgörünümlerini sorgulayabilir.

İlgili kavramlar

“Küme kuyruğu için MQOPEN seçeneği” sayfa 718
Kuyruk tanıtıcısı için kullanılan bağ tanımı, MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXEDya da MQBND_BIND_ON_GROUPdeğerini alabilen **DefBind** kuyruk özniteliğinden alınır.

MQSET ve kümeler

MQOPEN option MQOO_SET seçeneği, MQSET ' in başarılı olması için bir küme kuyruğunun yerel bir eşgörünümünün olmasını gerektirir.

Kümenin başka bir yerindeki bir kuyruğun özniteliklerini ayarlamak için MQSET çağrıyı kullanamazsınız.

Küme özniteliğiyle tanımlanmış bir yerel diğer adı ya da uzak kuyruğu açabilir ve MQSET çağrıyı kullanabilirsiniz. Yerel diğer adın ya da uzak kuyruğun özniteliklerini ayarlayabilirsiniz. Hedef kuyruğun farklı bir kuyruk yöneticisinde tanımlı bir küme kuyruğu olması önemli değildir.

Dönüş kodları

Kümelere özgü dönüş kodları

MQRC_CLUSTER_EXIT_ERROR (2266 X'8DA')

Bir küme kuyruğunu açmak ya da üzerine bir ileti koymak için MQOPEN, MQPUTya da MQPUT1 çağrısı yayınlanır. Bir kuyruk yöneticisinin ClusterWorkloadExit özniteliği tarafından tanımlanan küme iş yükü çıkışı beklenmedik bir şekilde başarısız olur ya da zamanında yanıt vermez.

Bu hatayla ilgili daha fazla bilgi veren IBM MQ for z/OS üzerindeki sistem günlüğüne bir ileti yazılır.

Bu kuyruk tanıtıcısı için sonraki MQOPEN, MQPUTve MQPUT1 çağrıları, ClusterWorkloadExit özniteliği boş gibi işlenir.

MQRC_CLUSTER_EXIT_LOAD_ERROR (2267 X'8DB')

z/OSüzerinde, küme iş yükü çıkışı yüklenemiyor.

Sistem günlüğüne bir ileti yazılır ve ClusterWorkloadExit özniteliği boş gibi işleme devam eder.

Multi Çoklu platformlarsistemlerinde, bir kuyruk yöneticisine bağlanmak için bir MQCONN ya da MQCONNX çağrısı yayınlanır. Kuyruk yöneticisinin kuyruk yöneticisi ClusterWorkloadExit özniteliği tarafından tanımlanan küme iş yükü çıkışı yüklenemediği için çağrı başarısız oldu.

MQRC_CLUSTER_PUT_INHIBITED (2268 X'8DC')

Bir küme kuyruğu için MQOO_OUTPUT ve MQOO_BIND_ON_OPEN seçeneklerini içeren bir MQOPEN çağrısı yayınlanır. InhibitPut özniteliği MQQA_PUT_INHIBITEDolarak ayarlanarak, kümedeki kuyruğun tüm eşgörünümleri şu anda engelleniyor. İleti almak için kullanılabilecek kuyruk eşgörünümü olmadığından, MQOPEN çağrısı başarısız olur.

Bu neden kodu, aşağıdaki deyimlerin her ikisi de doğru olduğunda oluşur:

- Kuyruğun yerel eşgörünümü yok. Yerel bir yönetim ortamı varsa, yerel yönetim ortamı engellenmiş olsa da MQOPEN çağrısı başarılı olur.
- Kuyruk için küme iş yükü çıkışı yok ya da bir küme iş yükü çıkışı var, ancak bir kuyruk eşgörünümü seçmiyor. (Küme iş yükü çıkışı bir kuyruk eşgörünümü seçerse, bu yönetim ortamı engellenmiş olsa da, MQOPEN çağrısı başarılı olur.)

MQOPEN çağrısında MQOO_BIND_NOT_FIXED seçeneği belirtilirse, kümedeki tüm kuyruklar engellenmiş olsa da çağrı başarılı olabilir. Ancak, arama sırasında tüm kuyruklar yine de engellenirse, sonraki bir MQPUT çağrısı başarısız olabilir.

MQRC_CLUSTER_RESOLUTION_ERROR (2189 X'88D')

1. Bir küme kuyruğunu açmak ya da üzerine bir ileti koymak için MQOPEN, MQPUTya da MQPUT1 çağrısı yayınlanır. Kuyruk tanımlaması doğru çözülemiyor; tam havuz kuyruğu yöneticisinden bir yanıt gerekiyor, ancak kullanılabilir bir yanıt yok.
2. PUBSCOPE (ALL) ya da SUBSCOPE (ALL) belirtilerek bir konu nesnesi için MQOPEN, MQPUT, MQPUT1 ya da MQSUB çağrısı yayınlanır. Tam havuz kuyruğu yöneticisinden bir yanıt gerektiği, ancak kullanılabilir bir yanıt olmadığı için küme konusu tanımlaması doğru şekilde çözülemiyor.

MQRC_CLUSTER_RESOURCE_ERROR (2269 X'8DD')

Bir küme kuyruğu için MQOPEN, MQPUTya da MQPUT1 çağrısı yayınlandı. Kümeleme için gerekli bir kaynağı kullanma girişimi sırasında hata oluştu.

MQRC_NO_DESTINATIONS_AVAILABLE (2270 X'8DE')

Küme kuyruğuna ileti koymak için bir MQPUT ya da MQPUT1 çağrısı yayınlanır. Çağrı sırasında, artık kümede kuyruğun herhangi bir eşgörünümü yoktur. MQPUT başarısız olur ve ileti gönderilmez.

Kuyruğu açan MQOPEN çağrısında MQOO_BIND_NOT_FIXED belirtilirse ya da iletiyi koymak için MQPUT1 kullanılırsa hata oluşabilir.

MQRC_STOPPED_BY_CLUSTER_EXIT (2188 X'88C')

Bir iletiyi küme kuyruğuna açmak ya da koymak için MQOPEN, MQPUTya da MQPUT1 çağrısı yayınlanır. Küme iş yükü çıkışı çağrıyı reddediyor.

z/OS IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

Bu bilgiler, desteklenen ortamların her birinde çalışan programların kullanabileceği IBM MQ olanaklarını açıklar. Ayrıca,

- IBM MQ-CICS bridgekullanma hakkında bilgi için bkz. CICS ile IBM MQ kullanma .
- IMS ve IMS köprüsünü kullanma hakkında bilgi için bkz. “IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları” sayfa 65.

IBM MQ for z/OSüzerinde uygulamaları kullanma ve yazma hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “Ortama bağımlı IBM MQ for z/OS işlevleri” sayfa 852
- “Hata ayıklama olanakları, eşitleme noktası desteği ve kurtarma desteği” sayfa 853
- “Uygulama ortamıyla birlikte IBM MQ for z/OS arabirimi” sayfa 854
- “z/OS UNIX System Services uygulamaları yazılıyor” sayfa 855
- “Paylaşılan kuyruklar içeren uygulama programlama” sayfa 859

İlgili kavramlar

“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

“Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

“Kuyruktaki iletilerin konması” sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

“Kuyruktan ileti alma” sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

“Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

“Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

“MQI ve kümelerle çalışma” sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

“IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları” sayfa 65
Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

z/OS *Ortama bağımlı IBM MQ for z/OS işlevleri*

IBM MQ for z/OS işlevlerini göz önünde bulundururken bu bilgileri kullanın.

IBM MQ for z/OS ' in çalıştığı ortamlarda IBM MQ işlevleri arasında dikkate alınacak temel farklılıklar şunlardır:

- IBM MQ for z/OS , aşağıdaki tetikleyici izleme programlarını sağlar:
  - CICS ortamında kullanım için CKTI

– IMS ortamında kullanılmak üzere CSQQTRMN

Diğer ortamlarda uygulamaları başlatmak için kendi modülünüzü yazmanız gerekir.

- İki aşamalı kesinleştirme kullanılarak eşitleme, CICS ve IMS ortamlarında desteklenir. Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri (RRS) kullanılarak z/OS toplu iş ortamında da desteklenir. Tek fazlı kesinleştirme, z/OS ortamında IBM MQ tarafından desteklenir.
- Toplu iş ve IMS ortamları için MQI, programların bir kuyruk yöneticisine bağlanması ve bu yöneticiyle bağlantılarının kesilmesi için çağrılar sağlar. Programlar birden çok kuyruk yöneticisine bağlanabilir.
- CICS sistemi tek bir kuyruk yöneticisine bağlanabilir. Altsistem adı CICS sistem başlatma işinde tanımlandıysa, CICS başlatıldığında bu durum oluşabilir. CICS ortamında MQI bağlantı ve bağlantı kesme çağrılarına izin verilse de bu çağrıların bir etkisi yoktur.
- API geçişi çıkışı, bir programın tüm MQI çağrılarının işlenmesine müdahale etmesini sağlar. Bu çıkış yalnızca CICS ortamında kullanılabilir.
- Çok işlemcili sistemlerde CICS ürününde, MQI çağrıları birden çok z/OS TCB altında yürütülebildiğinden bazı performans avantajları elde edilir. Daha fazla bilgi için bkz. z/OS üzerinde planlama *IBM MQ for z/OS Concepts and Planning Guide*.

Bu özellikler Çizelge 130 sayfa 853içinde özetlenmiştir.

Çizelge 130. z/OS ortam özellikleri

	CICS	**IMS**	**Toplu İş/TSO**
Tetikleyici izleyicisi sağlandı	Evet	Evet	Hayır
İki aşamalı kesinleştirme	Evet	Evet	Evet
Tek aşamalı kesinleştirme	Evet	Hayır	Evet
MQI çağrılarını bağla/bağlantısını kes	Tolere edilmiş	Evet	Evet
API-geçiş çıkışı	Evet	Hayır	Hayır

Not: İki aşamalı kesinleştirme, RRS kullanılarak Batch/TSO ortamında desteklenir.

z/OS *Hata ayıklama olanakları, eşitleme noktası desteği ve kurtarma desteği*

Program hata ayıklama olanakları, syncpoint desteği ve kurtarma desteği hakkında bilgi edinmek için bu bilgileri kullanın.

Program hata ayıklama olanakları

IBM MQ for z/OS , tüm ortamlardaki programlarda hata ayıklamak için kullanabileceğiniz bir izleme olanağı sağlar.

Ayrıca, CICS ortamında aşağıdakileri de kullanabilirsiniz:

- CICS Yürütme Tanılama Olanağı (CEDF)
- CICS İzleme Denetimi Hareketi (CETR)
- IBM MQ for z/OS API-geçiş çıkışı

z/OS platformunda, kullandığınız programlama dili tarafından desteklenen herhangi bir etkileşimli hata ayıklama aracını kullanabilirsiniz.

Syncpoint desteği

Hareket işleme ortamının güvenli bir şekilde kullanılabilmesi için, iş birimlerinin başlangıç ve bitiş zamanlarının uyumlulaştırılması gerekir.

Bu, CICS ve IMS ortamlarında IBM MQ for z/OS tarafından tam olarak desteklenir. Tam destek, kaynak yöneticileri arasında işbirliği anlamına gelir; böylece iş birimleri, CICS ya da IMS' in kontrolü altında aynı

anda ya da aynı anda kesinleştirilebilir ya da geri çekilebilir. Kaynak yöneticilerine örnek olarak Db2, CICS Dosya Denetimi, IMSve IBM MQ for z/OSverilebilir.

z/OS toplu iş uygulamaları, tek aşamalı kesinleştirme olanağı vermek için IBM MQ for z/OS çağrılarını kullanabilir. Başka bir deyişle, uygulama tanımlı bir kuyruk işlemleri kümesi, diğer kaynak yöneticilerine başvurmadan kesinleştirilebilir ya da geriletilebilir.

İki aşamalı kesinleştirme, hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri (RRS) kullanılarak z/OS toplu iş ortamında da desteklenir. Daha fazla bilgi için bkz. z/OS toplu uygulamalarda Syncpoints.

Kurtarma desteği

Bir hareket sırasında bir kuyruk yöneticisi ile bir CICS ya da IMS sistemi arasındaki bağlantı kesilirse, bazı iş birimleri başarıyla yedeklenmeyebilir.

Ancak bu iş birimleri, CICS ya da IMS sistemiyle yeniden bağlantı kurulduğunda kuyruk yöneticisi (syncpoint yöneticisinin denetimi altında) tarafından çözülür.

z/OS *Uygulama ortamıyla birlikte IBM MQ for z/OS arabirimi*

Farklı ortamlarda çalışan uygulamaların ileti kuyruklama ağı üzerinden ileti göndermelerine ve almalarına izin vermek için IBM MQ for z/OS , desteklediği ortamların her biri için bir *bağdaştırıcı* sağlar.

Bu bağdaştırıcılar, uygulama programları ile IBM MQ for z/OS altsistemleri arasındaki arabirimdir. Bu uygulamalar, programların MQI ' ı kullanmasına izin verir.

z/OS *Toplu iş bağdaştırıcısı*

Toplu iş bağdaştırıcısı ve desteklediği kesinleştirme protokolü hakkında bilgi edinmek için bu bilgileri kullanın.

Toplu iş bağdaştırıcısı , aşağıda belirtilenlerde çalışan programlar için IBM MQ for z/OS kaynaklarına erişim sağlar:

- Görev (TCB) kipi
- Sorun ya da gözetmen durumu
- Birincil adres alanı denetim kipi

Programlar, bellekler arası kipte olmamalıdır.

Uygulama programları ile IBM MQ for z/OS arasındaki bağlantılar görev düzeyindedir. Bağdaştırıcı, bir uygulama görevi denetim bloğundan (TCB) IBM MQ for z/OS' ye tek bir bağlantı iş parçacığı sağlar.

Bağdaştırıcı, IBM MQ for z/OS ' in sahip olduğu kaynaklarda yapılan değişiklikler için tek aşamalı bir kesinleştirme iletişim kuralını destekler; çoklu kesinleştirme protokollerini desteklemez.

z/OS *RRS toplu iş bağdaştırıcısı*

RRS toplu iş bağdaştırıcısı ve IBM MQtarafından sağlanan iki RRS toplu iş bağdaştırıcısı hakkında bilgi edinmek için bu bilgileri kullanın.

Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri (RRS) bağdaştırıcısı:

- Kesinleştirme denetimi için z/OS RRS kullanır.
- Tek bir görevden tek bir z/OS eşgörünümünde çalışan birden çok IBM MQ altsistemine eşzamanlı bağlantıları destekler.
- Aşağıdakiler için, z/OS RRS uyumlu kurtarılabilir yöneticiler aracılığıyla erişilen kurtarılabilir kaynaklar için z/OSçapında eşgüdümlü kesinleştirme denetimi (z/OS RRS kullanarak) sağlar:
  - RRS toplu iş bağdaştırıcısını kullanarak IBM MQ ' e bağlanan uygulamalar.
  - Db2-saklanmış yordamlar, z/OSüzerinde bir iş yükü yöneticisi (WLM) tarafından yönetilen Db2-saklanmış yordamları adres alanını belirtir.
- Bir IBM MQ toplu iş iş parçacığını TCB ' ler arasında değiştirme yeteneğini destekler.

IBM MQ for z/OS iki RRS toplu iş bağdaştırıcısı sağlar:

CSQBRSTB
Bu bağdaştırıcı, IBM MQ uygulamanızda herhangi bir MQCMIT deyimini SRRCMIT ve herhangi bir MQBACK deyimini SRRBACK olarak değiştirmenizi gerektirir. (MQCMIT ya da MQBACK ' i CSQBRSTB ile bağlantılı bir uygulamada kodluyorsanız, MQRC_ENVIRONMENT_ERROR alırsınız.)

CSQBRRSI
Bu bağdaştırıcı, IBM MQ uygulamanızın MQCMIT ve MQBACK ya da SRRCMIT ve SRRBACK kullanmasına olanak sağlar.

Not: CSQBRSTB ve CSQBRRSI, AMODE (31) RMODE (ANY) bağ öznitelikleriyle birlikte gönderilir. Uygulamanız 16 MB satırının altındaki herhangi bir sınırlı kod öbeğini yüklerse, önce sınırlı kod öbeğini RMODE (24) ile yeniden bağlayın.

Veri Taşıma

Var olan Batch/TSO IBM MQ uygulamalarını, RRS koordinasyonunu birkaç değişiklik olmadan ya da hiç değişiklik olmadan kullanmak üzere geçirebilirsiniz.

IBM MQ uygulamanızı CSQBRRSI bağdaştırıcısı, MQCMIT ve MQBACK ile IBM MQ ve RRS kullanabilen diğer tüm kaynak yöneticileriyle eşzamanlıyorsanız. IBM MQ uygulamanızı CSQBRSTB bağdaştırıcısıyla bağlayıp düzenlerseniz, MQCMIT ' i SRRCMIT ve MQBACK 'i SRRBACK olarak değiştirin. İkinci yaklaşım tercih edilir; bu, eşitleme noktasının yalnızca IBM MQ kaynaklarıyla sınırlı olmadığını açıkça gösterir.

z/OS *IMS bağdaştırıcısı*
IBM MQ for z/OS sisteminden IMS bağdaştırıcısını kullanıyorsanız, IMS 'in iletileri 100 MB' ye kadar sığdırmak için yeterli depolama alanı alabildiğinden emin olun.

Kullanıcılara not

IMS bağdaştırıcısı , aşağıdakiler için IBM MQ for z/OS kaynaklarına erişim sağlar:

- Çevrimiçi ileti işleme programları (MPPs)
- Etkileşimli hızlı yol programları (IFP)
- Toplu ileti işleme programları (BMP)

Bu kaynakları kullanabilmek için, programların görev (TCB) kipinde ve sorun durumunda çalışıyor olması gerekir; bunlar bellek arası kipte ya da erişim kaydı kipinde olmamalıdır.

Bağdaştırıcı, bir uygulama görevi denetim bloğundan (TCB) IBM MQ' ye bir bağlantı iş parçacığı sağlar. Bağdaştırıcı, IBM MQ for z/OS' in sahip olduğu kaynaklarda yapılan değişiklikler için iki aşamalı bir kesinleştirme protokolünü destekler; IMS , syncpoint eşgüdümcüsü görevi görür.

Bağdaştırıcı, kuyruktaki belirli tetikleyici koşulları karşılandığında programları otomatik olarak başlatabilen bir tetikleyici izleme programı da sağlar. Daha fazla bilgi için bkz “Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829.

Toplu DL/I programları yazıyorsanız, z/OS toplu iş programları için bu konuda verilen kılavuzu izleyin.

z/OS *z/OS UNIX System Services uygulamaları yazılıyor*

Toplu iş bağdaştırıcısı, toplu iş ve TSO adres alanlarından kuyruk yöneticisi bağlantılarını destekler.

Bir toplu adres alanı için bağdaştırıcı, adres alanı içindeki birden çok TCB ' den gelen bağlantıları aşağıdaki gibi destekler:

- Her TCB, MQCONN ya da MQCONNX çağrısıyla birden çok kuyruk yöneticisine bağlanabilir (ancak bir TCB ' nin belirli bir kuyruk yöneticisine aynı anda yalnızca bir bağlantı eşgörünümü olabilir).
- Birden çok TCB aynı kuyruk yöneticisine bağlanabilir (ancak, herhangi bir MQCONN ya da MQCONNX çağrısında döndürülen kuyruk yöneticisi tanıtıcısı, yayınlayan TCB ' ye bağlıdır ve başka bir TCB tarafından kullanılamaz).

z/OS UNIX System Services , iki tip pthread_create çağrıyı destekler:

1. z/OSile başlayan ve biten iş parçacıklarında ATTACHed ve DETACHed olan her TCB için ağır sıklet iş parçacıkları çalıştırın.
2. Orta ağırlıklı iş parçacıkları, her TCB için bir tane çalıştırın, ancak TCB, uzun süreli TCB havuzlarından biri olabilir. Uygulama, bir sunucuya bağlıysa, sunucu tarafından görev (TCB) sonlandırması sırasında sağlanabilecek varsayılan iş parçacığı sonlandırması her zaman **yönlendirilmediğinden** , gerekli tüm uygulama temizlemesini gerçekleştirmelidir.

Basit iş parçacıkları desteklenmez. (Bir uygulama kendi iş isteklerini dağıtan kalıcı iş parçacıkları oluşturursa, ***uygulaması*** , sonraki iş isteğini başlatmadan önce tüm kaynakları temizlemekten sorumludur.)

IBM MQ for z/OS , aşağıda gösterildiği gibi Toplu İş Bağdaştırıcısı kullanan z/OS UNIX System Services iş parçacıklarını destekler:

1. Ağır sıklet iş parçacıkları toplu bağlantılar olarak tam olarak desteklenir. Her iş parçacığı, iş parçacığı başlangıcında ve sonunda eklenen ve ayrılan kendi TCB ' de çalışır. İş parçacığı bir MQDISC çağrısı yayınlamadan önce sona ererse, IBM MQ for z/OS standart görev temizleme işlemini gerçekleştirir; bu, iş parçacığı olağan şekilde sonlandıysa, bekleyen iş birimlerini kesinleştirmeyi ya da iş parçacığı olağandışı sonlandıysa, iş parçacığını geri yedeklemeyi içerir.
2. Orta ağırlıklı iş parçacıkları tam olarak desteklenir, ancak TCB başka bir iş parçacığı tarafından yeniden kullanılacaksa, uygulama sonraki iş parçacığı başlamadan önce MQCMIT ya da MQBACK ile başlayan bir MQDISC çağrısına dikkat edilmelidir. Bu, uygulama bir Program Kesme İşleyicisi oluşturduysa ve uygulama olağandışı sonlandıysa, Interrupt Handler 'ın TCB ' yi başka bir iş parçacığı için yeniden kullanmadan önce MQCMIT ve MQDISC çağrıları yayınlaması gerektiği anlamına gelir.

Not: İş parçacığı oluşturma modelleri, birden çok iş parçacığından ortak IBM MQ kaynaklarına erişimi **desteklemez** .

z/OS *z/OS için API geçiş çıkışı*

Bu konu, ürüne duyarlı programlama arabirimi bilgilerini içerir.

Çıkış, IBMtarafından sağlanan kodda kendi kodunuzu çalıştırabileceğiniz bir noktadır. IBM MQ for z/OS , MQI çağrılarını engellemek ve MQI çağrılarının işlevini izlemek ya da değiştirmek için kullanabileceğiniz bir *API geçiş çıkışı* sağlar. Bu bölümde, API geçiş çıkışının nasıl kullanılacağı ve IBM MQ for z/OSile birlikte verilen örnek çıkış programı açıklanmaktadır.

Bu bölüm yalnızca CICS TS V3.1 ve önceki sürümlerin kullanıcıları için geçerlidir. CICS TS V3.2 ve sonraki sürümlerin kullanıcıları, CICS ürün belgelerinde CICS Integration with IBM MQ (Bütünleştirme) bölümüne bakmalıdır.

Not

API geçiş çıkışı yalnızca IBM MQ for z/OS CICS bağdaştırıcısı tarafından çağrılır. Çıkış programı CICS adres alanında çalışır.

z/OS *Kendi çıkış programınızı yazma*

IBM MQ for z/OS ile birlikte sağlanan örnek API geçiş çıkış programını (CSQCAPX) kendi programınız için bir çerçeve olarak kullanabilirsiniz.

Bu, “Örnek API-geçiş çıkış programı, CSQCAPX” sayfa 857içinde açıklanmıştır.

Bir çıkış programı yazarken, bir uygulama tarafından verilen bir MQI çağrısının adını bulmak için MQXP yapısının *ExitCommand* alanını inceleyin. Çağrıdaki parametre sayısını bulmak için *ExitParmCount* alanını inceleyin. Uygulamanın edindiği herhangi bir dinamik saklama alanının adresini saklamak için 16 baytlık *ExitUserArea* alanını kullanabilirsiniz. Bu alan, çıkışın çağrılmaları boyunca alıkonur ve bir CICS göreviyle aynı yaşam süresine sahiptir.

CICS Transaction Server V3.2kullanıyorsanız, çıkış programınızı iş parçacığı korumalı olacak şekilde yazmanız ve çıkış programınızı iş parçacığı korumalı olarak bildirmeniz gerekir. Daha önceki CICS

yayınlarını kullanıyorsanız, çıkış programlarınızı CICS Transaction Server V3.2' ye geçiş için hazır olması için iş parçacığı korumalı olarak yazmanız ve bildirmeniz de önerilir.

Çıkış programınız, *ExitResponse* alanında MQXCC_SUPPRESS_FUNCTION ya da MQXCC_SKIP_FUNCTION döndürerek bir MQI çağrısının yürütülmesini engelleyebilir. Çağrım yürütülmesine (ve çağrı tamamlandıktan sonra çıkış programının yeniden çağrılmasına) izin vermek için çıkış programınız MQXCC_OK döndürmelidir.

Bir MQI çağrıldıktan sonra çağrıldığında, bir çıkış programı çağrı tarafından ayarlanan tamamlanma ve neden kodlarını inceleyebilir ve değiştirebilir.

Kullanım notları

Çıkış programınızı yazarken dikkate alınması gereken bazı genel noktalar şunlardır:

- Performans nedenleriyle, programınızı çevirici dilde yazın. Bunu IBM MQ for z/OStarafından desteklenen diğer dillerden herhangi birine yazarsanız, kendi veri tanımlama dosyanızı sağlamanız gerekir.
- Programınızı AMODE (31) ve RMODE (ANY) olarak düzenleyin.
- Programınıza çıkış parametresi öbeğini tanımlamak için çevirici dili makrosunu (CMQXPA) kullanın.
- Çıkış programınızı ve çıkış programınızın çağırdığı programları tanımlarken CONCURRENCY (THREADSAFE) değerini belirleyin.
- CICS Transaction Server for z/OS depolama koruma özelliğini kullanıyorsanız, programınızın CICS yürütme anahtarında çalışması gerekir. Yani, EXECKEY (CICS) belirlemelisiniz Hem çıkış programınızı, hem de denetimi aktardığı programları tanımlarken. CICS çıkış programları ve CICS depolama koruma olanağı hakkında bilgi için bkz. *CICS Özelleştirme Kılavuzu*.
- Programınız tüm API ' leri kullanabilir (örneğin, IMS, Db2ve CICS) CICS göreviyle ilgili bir kullanıcı çıkış programının kullanabileceği. MQCONN, MQCONNX ve MQDISC dışında MQI çağrılarını da kullanabilir. Ancak, çıkış programı içindeki MQI çağrıları çıkış programını ikinci kez çağırmaz.
- Programınız EXEC CICS SYNCPOINT ya da EXEC CICS SYNCPOINT ROLLBACK komutlarını verebilir. Ancak bu komutlar, görev tarafından yapılan güncellemeleri, çıkışın kullanıldığı noktaya kadar kesinleştirme ya da geri alma ***tümünü*** ve bu nedenle bunların kullanılması önerilmez.
- Programınız bir EXEC CICS RETURN komutu vererek sona ermelidir. Denetimi bir XCTL komutuyla aktarmamalıdır.
- Çıkışlar, IBM MQ for z/OS kodunun uzantıları olarak yazılır. Çıkışınızın MQI kullanan herhangi bir IBM MQ for z/OS programını ya da hareketini kesintiye uğratmadığından emin olun. Bunlar genellikle CSQ ya da CK önekiyle gösterilir.
- CSQCAPX, CICSolarak tanımlanırsa, CICS sistemi, CICS IBM MQ for z/OSile bağlantı kurduğunda çıkış programını yüklemeye çalışır. Bu girişim başarılı olursa, CKQC panosuna ya da sistem konsoluna CSQC301I iletisi gönderilir. Yükleme başarısız olursa (örneğin, yükleme modülü DFHRPL birleştirmesindeki kitaplıkların hiçbirinde yoksa), CKQC panosuna ya da sistem konsoluna CSQC315 iletisi gönderilir.
- İletişim alanındaki parametreler adres olduğundan, çıkış programı CICS sistemi için yerel olarak tanımlanmalıdır (uzak program olarak değil).

z/OS *Örnek API-geçiş çıkış programı, CSQCAPX*

Örnek çıkış programı çevirici dil programı olarak sağlanır. Kaynak dosya (CSQCAPX), **thlqual**.SCSQASMS (burada **thlqual** , kuruluşunuz tarafından kullanılan üst düzey niteleyicidir) kitaplığında sağlanır. Bu kaynak dosya, program mantığını açıklayan sözde kodu içerir.

Örnek program, başlatma kodunu ve kendi çıkış programlarınızı yazarken kullanabileceğiniz bir yerleşim düzenini içerir.

Örnek, aşağıdakileri nasıl yapılacağını gösterir:

- Çıkış değiştirgesi öbeğinin ayarlanması
- Çağrı ve çıkış parametresi bloklarını ele al

- Çıkışın çağrıldığı MQI çağrıyı sapta
- Çıkışın MQI çağrısından önce mi, yoksa sonra mı çağrıldığını saptayın
- İletiyi CICS geçici depolama kuyruğuna koyma
- Yeniden girişin sürdürülmesi için dinamik depolama edinimi için DFHEIENT makrosunu kullanın
- CICS exec arabirim denetim bloğu için DFHEIBLK kullan
- Tuzak hatası koşulları
- Denetimi çağırana geri döndür

Örnek çıkış programının tasarımı

Örnek çıkış programı, çıkış işlemini göstermek için iletileri CICS geçici depolama kuyruğuna (CSQ1EXIT) yazar.

İletiler, çıkışın MQI çağrısından önce mi, yoksa sonra mı çağrıldığını gösterir. Çıkış çağrıldıktan sonra çağrılırsa, ileti, çağrı tarafından döndürülen tamamlanma kodunu ve neden kodunu içerir. Örnek, giriş tipini (çağrıdan önce ya da sonra) denetlemek için CMQXPA makrosunda belirtilen değişmezleri kullanır.

Örnek herhangi bir izleme işlevi gerçekleştirmez, ancak zaman damgalı iletileri işlediği çağrı tipini gösteren bir CICS kuyruğuna yerleştirir. Bu, çıkış programının doğru çalışmasının yanı sıra MQI ' ın başarımının da göstergesidir.

Not: Örnek çıkış programı, program çalışırken yapılan her MQI çağrısı için altı EXEC CICS çağrısı yayınlar. Bu çıkış programını kullanırsanız, IBM MQ for z/OS performansı düşer.

z/OS *API geçiş çıkışının hazırlanması ve kullanılması*

Örnek çıkış yalnızca kaynak biçimde sağlanır.

To use the sample exit, or an exit program that you have written, create a load library, as you would for any other CICS program, as described in “z/OS içinde CICS uygulamaları oluşturma” sayfa 983.

- CICS Transaction Server for z/OS ve CICS for MVS/ESAiçin, CICS sistem tanımlaması (CSD) veri kümesini güncellediğinizde, gerek duyduğunuz tanımlamalar **thlqual**.SCSQPROC (CSQ4B100) üyesinde bulunur.

  Not: Tanımlamalar MQsonekini kullanır. Bu sonek kuruluşunuzda zaten kullanılıyorsa, bu, montaj aşamasından önce değiştirilmelidir.

Sağlanan varsayılan CICS program tanımlarını kullanırsanız, çıkış programı CSQCAPX, ***devre dışı*** durumunda kurulur. Bunun nedeni, çıkış programının kullanılması başarımı önemli ölçüde azaltabilir.

API geçiş çıkışını geçici olarak etkinleştirmek için:

1. **CEMT S PROGRAM(CSQCAPX) ENABLED** komutunu CICS ana uçbiriminden verin.
2. CKQC hareketini çalıştırın ve API geçiş çıkışının durumunu **Etkin**olarak değiştirmek için Bağlantı çekme işleminde seçenek 3 'ü kullanın.

IBM MQ for z/OS ' i API-geçiş çıkışı kalıcı olarak etkinleştirilmiş olarak, CICS Transaction Server for z/OS ve CICS for MVS/ESAile çalıştırmak istiyorsanız aşağıdakilerden birini yapın:

- CSQ4B100üyesindeki CSQCAPX tanımlamasını değiştirerek, STATUS (DISABLED) değerini STATUS (ENABLED) olarak değiştirin. CICS CSD tanımını, CICStarafından sağlanan DFHCSDUP toplu iş programını kullanarak güncelleyebilirsiniz.
- Durumu DISABLED olarak ENABLED olarak değiştirerek CSQCAT1 grubundaki CSQCAPX tanımlamasını değiştirin.

Her iki durumda da grubu yeniden kurmanız gerekir. Bunu CICS sisteminizi soğuk başlatarak ya da CICS çalışırken grubu yeniden kurmak için CICS CEDA hareketini kullanarak yapabilirsiniz.

Not: Gruptaki girişlerden herhangi biri kullanımdaysa, CEDA ' nın kullanılması hataya neden olabilir.

Ürüne duyarlı programlama arabirimi bilgilerinin sonu.

z/OS *Paylaşılan kuyruklar içeren uygulama programlama*

Bu konuda, paylaşılan kuyrukları kullanmak üzere yeni uygulamalar tasarlarken ve var olan uygulamaları paylaşılan kuyruk ortamına geçirirken dikkate almanız gereken bazı etkenler hakkında bilgi sağlanır.

z/OS *Uygulamalarınızı diziselleştirme*

Bazı uygulama tipleri, iletilerin kuyruktan tam olarak kuyruğa geldikleri sırada alındığından emin olmak zorunda olabilir.

Örneğin, IBM MQ veritabanı güncellemelerini uzak bir sisteme gölgelemek için kullanılıyorsa, o kaydın eklenmesini açıklayan bir iletiden sonra bir kayıt güncellemesini açıklayan bir ileti işlenmelidir. Yerel bir kuyruğa alma ortamında, bu genellikle MQOO_INPUT_EXCLUSIVE seçeneğiyle kuyruğu açan iletileri alan uygulama tarafından elde edilir ve böylece, başka bir uygulamanın kuyruğu aynı anda işlemesini önler.

IBM MQ , uygulamaların paylaşılan kuyrukları yalnızca aynı şekilde açmalarını sağlar. Ancak, uygulama bir kuyruğun bölümünden çalışıyorsa (örneğin, tüm veritabanı güncellemeleri aynı kuyruktaysa, ancak A çizelgesine ilişkin ilinti tanıtıcısı A ise, B çizelgesine ilişkin ilinti tanıtıcısı B ise) ve uygulamalar A çizelgesine ilişkin iletileri ve B çizelgesine ilişkin güncellemeleri eşzamanlı olarak almak istiyorlarsa, kuyruğun yalnızca açılmasına ilişkin basit bir mekanizma mümkün değildir.

Bu tip bir uygulama, paylaşılan kuyrukların yüksek kullanılabilirliğinden yararlanacaksa, birincil alma uygulaması ya da kuyruk yöneticisi başarısız olursa, aynı paylaşılan kuyruklara erişen uygulamanın ikincil bir kuyruk yöneticisinde çalışan başka bir eşgörünümünün devralması gerektiğine karar verebilirsiniz.

Birincil kuyruk yöneticisi başarısız olursa, iki durum gerçekleşir:

- Paylaşılan kuyruk eşdüzey kurtarma, birincil uygulamadaki eksik güncellemelerin tamamlanmasını ya da geriletilmesini sağlar.
- İkincil uygulama, kuyruğun işlenmesini devralır.

İkincil uygulama, tüm tamamlanmamış iş birimleri ele alınmadan önce başlatılabilir; bu, ikincil uygulamanın iletileri sırasız olarak almasına neden olabilir. Bu tip bir sorunu çözmek için uygulama *diziselleştirilmiş uygulama*olmayı seçebilir.

Diziselleştirilmiş bir uygulama, kuyruk yöneticisine bağlanmak için MQCONNX çağrıyı kullanarak, o uygulama için benzersiz olan bir bağlantı etiketini belirtir. Uygulama tarafından gerçekleştirilen tüm iş birimleri bağlantı etiketiyle işaretlenir. IBM MQ , aynı bağlantı etiketine sahip kuyruk paylaşım grubu içindeki iş birimlerinin diziselleştirilmesini sağlar (MQCONNX çağrısındaki diziselleştirme seçeneklerine göre).

Başka bir deyişle, birincil uygulama MQCONNX çağrıyı `Database shadow retriever`bağlantı etiketiyle kullanıyorsa ve ikincil devralma uygulaması MQCONNX çağrısının aynı bağlantı etiketiyle kullanılmaya çalışıyorsa, ikincil uygulama, bu durumda eşdüzey kurtarma işlemi tamamlanıncaya kadar ikinci IBM MQ ' e bağlanamaz.

Kuyruktaki iletilerin tam sırasına bağlı olan uygulamalar için diziselleştirilmiş uygulama tekniğini kullanmayı düşünün. Özellikle:

- Uygulamanın önceki yürütülmesine ilişkin tüm kesinleştirme ve geriletme işlemleri tamamlanıncaya kadar, bir uygulama ya da kuyruk yöneticisi hatasından sonra yeniden başlatılmaması gereken uygulamalar.

  Bu durumda, diziselleştirilmiş uygulama tekniği yalnızca uygulama uyumluluk noktasında çalışıyorsa geçerlidir.
- Aynı uygulamanın başka bir örneği çalışırken başlatılmaması gereken uygulamalar.

  Bu durumda, diziselleştirilmiş uygulama tekniği yalnızca uygulama kuyruğu dışlayıcı giriş için açamazsa gereklidir.

Not: IBM MQ yalnızca belirli ölçütler karşılandığında iletilerin sırasını korumayı garanti eder. Bunlar MQGETtanımında açıklanmıştır.

z/OS *Paylaşılan kuyruklarla kullanılmaya uygun olmayan uygulamalar*

Paylaşılan kuyrukları kullanırken IBM MQ ' un bazı özellikleri desteklenmez; bu nedenle, bu özellikleri kullanan uygulamalar paylaşılan kuyruk ortamı için uygun değildir.

Paylaşılan kuyruk uygulamalarınızı tasarlarken aşağıdaki noktaları göz önünde bulundurun:

- Kuyruk dizinleme, paylaşılan kuyruklar için sınırlıdır. Kuyruktan almak istediğiniz iletiyi seçmek için ileti tanıtıcısını ya da ilinti tanıtıcısını kullanmak istiyorsanız, kuyruk doğru değerle dizinlenmelidir. İletileri yalnızca ileti tanıtıcısına göre seçecekseniz, kuyruk için MQIT_MSG_ID dizin tipi gerekir (MQIT_NONE da kullanılabilir). İletileri yalnızca ilinti tanıtıcısına göre seçiyorsa, kuyruğun dizin tipi MQIT_CORREL_ID olmalıdır.
- Paylaşılan kuyruklar olarak geçici dinamik kuyrukları kullanamazsınız. Ancak, kalıcı dinamik kuyruklar kullanabilirsiniz. Paylaşılan dinamik kuyruklar için modeller, PERMDYN (kalıcı dinamik) kuyruklarıyla aynı şekilde yaratılıp yok edilmelerine rağmen, DEFTYPE ' de SHAREDYN (paylaşılan dinamik) vardır.

z/OS *Uygulama dışı kuyrukların paylaşılıp paylaşılmayacağına karar verilmesi*

Uygulama dışı kuyrukları paylaşmayı düşünürken bu bilgileri kullanın.

Paylaşmak isteyebileceğiniz uygulama kuyrukları dışında kuyruklar var:

Başlatma kuyrukları

Paylaşılan başlatma kuyruğu tanımlarsanız, en az bir tetikleyici izleme programı çalıştığı sürece, kuyruk paylaşım grubundaki her kuyruk yöneticisinde çalışan bir tetikleyici izleme programına gerek yoktur. (Kuyruk paylaşım grubundaki her kuyruk yöneticisinde çalışan bir tetikleyici izleme programı olsa bile, paylaşılan başlatma kuyruğunu da kullanabilirsiniz.)

Paylaşılan bir uygulama kuyruğunuz varsa ve EVERY tetikleyici tipini (ya da EVERY tetikleyici tipi gibi davranan, küçük bir tetikleyici aralığı olan bir FIRST tetikleyici tipini) kullanırsanız, başlatma kuyruğunuz her zaman paylaşılan bir kuyruk olmalıdır. Paylaşılan başlatma kuyruğunun ne zaman kullanılacağı hakkında daha fazla bilgi için bkz. Çizelge 131 sayfa 861.

SISTEM.* Kuyruklar

SYSTEM.ADMIN.* Olay iletilerini paylaşılan kuyruklar olarak tutmak için kullanılan kuyruklar. Bu, bir kural dışı durum oluşursa yük dengelemeyi denetlemek için yararlı olabilir. IBM MQ tarafından yaratılan her olay iletisi, bunu hangi kuyruk yöneticisinin ürettiğini gösteren bir ilinti tanıtıcısı içerir.

SYSTEM.QSG.* paylaşılan kanallar için kullanılan kuyruklar ve paylaşılan kuyruklar olarak grup içi kuyruğa alma.

SYSTEM.DEFAULT.LOCAL.QUEUE KUYRUğU ya da kendi varsayılan paylaşılan kuyruk tanımlamanızı tanımlayın. Ek bilgi için IBM MQ for z/OSiçin sistem nesnelerini tanımlama başlıklı konuya bakın.

Başka bir SYSTEM.* tanımlayamazsınız kuyruklar, paylaşılan kuyruklar olarak.

z/OS *Var olan uygulamalarınızı paylaşılan kuyrukları kullanacak şekilde geçirme*

Neden kodları, tetikleme ve MQINQ API çağrısı, paylaşılan bir kuyruk ortamında farklı şekilde çalışabilir.

Varolan kuyruklarınızın paylaşılan kuyruklara geçirilmesine ilişkin bilgi için Paylaşılan olmayan kuyrukların paylaşılan kuyruklara geçirilmesi başlıklı konuya bakın.

Var olan uygulamalarınızı yeni düzeye geçirirken, paylaşılan kuyruk ortamında farklı bir şekilde çalışabilecek aşağıdaki özellikleri göz önünde bulundurun:

Neden kodları

Var olan uygulamalarınızı paylaşılan kuyrukları kullanacak şekilde geçirirken, yayınlanabilecek yeni neden kodlarını denetleyin.

Tetikleme

Paylaşılan bir uygulama kuyruğu kullanıyorsanız, tetikleme yalnızca kesinleştirilmiş iletilerde çalışır (paylaşılmayan bir uygulama kuyruğunda, tetikleme tüm iletilerde çalışır).

Uygulamaları başlatmak için tetikleme özelliğini kullanırsanız, paylaşılan bir başlatma kuyruğu kullanmak isteyebilirsiniz. Çizelge 131 sayfa 861 içinde, hangi başlatma kuyruğu tipinin kullanılacağına karar verirken göz önünde bulundurmanız gereken noktalar açıklanır.

Çizelge 131. Paylaşılan başlatma kuyruğunun ne zaman kullanılacağı

	Paylaşılmayan uygulama kuyruğu	**Paylaşılan uygulama kuyruğu**
Paylaşılmayan başlatma kuyruğu	Önceki yayınlara gelince.	FIRST ya da DEPTH tetikleyici tipi kullanırsanız, paylaşılan uygulama kuyruğuyla paylaşılmayan bir başlatma kuyruğu kullanabilirsiniz. Fazladan tetikleyici iletileri oluşturulabilir, ancak bu ayar uzun süreli uygulamaları (CICS bridgegibi) tetiklemek için iyidir ve yüksek kullanılabilirlik sağlar. Tetikleyici tipi FIRST ya da DEPTH için, tetikleyici iletisi, tetikleyici izleme programı çalıştıran ve giriş için uygulama kuyruğu açık olmayan her kuyruk yöneticisinde uygulamanın bir eşgörünümünü tetikler. Her kuyruk yöneticisi için bir tetikleyici iletisi üretilir; paylaşılmayan yerel başlatma kuyruğuna karşı çalışan birden fazla tetikleyici varsa, belirli bir kuyruk yöneticisinde bu tetikleyici, iletiyi işlemek için rekabet eder.
Paylaşılan başlatma kuyruğu	Paylaşılmayan bir uygulama kuyruğuyla paylaşılan başlatma kuyruğu kullanmayın.	EVERY tetikleyici tipi için, bir uygulama paylaşılan uygulama kuyruğuna bir ileti yerleştirdiğinde, kuyruk yöneticisi hangi kuyruk yöneticilerinin tetikleyiciyle ilgilendiğini belirler-her olay ve bu kuyruk yöneticilerinden birine bildirim gönderir. Bildirilen kuyruk yöneticisinde, sonuçta ortaya çıkan işlem, başlatma kuyruğuna bir tetikleyici iletisi oluşturulmasıdır. **Not:** Tetikleyici tipi EVERY olan bir paylaşılan uygulama kuyruğunuz varsa, paylaşılan bir başlatma kuyruğu kullanın ya da belirli durumlarda tetikleyici iletilerini kaybedebilirsiniz; örneğin, bir kuyruk yöneticisi başarısız olabilir. FIRST ya da DEPTH tetikleyici tipi için, adı belirtilen başlatma kuyruğu giriş için açık olan her kuyruk yöneticisi tarafından bir tetikleyici iletisi üretilir. **Not:** Tetikleyici tipi FIRST ya da DEPTH için, bir tetikleyici izleyici eşgörünümü meşgulse, bu, daha az meşgul tetikleyici izleme programlarının paylaşılan başlatma kuyruğundan birden çok tetikleyici iletisini işlemesi olasılığını bırakır. Bu nedenle, belirli bir kuyruk yöneticisi için sunucu uygulamasının birden çok eşgörünümü başlatılabilir. Birden çok tetikleyici iletilerin işlenmesi sonucunda bu birden çok yönetim ortamının başlatıldığını unutmayın. Genellikle, FIRST ya da DEPTH tetikleyici tipi için, bir uygulama eşgörünümü zaten bir uygulama kuyruğuna hizmet veriyorsa, uygulamanın bağlı olduğu kuyruk yöneticisi tarafından başka bir tetikleyici iletisi oluşturulmaz.

MQINQ

Paylaşılan bir kuyruğa ilişkin bilgileri görüntülemek için MQINQ çağrısını kullandığınızda, kuyruğun giriş ve çıkış için açık olduğu MQOPEN çağrılarının değerleri yalnızca, çağrıyı yayınlayan kuyruk yöneticisiyle ilişkilidir. Kuyruğu açık olan kuyruk paylaşım grubundaki diğer kuyruk yöneticileri hakkında bilgi üretilmez.

z/OS IBM MQ for z/OS üzerinde IMS ve IMS köprü uygulamaları

Bu bilgiler, IBM MQkullanarak IMS uygulamaları yazmanıza yardımcı olur.

- IMS uygulamalarında eşitleme noktalarını ve MQI çağrılarını kullanmak için bkz. “IMS uygulamalarının IBM MQ kullanılarak yazılması” sayfa 66.
- IBM MQ - IMS köprüsünü kullanan uygulamalar yazmak için bkz. “IMS köprü uygulamalarının yazılması” sayfa 70.

IBM MQ for z/OSüzerinde IMS ve IMS köprü uygulamaları hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “IMS uygulamalarının IBM MQ kullanılarak yazılması” sayfa 66
- “IMS köprü uygulamalarının yazılması” sayfa 70

İlgili kavramlar

“İleti Kuyruğu Arabirimine Genel Bakış” sayfa 692
İleti Kuyruğu Arabirimi (MQI) bileşenleri hakkında bilgi edinin.

“Kuyruk yöneticisine bağlanma ve bağlantı kesme” sayfa 705
IBM MQ programlama hizmetlerini kullanabilmek için, bir programın kuyruk yöneticisiyle bağlantısı olmalıdır. Bir kuyruk yöneticisine nasıl bağlanılacağını ve bağlantı kesileceğini öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 712
Bu bilgiler, IBM MQ nesnelerini açmaya ve kapatmaya ilişkin bir öngörü sağlar.

“Kuyruktaki iletilerin konması” sayfa 722
İletilerin kuyruğa nasıl yerleştirileceğini öğrenmek için bu bilgileri kullanın.

“Kuyruktan ileti alma” sayfa 736
Kuyruktan ileti almaya ilişkin bilgi edinmek için bu bilgileri kullanın.

“Nesne öznitelikleri hakkında sorma ve bunları ayarlama” sayfa 815
Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“İş birimlerinin kesinleştirilmesi ve geri çekilmesi” sayfa 818
Bu bilgiler, bir iş biriminde gerçekleştirilen kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceğini ve geri çekileceğini açıklar.

“Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829
Tetikleyiciler ve tetikleyiciler kullanarak IBM MQ uygulamalarının nasıl başlatılacağını öğrenin.

“MQI ve kümelerle çalışma” sayfa 847
Çağrılarda ve dönüş kodlarında kümeleme ile ilgili özel seçenekler vardır.

“IBM MQ for z/OS üzerinde uygulamaları kullanma ve yazma” sayfa 852
IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan yapılabilir. Bu, birden fazla ortamda bulunan tesislerden yararlanabilecekleri anlamına gelir.

z/OS *IMS uygulamalarının IBM MQ kullanılarak yazılması*

IMS uygulamalarında IBM MQ kullanılırken hangi MQ API çağrılarının kullanılabileceğini ve uyumluluk noktası için kullanılan mekanizmayı içeren başka noktalar da vardır.

IBM MQ for z/OSüzerinde IMS uygulamaları yazma hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “IMS uygulamalarında eşitleme noktaları” sayfa 67

- “IMS uygulamalarında MQI çağrıları” sayfa 67

Sınırlamalar

IMS bağdaştırıcısını kullanan bir uygulama tarafından kullanılan IBM MQ API çağrılarına ilişkin kısıtlamalar vardır.

Aşağıdaki IBM MQ API çağrıları, IMS bağdaştırıcısını kullanan bir uygulamada desteklenmez:

- MQCB
- MQCB_FUNCTION
- MQCTL

İlgili kavramlar

“IMS köprü uygulamalarının yazılması” sayfa 70
Bu konuda, IBM MQ - IMS köprüsünü kullanmak üzere uygulamalar yazılmasına ilişkin bilgiler yer alır.

z/OS *IMS uygulamalarında eşitleme noktaları*

Bir IMS uygulamasında, IOPCB ve CHKP (denetim noktası) için GU (benzersiz olsun) gibi IMS çağrılarını kullanarak bir eşitleme noktası oluşturabilirsiniz.

Önceki denetim noktasından bu yana yapılan tüm değişiklikleri geri almak için IMS ROLB (rollback) çağrıyı kullanabilirsiniz. Daha fazla bilgi için IMS belgelerinde ROLB call başlıklı konuya bakın.

Kuyruk yöneticisi, iki aşamalı bir kesinleştirme protokolünün katılımcısıdır; eşgüdümcü IMS syncpoint yöneticisidir.

Tüm açık tutamaçlar, IMS bağdaştırıcısı tarafından bir eşitleme noktasında (toplu ya da ileti odaklı olmayan BMP ortamı dışında) kapatılır. Bunun nedeni, MQPUT ya da MQGET çağrıları yapıldığında değil, MQCONN, MQCONNX ve MQOPEN çağrıları yapıldığında farklı bir kullanıcı sonraki iş birimini başlatabilir ve IBM MQ güvenlik denetimi gerçekleştirilir.

Ancak, bir WFI (Wait-for-Input) ya da pseudo Wait-for-Input (PWFI) ortamında, bir sonraki ileti gelinceye ya da uygulamaya bir QC durum kodu dönünceye kadar tanıtıcıları kapatmaları IMS bildirilmez IBM MQ . Uygulama IMS bölgesinde bekliyorsa ve bu tanıtıcılardan herhangi biri tetiklenen kuyruklara aitse, kuyruklar açık olduğu için tetikleme gerçekleşmez. Bu nedenle, bir WFI ya da PWFI ortamında çalışan uygulamaların, sonraki ileti için GU 'yu IOPCB' ye yapmadan önce kuyruk tanıtıcılarını belirtik olarak MQCLOSE olarak belirtmeleri gerekir.

Bir IMS uygulaması (BMP ya da MPP) MQDISC çağrısını yaparsa, açık kuyruklar kapanır, ancak örtük eşitleme noktası alınmaz. Uygulama olağan bir şekilde sona ererse, açık kuyruklar kapatılır ve örtük kesinleştirme gerçekleşir. Uygulama olağandışı bir şekilde sona ererse, açık kuyruklar kapanır ve örtük bir geriletme oluşur.

z/OS *IMS uygulamalarında MQI çağrıları*

Sunucu uygulamalarında ve Sorgu uygulamalarında MQI çağrılarının kullanımı hakkında bilgi edinmek için bu bilgileri kullanın.

Bu kısım, aşağıdaki IMS uygulaması tiplerinde MQI çağrılarının kullanımını kapsar:

- “Sunucu uygulamaları” sayfa 863
- “Sorgu uygulamaları” sayfa 866

Sunucu uygulamaları

MQI sunucusu uygulama modelinin anahattı:

```
Initialize/Connect
.
Open queue for input shared
.
Get message from IBM MQ queue
.
```

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

CSQ4ICB3 örnek programı, bu modeli kullanan bir BMP ' nin somutlamasını C/370içinde gösterir. Program önce IMS ile, sonra da IBM MQile iletişim kurar:

```

main()
----
Call InitIMS
If IMS initialization successful
Call InitMQM
If IBM MQ initialization successful
Call ProcessRequests
Call EndMQM
End-if
End-if

Return
```

IMS kullanıma hazırlama, programın ileti odaklı bir BMP olarak mı, yoksa toplu iş odaklı bir BMP olarak mı çağrılacağını belirler ve IBM MQ kuyruk yöneticisi bağlantısını ve kuyruk tanıtıcılarını uygun şekilde denetler:

```

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

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

Return to calling function
```

IBM MQ kullanıma hazırlama, kuyruk yöneticisine bağlanır ve kuyrukları açar. İleti odaklı bir BMP 'de bu, her IMS eşitleme noktası alındıktan sonra çağrılır; toplu iş odaklı BMP' de bu yalnızca program başlatılırken çağrılır:

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

Return to calling function
```

MPP 'de sunucu modelinin uygulanması, MPP' nin her çağırma için tek bir iş birimini işlemesi gerçeğinden etkilenir. Bunun nedeni, bir eşitleme noktası (GU) alındığında, bağlantı ve kuyruk tanıtıcıları kapatılır ve sonraki IMS iletisi teslim edilir. Bu sınırlama, aşağıdakilerden biri tarafından kısmen aşılabilir:

- **Tek bir iş birimi içindeki birçok iletinin işlenmesi**

  Bu şunları içerir:

  – İleti okuma

  – Gerekli güncellemeler işleniyor

  – Yanıt konuyor

  Tüm iletiler işleninceye kadar ya da ileti sayısı üst sınırı belirleninceye kadar bir döngüde. Bu durumda bir eşitleme noktası alınır.

  Yalnızca belirli uygulama tiplerine (örneğin, basit bir veritabanı güncellemesi ya da sorgusu) bu şekilde yaklaşılabilir. MQI yanıt iletileri, işlenmekte olan MQI iletisinin yaratıcısının yetkisiyle birlikte konabilse de, IMS kaynak güncellemelerinin güvenlik etkilerinin dikkatli bir şekilde ele alınması gerekir.

- **MPP çağrısı başına bir iletinin işlenmesi ve kullanılabilir tüm iletileri işlemek için MPP ' nin birden çok zamanlanmasının sağlanması.**

  MPP hareketini, IBM MQ kuyruğunda ileti olduğunda ve ona hizmet eden uygulama olmadığında zamanlamak için IBM MQ IMS tetikleyici izleme programını (CSQQTRMN) kullanın.

  Tetikleyici izleyicisi MPP ' yi başlatırsa, kuyruk yöneticisi adı ve kuyruk adı aşağıdaki COBOL kod alımlarında gösterildiği gibi programa geçirilir:

```
* Data definition extract
01 WS-INPUT-MSG.
05 IN-LL1         PIC S9(3) COMP.
05 IN-ZZ1         PIC S9(3) COMP.
05 WS-STRINGPARM      PIC X(1000).
01 TRIGGER-MESSAGE.
COPY CMQTMC2L.
*
* Code extract
GU-IOPCB SECTION.
MOVE SPACES TO WS-STRINGPARM.
CALL 'CBLTDLI' USING GU,
IOPCB,
WS-INPUT-MSG.
IF IOPCB-STATUS = SPACES
MOVE WS-STRINGPARM TO MQTMC.
*  ELSE handle error
*
```

```
* Now use the queue manager and queue names passed
DISPLAY 'MQTMC-QMGRNAME     ='
MQTMC-QMGRNAME OF MQTMC '='.
DISPLAY 'MQTMC-QNAME        ='
MQTMC-QNAME  OF MQTMC '='.
```

Uzun süreli bir görev olması beklenen sunucu modeli, bir toplu işleme bölgesinde daha iyi desteklenir, ancak BMP CSQQTRMN kullanılarak tetiklenemez.

Sorgu uygulamaları

Sorgu ya da güncelleme başlatan tipik bir IBM MQ uygulaması aşağıdaki gibi çalışır:

- Kullanıcıdan veri topla
- Bir ya da daha fazla IBM MQ iletisi koy
- Yanıt iletilerini al (bunları beklemeniz gerekebilir)
- Kullanıcıya bir yanıt sağlayın

IBM MQ kuyruklarına konan iletiler, kesinleştirilinceye kadar diğer IBM MQ uygulamaları tarafından kullanılamayacağından, bunların ya uyumluluk noktasından dışarı konması ya da IMS uygulamasının iki harekete bölünmesi gerekir.

Sorgu tek bir ileti koymayı içeriyorsa, *eşitleme noktası yok* seçeneğini kullanabilirsiniz; ancak sorgu daha karmaşıksa ya da kaynak güncellemeleri söz konusu ise, hata oluşursa ve eşitleme kullanmazsanız tutarlılık sorunları alabilirsiniz.

Bunun üstesinden gelmek için, bir programdan programa ileti anahtarı kullanarak MQI çağrılarını kullanarak IMS MPP hareketlerini bölebilirsiniz; bununla ilgili bilgi için *IMS Sistemlerarası İletişim (ISC)* başlıklı konuya bakın. Bu, bir sorgu programının MPP ' de uygulanmasını sağlar:

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

z/OS *IMS köprü uygulamalarının yazılması*

Bu konuda, IBM MQ - IMS köprüsünü kullanmak üzere uygulamalar yazılmasına ilişkin bilgiler yer alır.

IBM MQ - IMS köprüsü hakkında bilgi için bkz. IMS köprüsü.

IBM MQ for z/OSüzerinde IMS köprü uygulamalarının yazılmasına ilişkin daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- “IMS köprüsü iletileri nasıl ele alır?” sayfa 70
- “IMS hareket programlarını IBM MQ aracılığıyla yazma” sayfa 873

İlgili kavramlar

“IMS uygulamalarının IBM MQ kullanılarak yazılması” sayfa 66

IMS uygulamalarında IBM MQ kullanılırken hangi MQ API çağrılarının kullanılabileceğini ve uyumluluk noktası için kullanılan mekanizmayı içeren başka noktalar da vardır.

z/OS *IMS köprüsü iletileri nasıl ele alır?*

Bir IMS uygulamasına ileti göndermek için IBM MQ - IMS köprüsünü kullandığınızda, iletilerinizi özel bir biçimde oluşturmanız gerekir.

İletilerinizi, hedef IMS sisteminin XCF grubunu ve üye adını belirten bir depolama sınıfıyla tanımlanmış IBM MQ kuyruklarına da koymanız gerekir. Bunlar MQ-IMS köprü kuyrukları ya da yalnızca **köprü** kuyrukları olarak bilinir.

IBM MQ-IMS köprüsü, QSGDISP (QMGR) ile tanımlandıysa ya da NOSHARE seçeneğiyle birlikte QSGDISP (SHARED) ile tanımlandıysa, köprü kuyruğuna dışlayıcı giriş erişimi (MQOO_INPUT_EXCLUSIVE) gerektirir.

Bir kullanıcının IMS uygulamasına ileti göndermeden önce IMS ' da oturum açması gerekmez. Güvenlik denetimi için MQMD yapısının *UserIdentifier* alanındaki kullanıcı kimliği kullanılır. Denetim düzeyi, IBM MQ IMSile bağlantı kurduğunda belirlenir ve IMS köprüsü için uygulama erişim denetimiiçinde açıklanır. Bu, sözde oturum açma gerçekleştirilmesini sağlar.

IBM MQ - IMS köprüsü aşağıdaki ileti tiplerini kabul eder:

- IMS hareket verilerini ve MQIIH yapısını içeren iletiler (MQIIH içinde açıklanmıştır):

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

  Not:

  1. Köşeli ayraç, [], isteğe bağlı çoklu kesimleri temsil eder.
  2. MQIIH yapısını kullanmak için MQMD yapısının *Format* alanını MQFMT_IMS olarak ayarlayın.
- IMS hareket verilerini içeren, ancak MQIIH yapısını içermeyen iletiler:

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

IBM MQ , ileti verilerinin geçerliliğini denetleyerek, LL byte 'ları toplamının yanı sıra MQIIH ' nin uzunluğunun (varsa) ileti uzunluğuna eşit olmasını sağlar.

IBM MQ - IMS köprüsü, köprü kuyruklarından ileti aldığında, bu iletileri aşağıdaki gibi işler:

- İleti bir MQIIH yapısı içeriyorsa, köprü MQIIH 'yi doğrular (bkz. MQIIH), OTMA üstbilgilerini oluşturur ve iletiyi IMS' e gönderir. Giriş iletisinde hareket kodu belirtildi. Bu bir LTERM ise, IMS bir DFS1288E iletisiyle yanıt gönderir. Hareket kodu bir komutu gösteriyorsa, IMS komutu yürütür; tersi durumda, ileti hareket için IMS içinde kuyruğa alınır.
- İleti IMS işlem verileri içeriyorsa, ancak MQIIH yapısı yoksa, IMS köprüsü aşağıdaki varsayımları yapar:
  - Hareket kodu, kullanıcı verilerinin 5-12 bayttır.
  - Hareket, etkileşimli olmayan kipte
  - Hareket, kesinleştirme kipi 0 'da (kesinleştirme sonrası gönderme)
  - MQMD ' deki *Format* , *MFSMapName* (girişte) olarak kullanılır
  - Güvenlik kipi MQISS_CHECK

  Yanıt iletisi MQIIH yapısı olmadan da oluşturulur ve IMS çıkışının *MFSMapName* ürününden MQMD için *Format* kullanılır.

IBM MQ - IMS köprüsü, her IBM MQ kuyruğu için bir ya da iki Tpipo kullanır:

- Synchronized Tpipe, Kesinleştirme kipi 0 (COMMIT_THEN_SEND) kullanan tüm iletiler için kullanılır (bunlar IMS /DIS TMEMBER istemcisi TPIPE xxxx komutunun durum alanında SYN ile gösterilir)
- Kesinleştirme kipi 1 'i (SEND_THEN_COMMIT) kullanan tüm iletiler için eşitlenmemiş bir Tpipe kullanılır

Tpipo 'lar, ilk kullanıldığında IBM MQ tarafından oluşturulur. IMS yeniden başlatılıncaya kadar eşitlenmemiş bir Tpipe var. Eşitlenmiş Tpipo 'lar, IMS soğuk başlatılıncaya kadar var olur. Bu Tpipo ' ları kendiniz silemezsiniz.

IBM MQ - IMS köprüsünün iletilerle nasıl ilgilendiğine ilişkin ek bilgi için aşağıdaki konulara bakın:

- “IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi” sayfa 72
- “İleti IMS kuyruğuna konamazsa” sayfa 72
- “IMS köprü geribildirim kodları” sayfa 73
- “IMS köprüsündeki iletilerdeki MQMD alanları” sayfa 73
- “IMS köprüsündeki iletilerdeki MQIIH alanları” sayfa 74
- “IMS ' den yanıt iletileri” sayfa 75
- “IMS hareketlerinde alternatif yanıt PCB ' lerinin kullanılması” sayfa 75
- “IMS ' dan istenmeyen iletiler gönderme” sayfa 75
- “İleti bölümlemesi” sayfa 76
- “IMS köprüsüne/köprüsünden gelen iletiler için veri dönüştürme” sayfa 76

İlgili kavramlar

“IMS hareket programlarını IBM MQ aracılığıyla yazma” sayfa 873
IMS hareketlerini IBM MQ aracılığıyla işlemek için gereken kodlama, IMS işleminin gerektirdiği ileti biçimine ve döndürebileceği yanıt aralığına bağlıdır. Ancak, uygulamanız IMS ekran biçimlendirme bilgilerini işlerken göz önünde bulundurulması gereken birkaç nokta vardır.

z/OS *IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi*

IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesini açıklayan bir tablo.

Çizelge 132. IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi

IBM MQ ileti tipi	**Commit-then-send (kip 0)- eşitlenmiş IMS Tpipes kullanır**	**Send-then-commit (kip 1)- eşitlenmemiş IMS Tpipes kullanır**
Kalıcı IBM MQ iletileri	• Kurtarılabilir tam işlevli hareketler • Kurtarılamaz hareketler IMS tarafından reddedilir	• Fastpath işlemleri • Etkileşimli hareketler • Tam işlevli hareketler
Kalıcı olmayan IBM MQ iletileri	• Kurtarılamaz tam işlevli hareketler • IMS V8 ve APAR PQ61404 ve IMS ürününün sonraki tüm sürümleriyle kurtarılabilir hareketlere izin verilir.	• Fastpath işlemleri • Etkileşimli hareketler • Tam işlevli hareketler

Not: IMS komutları, kesinleştirme kipi 0 olan kalıcı IBM MQ iletilerini kullanamaz. Ek bilgi için Kesinleştirme kipi (commitMode) konusuna bakın.

z/OS *İleti IMS kuyruğuna konamazsa*

İleti IMS kuyruğuna konamazsa yapılacak işlemlerle ilgili bilgi edinin.

İleti IMS kuyruğuna konamazsa, IBM MQtarafından aşağıdaki işlem kullanılır:

- İleti geçersiz olduğu için bir ileti IMS ' e konamazsa, ileti teslim edilmeyen ileti kuyruğuna yerleştirilir ve sistem konsoluna bir ileti gönderilir.
- İleti geçerliyse, ancak IMStarafından reddedildiyse, IBM MQ sistem konsoluna bir hata iletisi gönderirse, ileti IMS algılama kodunu içerir ve IBM MQ iletisi teslim edilmeyen ileti kuyruğuna yerleştirilir. IMS algılama kodu 001Aise, IMS yanıt kuyruğuna hatanın nedenini içeren bir IBM MQ iletisi gönderir.

Not: Daha önce listelenen durumlarda, IBM MQ iletiyi herhangi bir nedenle teslim edilmeyen iletiler kuyruğuna koyamazsa, ileti kaynak IBM MQ kuyruğuna döndürülür. Sistem konsoluna bir hata iletisi gönderilir ve kuyruktan başka ileti gönderilmez.

İletileri yeniden göndermek için aşağıdakilerden ***birini*** gerçekleştirin:

- Kuyruğun karşılığı olan IMS içindeki Tpipes öğesini durdurun ve yeniden başlatın.
- Kuyruğu GET (DISABLED) olarak değiştirin ve yeniden GET (ENABLED) olarak değiştirin
- IMS ya da OTMA ' yı durdurun ve yeniden başlatın
- IBM MQ altsisteminizi durdurun ve yeniden başlatın

- İleti IMS tarafından bir ileti hatası dışında bir hata için reddedilirse, IBM MQ iletisi kaynak kuyruğa döndürülür, IBM MQ kuyruğu işlemeyi durdurur ve sistem konsoluna bir hata iletisi gönderilir.

  Bir kural dışı durum raporu iletisi gerekiyorsa, köprü bunu, oluşturanın yetkisiyle yanıt kuyruğuna koyar. İleti kuyruğa konamazsa, rapor iletisi köprünün yetkisiyle teslim edilmeyen iletiler kuyruğuna yerleştirilir. DLQ ' ya konamazsa atılır.

z/OS *IMS köprü geribildirim kodları*

IMS algılama kodları genellikle CSQ2001I gibi IBM MQ konsol iletilerinde onaltılı biçimde çıkılır (örneğin, algılama kodu 0x001F). Gönderilmeyen iletiler kuyruğuna konan iletilerin girilmeyen harf üstbilgisinde görülen IBM MQ geribildirim kodları ondalık sayılardır.

IMS köprüsü geribildirim kodları 301-399 ya da NACK algılama kodu 0x001Aiçin 600-855 aralığındadır. Bunlar IMS-OTMA algılama kodlarından aşağıdaki gibi eşlenir:

1. IMS-OTMA algılama kodu onaltılı bir sayıdan ondalık sayıya dönüştürülür.
2. IBM MQ *Feedback* kodu verilerek, 1' deki hesaplamadan elde edilen sayıya 300 eklenir.
3. IMS-OTMA algılama kodu 0x001A, ondalık 26 özel bir durumdur. 600-855 aralığında bir *Geribildirim* kodu oluşturulur.
   a. IMS-OTMA neden kodu onaltılı sayıdan ondalık sayıya dönüştürülür.
   b. aiçindeki hesaplamadan elde edilen sayıya 600 eklenir ve IBM MQ *Geribildirim* kodu verir.

IMS-OTMA algılama kodları hakkında bilgi için bkz. NAK iletileri için OTMA algılama kodları.

z/OS *IMS köprüsündeki iletilerdeki MQMD alanları*

IMS köprüsünden iletilerdeki MQMD alanları hakkında bilgi edinin.

Kaynak iletinin MQMD ' si, OTMA üstbilgilerinin Kullanıcı Verileri kısmında IMS tarafından taşınır. İleti IMSiçinde oluşursa, bu, IMS Hedef Çözüm Çıkışı tarafından oluşturulur. IMS 'den alınan bir iletinin MQMD' si aşağıdaki gibi oluşturulmuştur:

StrucID
"MD"

Sürüm
MQMD_VERSION_1

Rapor
MQRO_NONE

MsgType
MQMT_REPLY

Son kullanma tarihi
MQIIH ' nin İşaretler alanında MQIIH_PASS_EXPIRATION ayarlandıysa, bu alan kalan süre bitimini içerir, aksi takdirde MQEI_UNLIMITED olarak ayarlanır

Geri bildirim
MQFB_NONE

Kodlama
MQENC.Native (z/OS sisteminin kodlaması)

CodedCharSetId
MQCCSI_Q_MGR (z/OS sisteminin CodedCharSetID)

Biçim
MQMD.Format biçimi MQFMT_IMS, tersi durumda IOPCB.MODNAME

Öncelik
MQMD.Priority

Kalıcılık
Kesinleştirme kipine bağlıdır: CM-1; kalıcılık, CM-0 ise IMS iletisinin kurtarılabilirliği ile eşleşiyorsa, giriş iletisinin MQMD.Persistence .

MsgId
MQMD.MsgId MQRO_PASS_MSG_ID ise, tersi durumda Yeni MsgId (varsayılan)

CorrelId
MQMD.CorrelId (MQRO_PASS_CORREL_ID ise), tersi durumda MQMD.MsgId (varsayılan)

BackoutCount
0

ReplyToQ
Boşluklar

ReplyToQMgr
Boşluklar (MQPUT sırasında kuyruk yöneticisi tarafından yerel qmgr adına ayarlayın)

UserIdentifier
MQMD.UserIdentifier

AccountingToken
MQMD.AccountingToken

ApplIdentityVerileri
MQMD.ApplIdentityData

PutApplTipi
Hata yoksa MQAT_XCF, yoksa MQAT_BRIDGE

PutApplAdı
<XCFgroupName> <XCFmemberName> hata yoksa, QMGR adı

PutDate
İletinin konma tarihi

PutTime
İletinin konma zamanı

ApplOriginVerileri
Boşluklar

z/OS *IMS köprüsündeki iletilerdeki MQIIH alanları*

IMS köprüsünden iletilerdeki MQIIH alanları hakkında bilgi edinin.

IMS ' den alınan bir iletinin MQIIH 'si aşağıdaki gibi oluşturulmuştur:

StrucId
"IIH"

Sürüm
1

StrucLength
84

Kodlama
MQENC_NATIVE

CodedCharSetId
MQCCSI_Q_MGR

Biçim
MQIIH.ReplyToFormat ya da MQIIH.ReplyToFormat boş değilse, IOPCB.MODNAME

İşaretler
0

LTermOverride
OTMA üstbilgisinden LTERM adı (Tpipe)

MFSMapName
OTMA üstbilgisinden eşlem adı

ReplyToBiçimi
Boşluklar

Kimlik doğrulayıcı
Yanıt iletisi bir MQ-IMS köprü kuyruğuna konuluyorsa, giriş iletisinin MQIIH.Authenticator , tersi durumda boşluklar.

TranInstanceTanıtıcısı
Etkileşimde ise, OTMA üstbilgisinden etkileşim tanıtıcısı/sunucu simgesi. IMS ' nin V14öncesi sürümlerinde, bu alan etkileşim içinde değilse her zaman boş olur. IMS V14 ' ten itibaren bu alan, etkileşim içinde olmasa da IMS tarafından ayarlanabilir.

TranState
Sohbette "C" varsa, tersi durumda boş

CommitMode
OTMA üstbilgisinden kesinleştirme kipi ("0" ya da "1")

SecurityScope
Boş

Ayrıldı
Boş

z/OS *IMS ' den yanıt iletileri*

Bir IMS hareketi ISR 'leri IOPCB 'sine yönlendirdiğinde, ileti kaynak LTERM ya da TPIPE' ye geri yönlendirilir.

Bunlar IBM MQ içinde yanıt iletileri olarak görülür. IMS ' den gelen yanıt iletileri, özgün iletide belirtilen yanıt kuyruğuna yerleştirilir. İleti, yanıt kuyruğuna konamazsa, köprünün yetkisi kullanılarak teslim edilmeyen iletiler kuyruğuna yerleştirilir. İleti, teslim edilmeyen iletiler kuyruğuna konamazsa, iletinin alınamadığını söylemek için IMS ' e negatif bir alındı bildirimi gönderilir. Daha sonra, iletinin sorumluluğu IMS' e döndürülür. Kesinleştirme kipi 0 'ı kullanıyorsanız, bu Tpipe ' dan gelen iletiler köprüye gönderilmez ve IMS kuyruğunda kalır; yani, yeniden başlatılıncaya kadar başka ileti gönderilmez. Kesinleştirme kipi 1 'i kullanıyorsanız, diğer işler devam edebilir.

Yanıtın MQIIH yapısı varsa, biçim tipi MQFMT_IMS; değilse, biçim tipi iletiyi eklerken kullanılan IMS MOD adıyla belirtilir.

z/OS *IMS hareketlerinde alternatif yanıt PCB ' lerinin kullanılması*

Bir IMS hareketi alternatif yanıt PCB 'leri (ALTPCB' ye ilişkin IST 'ler) kullandığında ya da değiştirilebilir bir PCB' ye CHNG çağrısı verdiğinde, iletinin yeniden yönlendirilip yönlendirilmeyeceğini belirlemek için ön yöneltme çıkışı (DFSYPRX0) çağrılır.

İleti yeniden yönlendirilirse, hedefi onaylamak ve üstbilgi bilgilerini hazırlamak için hedef çözüm çıkışı (DFSYDRU0) çağrılır. Bu çıkış programlarıyla ilgili bilgi için bkz. IMS içinde OTMA çıkışlarının kullanılması ve Ön yönlendirme çıkışı DFSYPRX0 .

Çıkışlarda işlem yapılmazsa, IOPCB 'ye ya da ALTPCB' ye IBM MQ kuyruk yöneticisinden başlatılan tüm IMS hareketleri aynı kuyruk yöneticisine döndürülür.

z/OS *IMS ' dan istenmeyen iletiler gönderme*

IMS 'den IBM MQ kuyruğuna ileti göndermek için, IST' lerin ALTPCB ' ye gönderdiği bir IMS işlemini başlatmanız gerekir.

IMS 'den istenmeyen iletileri yönlendirmek ve OTMA kullanıcı verilerini oluşturmak için ön yöneltme ve hedef çözüm çıkışlarını yazmanız gerekir; böylece, iletinin MQMD' si doğru bir şekilde oluşturulabilir. Bu çıkış programlarına ilişkin bilgi için Ön yöneltme çıkışı DFSYPRX0 ve Hedef çözüm kullanıcı çıkışı başlıklı konuya bakın.

Not: IBM MQ - IMS köprüsü, aldığı bir iletinin yanıt mı, yoksa istenmeyen bir ileti mi olduğunu bilmiyor. İletiyle birlikte gelen OTMA UserData 'ya dayalı olarak yanıtın MQMD ve MQIIH' yi oluşturarak, her durumda iletiyi aynı şekilde işler.

İstenmeyen iletiler yeni Tpipes yaratabilir. Örneğin, var olan bir IMS hareketi yeni bir LTERM ' ye (örneğin, PRINT01) geçtiyse, ancak uygulama çıkışın OTMA aracılığıyla teslim edilmesini gerektiriyorsa, yeni bir Tpipe (bu örnekte PRINT01 olarak adlandırılır) yaratılır. Varsayılan olarak bu, eşitlenmemiş bir Tpipe 'dir. Uygulama iletinin kurtarılabilir olmasını gerektiriyorsa, hedef çözüm çıkış işaretini ayarlayın. Daha fazla bilgi için bkz. *IMS Özelleştirme Kılavuzu* .

z/OS *İleti bölümlemesi*

IMS hareketlerini tek ya da çok bölümlü giriş beklenirken tanımlayabilirsiniz.

Kaynak IBM MQ uygulaması, MQIIH yapısını izleyen kullanıcı girişini bir ya da daha çok LLZZ veri bölümü olarak oluşturmalıdır. IMS iletisinin tüm bölümleri, tek bir MQPUT ile gönderilen tek bir IBM MQ iletisinde bulunmalıdır.

Bir LLZZ veri kesiminin uzunluk üst sınırı IMS/OTMA (32767 bayt) tarafından tanımlanır. Toplam IBM MQ ileti uzunluğu, LL baytlarının toplamı ve MQIIH yapısının uzunluğudur.

Yanıtın tüm bölümleri tek bir IBM MQ iletisinde bulunur.

MQFMT_IMS_VAR_STRING biçimindeki iletilere ilişkin 32 KB sınırlaması üzerinde ek bir kısıtlama vardır. ASCII karışık CCSID iletisindeki veriler EBCDIC karışık CCSID iletisine dönüştürüldüğünde, SBCS ve DBCS karakterleri arasında her geçişte bir çift bayt ya da çift bayt dizilimi başlangıç bayt dizilimi başlangıç ve bitiş karakterleri eklenir. 32 KB sınırlaması, iletinin büyüklük üst sınırı için geçerlidir. Yani, iletideki LL alanı 32 KB 'yi aşamayacağı için, iletinin tüm çift bayt dizilimi başlangıç ve bitiş karakterleri de içinde olmak üzere 32 KB' yi aşmaması gerekir. İletinin oluşturulmasına izin veren uygulama buna izin vermelidir.

z/OS *IMS köprüsüne/köprüsünden gelen iletiler için veri dönüştürme*

Veri dönüştürme, bir iletiyi depolama sınıfı için tanımlanmış XCF bilgilerini içeren bir hedef kuyruğa yerleştirdiğinde, dağıtılmış kuyruğa alma olanağı (gerekli çıkışları çağırabilecek) ya da grup içi kuyruğa alma aracısı (çıkış kullanımını desteklemeyen) tarafından gerçekleştirilir. Bir ileti yayınlama/abone olma yoluyla kuyruğa teslim edildiğinde veri dönüştürme gerçekleşmez.

Gereken çıkışlar, CSQXLIB DD deyimiyle gönderme yapılan veri kümesindeki dağıtılmış kuyruğa alma olanağı için kullanılabilir olmalıdır. Bu, herhangi bir IBM MQ platformundan IBM MQ - IMS köprüsünü kullanarak bir IMS uygulamasına ileti gönderebileceğiniz anlamına gelir.

Dönüştürme hataları varsa, ileti dönüştürülmeden kuyruğa yerleştirilir; bu, sonunda köprünün üstbilgi biçimini tanıyamamasından dolayı IBM MQ - IMS köprüsü tarafından hata olarak kabul edilmesiyle sonuçlanır. Bir dönüştürme hatası ortaya çıkarsa, z/OS konsoluna bir hata iletisi gönderilir.

Genel olarak veri dönüştürmeye ilişkin ayrıntılı bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941 .

IBM MQ - IMS köprüsüne ileti gönderilmesi

Dönüştürmenin doğru gerçekleştirildiğinden emin olmak için, kuyruk yöneticisine iletinin biçiminin ne olduğunu söylemeniz gerekir.

İletinin MQIIH yapısı varsa, MQMD 'deki *Format* yerleşik biçimde MQFMT_IMS olarak ayarlanmalı ve MQIIH' deki *Format* , ileti verilerinizi tanımlayan biçimin adına ayarlanmalıdır. MQIIH yoksa, MQMD ' deki *Format* değerini biçim adınıza ayarlayın.

Verileriniz (LLZZ ' ler dışında) tüm karakter verileriyse (MQCHAR), yerleşik MQFMT_IMS_VAR_STRING biçimini biçim adınız (MQIIH ya da MQMD içinde) olarak kullanın. Tersi durumda, kendi biçim adınızı kullanın; bu durumda, biçiminiz için bir veri dönüştürme çıkışı da sağlamanız gerekir. Çıkış, verinin kendisine ek olarak, iletinizdeki LLZZ 'lerin dönüştürülmesini de işlemelidir (ancak iletinin başında herhangi bir MQIIH' yi işlemesi gerekmez).

Uygulamanız *MFSMapName*kullanıyorsa, bunun yerine MQFMT_IMS ile iletileri kullanabilir ve MQIIH ' nin MFSMapName alanında IMS hareketine geçirilen eşlem adını tanımlayabilirsiniz.

IBM MQ - IMS köprüsünden ileti alınması

IMS' e göndermekte olduğunuz özgün iletide bir MQIIH yapısı varsa, yanıt iletisinde de bir MQIIH yapısı vardır.

Yanıtınız doğru olarak dönüştürüldüğünden emin olmak için:

- Özgün iletinizde MQIIH yapısı varsa, özgün iletinin MQIIH *ReplytoFormat* alanında yanıt iletiniz için istediğiniz biçimi belirtin. Bu değer, yanıt iletisinin MQIIH *Format* alanına yerleştirilir. Bu özellikle, tüm çıkış verileriniz LLZZ < karakter verileri > biçimindeyse kullanışlıdır.
- Özgün iletinizde MQIIH yapısı yoksa, yanıt iletisi için istediğiniz biçimi IMS uygulamasının IOPCB 'ye ilişkin ISRT' de MFS MOD adı olarak belirtin.

z/OS *IMS hareket programlarını IBM MQ aracılığıyla yazma*

IMS hareketlerini IBM MQ aracılığıyla işlemek için gereken kodlama, IMS işleminin gerektirdiği ileti biçimine ve döndürebileceği yanıt aralığına bağlıdır. Ancak, uygulamanız IMS ekran biçimlendirme bilgilerini işlerken göz önünde bulundurulması gereken birkaç nokta vardır.

Bir IMS hareketi 3270 ekranından başlatıldığında, ileti IMS İleti Biçimi Hizmetleri 'nden geçer. Bu işlem tarafından görülen veri akışından tüm uçbirim bağımlılığını kaldırabilir. Bir işlem OTMA aracılığıyla başlatıldığında, MFS dahil olmaz. MFS ' de uygulama mantığı uygulanırsa, bu yeni uygulamada yeniden oluşturulmalıdır.

Bazı IMS hareketlerinde, son kullanıcı uygulaması bazı 3270 ekran davranışını değiştirebilir; örneğin, geçersiz veri girilmiş bir alanı vurgulayabilir. Bu tip bilgiler, program tarafından değiştirilmesi gereken her ekran alanı için IMS iletisine iki baytlık bir öznitelik alanı eklenerek iletilir.

Bu nedenle, bir uygulamayı 3270 ' i taklit etmek üzere kodluyorsanız, ileti oluştururken ya da alırken bu alanları dikkate almanız gerekir.

Aşağıdaki işlemleri yapmak için programınızdaki bilgileri kodlemeniz gerekebilir:

- Hangi tuşa basılacağı (örneğin, Enter ve PF1)
- İleti uygulamanıza geçirildiğinde imlecin bulunduğu yer
- Öznitelik alanlarının IMS uygulaması tarafından ayarlanıp ayarlanmadığını belirler
  - Yüksek, normal ya da sıfır yoğunluğu
  - Renk
  - IMS ' un bir sonraki Enter tuşuna basışında alanın geri dönmesinin beklenip beklenmediğini belirler.
- IMS uygulamasının herhangi bir alanda boş karakterler (X'3F') kullanıp kullanmadığını belirler.

IMS iletiniz yalnızca karakter verileri içeriyorsa (LLZZ veri kesimi dışında) ve bir MQIIH yapısı kullanıyorsanız, MQMD biçimini MQFMT_IMS ve MQIIH biçimini MQFMT_IMS_VAR_STRING olarak ayarlayın.

IMS iletiniz yalnızca karakter verileri içeriyorsa (LLZZ veri kesimi dışında) ve MQIIH yapısı **kullanmıyorsanız** , MQMD biçimini MQFMT_IMS_VAR_STRING olarak ayarlayın ve IMS uygulamanızın yanıt verirken MODname MQFMT_IMS_VAR_STRING belirttiğinden emin olun. Bir sorun ortaya çıkarsa

(örneğin, kullanıcı hareketi kullanma yetkisine sahip değilse) ve IMS bir hata iletisi gönderirse, DFSMOx biçiminde bir MODname değeri vardır; burada x, 1-5 aralığında bir sayıdır. Bu, MQMD.Format.

IMS iletiniz ikili, paketlenmiş ya da kayan noktalı veriler içeriyorsa (LLZZ veri kesimi dışında), kendi veri dönüştürme yordamlarınızı kodlayın. IMS ekran biçimlendirmesiyle ilgili bilgi için *IMS/ESA Application Programming: Transaction Manager* belgesine bakın.

IBM MQaracılığıyla IMS hareketlerini işlemek için kod yazarken aşağıdaki konuları göz önünde bulundurun.

- "IMS etkileşimli hareketlerini çağırmak için IBM MQ uygulamaları yazılıyor" sayfa 874
- "IMS komutlarını içeren programlar yazılması" sayfa 874
- "Tetikleme" sayfa 874

IMS etkileşimli hareketlerini çağırmak için IBM MQ uygulamaları yazılıyor

Bu bilgileri, IMS etkileşimli hareketlerini çağırmak için IBM MQ uygulamasını yazarken dikkat edilecek noktalar için bir kılavuz olarak kullanın.

IMS etkileşimi çağıran bir uygulama yazarken aşağıdakileri göz önünde bulundurun:

- Uygulama iletinize bir MQIIH yapısı ekleyin.
- MQIIH içindeki *CommitMode* öğesini MQICM_SEND_THEN_COMMIT olarak ayarlayın.
- Yeni bir etkileşimi çağırmak için, MQIIH ' de *TranState* değerini MQITS_NOT_IN_ETKILEŞIM olarak ayarlayın.
- Bir etkileşimin ikinci ve sonraki adımlarını çağırmak için *TranState* değerini MQITS_IN_ETKILEŞIM olarak ayarlayın ve *TranInstanceId* değerini, etkileşimin önceki adımında döndürülen alanın değerine ayarlayın.
- IMS' den gönderilen özgün iletiyi kaybederseniz, IMS içinde *TranInstanceId*değerini bulmanın kolay bir yolu yoktur.
- Uygulama, IMS hareketinin etkileşimi sonlandırıp sonlandırmadığını denetlemek için IMS iletilerinin *TranState* adresini denetlemelidir.
- Bir konuşmayı sonlandırmak için /EXIT kullanabilirsiniz. *TranInstanceId*değerini tırnak içine almalı, *TranState* değerini MQITS_IN_ETKILEŞIM olarak ayarlamalı ve etkileşimin gerçekleştirilmekte olduğu IBM MQ kuyruğunu kullanmalısınız.
- Bir etkileşimi tutmak ya da serbest bırakmak için /HOLD ya da /REL kullanamazsınız.
- IMS yeniden başlatılırsa, IBM MQ - IMS köprüsü aracılığıyla çağrılan etkileşimler sonlandırılır.

IMS komutlarını içeren programlar yazılması

Bir uygulama programı, *komut* ' un /DIS TRAN PART ya da /DIS POOL ALL biçiminde olduğu bir hareket yerine, LLZZ*komut*biçiminde bir IBM MQ iletisi oluşturabilir.

Çoğu IMS komutu bu şekilde yayınlanabilir; ayrıntılar için bkz. *IMS V11 İletişim ve Bağlantılar* . Komut çıkışı, görüntülenmek üzere 3270 uçbirimine gönderileceği şekilde, metin biçimindeki IBM MQ yanıt iletisinde alınır.

OTMA, IMS görüntü hareketi komutunun özel bir biçimini uyguladı; bu, çıkışın mimarlı bir biçimini döndürür. Tam biçim *IMS V11 İletişim ve Bağlantılar*içinde tanımlanır. Bu formu bir IBM MQ iletisinden çağırmak için, ileti verilerini önceki gibi (örneğin, /DIS TRAN PART) oluşturun ve MQIIH ' deki TranState alanını MQITS_ARCHITECTED olarak ayarlayın. IMS komutu işler ve yanıtı mimari biçimde döndürür. Mimarili bir yanıt, çıktının metin biçiminde bulunabilecek tüm bilgileri ve bir ek bilgi parçasını içerir: hareketin kurtarılabilir olarak mı, yoksa kurtarılamaz olarak mı tanımlandığını.

Tetikleme

IBM MQ - IMS köprüsü tetikleyici iletilerini desteklemez.

XCF parametreleriyle bir depolama sınıfı kullanan bir başlatma kuyruğu tanımlarsanız, o kuyruğa konan iletiler köprüye geldiklerinde reddedilir.

İstemci yordamsal uygulamaları yazılıyor

IBM MQ üzerinde yordam dili kullanarak istemci uygulamaları yazmak için bilmeniz gerekenleri.

Uygulamalar, IBM MQ istemci ortamında oluşturulabilir ve çalıştırılabilir. Uygulama oluşturulmalı ve kullanılan IBM MQ MQI client ile bağlantılandırılmalıdır. Uygulamaların oluşturulma ve bağlanma şekli, kullanılan platforma ve programlama diline göre değişir. İstemci uygulamalarının nasıl oluşturulacağına ilişkin bilgi için bkz. “IBM MQ MQI clients için uygulama oluşturma” sayfa 880.

Belirli koşulların yerine getirilmesi koşuluyla, bir IBM MQ uygulamasını hem tam IBM MQ ortamında hem de kodunuzu değiştirmeden IBM MQ MQI client ortamında çalıştırabilirsiniz. Uygulamalarınızı IBM MQ istemci ortamında çalıştırmaya ilişkin daha fazla bilgi için bkz. “IBM MQ MQI client ortamında uygulamaları çalıştırma” sayfa 882.

Bir IBM MQ MQI client ortamında çalışacak uygulamaları yazmak için ileti kuyruğu arabirimini (MQI) kullanırsanız, IBM MQ uygulama işleminin kesintiye uğramamasını sağlamak için MQI çağrısı sırasında uygulanması gereken bazı ek denetimler vardır. Bu denetimlerle ilgili daha fazla bilgi için bkz. “İstemci uygulamasında MQI kullanılması” sayfa 876.

Diğer uygulama tiplerinin istemci uygulamaları olarak hazırlanması ve çalıştırılması hakkında bilgi edinmek için aşağıdaki konulara bakın:

- “CICS ve Tuxedo uygulamalarının hazırlanması ve çalıştırılması” sayfa 894
- “Microsoft Transaction Server uygulamalarının hazırlanması ve çalıştırılması” sayfa 47
- “IBM MQ JMS uygulamalarının hazırlanması ve çalıştırılması” sayfa 896

İlgili kavramlar

“Uygulama geliştirme kavramları” sayfa 6
IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

“IBM MQ için uygulama geliştirilmesi” sayfa 5
İleti göndermek ve almak için uygulamalar geliştirebilir ve kuyruk yöneticilerinizi ve ilgili kaynakları yönetebilirsiniz. IBM MQ , birçok farklı dilde ve çerçevede yazılmış uygulamaları destekler.

“IBM MQ uygulamaları için tasarımla ilgili önemli noktalar” sayfa 47
Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğine karar verdiğinizde, IBM MQtarafından sunulan özellikleri nasıl kullanacağınıza karar vermeniz gerekir.

“Kuyruğa alma için yordam uygulaması yazılması” sayfa 692
Kuyruğa alma uygulamaları yazma, bir kuyruk yöneticisine bağlanma ve bağlantı kesme, yayınlama/abone olma ve nesneleri açma ve kapatma hakkında bilgi edinmek için bu bilgileri kullanın.

“Yayınlama/abone olma uygulamaları yazılıyor” sayfa 776
Yayınlama/abone olma IBM MQ uygulamaları yazmaya başlayın.

“Yordamsal uygulama oluşturma” sayfa 957
Birkaç yordam dilinden birinde bir IBM MQ uygulaması yazabilir ve uygulamayı birkaç farklı platformda çalıştırabilirsiniz.

“Yordamsal program hatalarının işlenmesi” sayfa 994
Bu bilgiler, bir çağrı yaptığında ya da iletisi son hedefine teslim edildiğinde, uygulamalarınızla ilişkili MQI çağrılarıyla ilgili hataları açıklar.

İlgili görevler

“IBM MQ örnek yordam programlarının kullanılması” sayfa 1013
Bu örnek programlar yordamsal dillerde yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir. Farklı platformlarda IBM MQ programları.

İstemci uygulamasında MQI kullanılması

Bu konu derlemi, IBM MQ uygulamanızın bir ileti kuyruğu arabirimi (MQI) istemci ortamında çalışması ve tam IBM MQ kuyruk yöneticisi ortamında çalışması arasındaki farkları dikkate alır.

Bir uygulama tasarlarken, IBM MQ uygulama işlemesinin kesintiye uğramamasını sağlamak için MQI çağrısı sırasında uygulamanız gereken denetimleri göz önünde bulundurun.

MQI kullanan uygulamaları çalıştırabilmeniz için önce belirli IBM MQ nesnelerini yaratmanız gerekir. Daha fazla bilgi için bkz. MQI kullanan uygulama programları.

İstemci uygulamasındaki bir iletinin boyutunun sınırlanması

Kuyruk yöneticisinin ileti uzunluğu üst sınırı vardır, ancak bir istemci uygulamasından iletebileceğiniz ileti büyüklüğü üst sınırı kanal tanımlamasıyla sınırlıdır.

Bir kuyruk yöneticisinin ileti uzunluğu üst sınırı (MaxMsgUzunluğu) özniteliği, o kuyruk yöneticisi tarafından işlenebilecek ileti uzunluğu üst sınırıdır.

Multi Çoklu platformlar' da, bir kuyruk yöneticisinin ileti uzunluğu üst sınırını artırabilirsiniz. Ek bilgi için bkz. ALTER QMGR.

Bir kuyruk yöneticisine ilişkin MaxMsguzunluğunun değerini MQINQ çağrısıyla öğrenebilirsiniz.

MaxMsgLength özniteliği değiştirilirse, yeni değerden daha uzun kuyruklar ve hatta iletiler olup olmadığı denetlenmez. Bu özniteliği değiştirdikten sonra, değişikliğin yürürlüğe girdiğinden emin olmak için uygulamaları ve kanalları yeniden başlatın. Bu durumda, kuyruk yöneticisinin ya da kuyruğun (kuyruk yöneticisi bölümlemesine izin verilmedikçe) MaxMsguzunluğunu aşan yeni iletiler oluşturulamaz.

Bir kanal tanımındaki ileti uzunluğu üst sınırı, bir istemci bağlantısı boyunca iletebileceğiniz bir iletinin büyüklüğünü sınırlar. Bir IBM MQ uygulaması MQPUT ya da MQGET çağrıyı bundan daha büyük bir iletiyle kullanmayı denerse, uygulamaya bir hata kodu döndürülür. Kanal tanımlamasının ileti büyüklüğü üst sınırı değiştirgesi, bir istemci bağlantısı üzerinden MQCB kullanılarak kullanılabilecek ileti büyüklüğü üst sınırını etkilemez.

İlgili kavramlar

“MQCONNX kullanılıyor” sayfa 880
MQCNO yapısında bir kanal tanımlaması (MQCD) yapısı belirtmek için MQCONNX çağrıyı kullanabilirsiniz.

İlgili başvurular

İleti uzunluğu üst sınırı (MAXMSGL)

KANAL DEğIşTIR

2010 (07DA) (RC2010): MQRC_DATA_LENGTH_ERROR

İstemci ya da sunucu CCSID değerinin seçilmesi

İstemci için yerel kodlanmış karakter takımı tanıtıcısını (CCSID) kullanın. Kuyruk yöneticisi gerekli dönüştürmeyi gerçekleştirir. CCSID ' yi geçersiz kılmak için **MQCCSID** ortam değişkenini kullanabilirsiniz. Uygulamanız birden çok PUT gerçekleştiriyorsa, ilk PUT tamamlandıktan sonra MQMD ' nin CCSID ve kodlama alanlarının üzerine yazılabilir.

Uygulamadan istemci sınırlı kod öbeğine iletilen veriler, IBM MQ MQI clientiçin kodlanmış yerel CCSID ' de olmalıdır. Bağlı kuyruk yöneticisi verilerin dönüştürülmesini gerektiriyorsa, dönüştürme kuyruk yöneticisindeki istemci destek kodu tarafından gerçekleştirilir.

IBM WebSphere MQ 7.0 ve sonraki sürümlerde, kuyruk yöneticisi bunu yapamazsa Java istemcisi dönüştürmeyi gerçekleştirebilir. Bkz. “IBM MQ classes for Java istemci bağlantıları” sayfa 359.

İstemci kodu, istemcideki MQI 'ı geçen karakter verilerinin o iş istasyonu için yapılandırılmış CCSID' de olduğunu varsayar. Bu CCSID desteklenmeyen bir CCSID ise ya da gerekli CCSID değilse, aşağıdaki komutlardan biri kullanılarak **MQCCSID** ortam değişkeniyle geçersiz kılınabilir:

- Windows

```
SET MQCCSID=850
```

- Linux AIX

```
export MQCCSID=850
```

- IBM i

```
ADDENVVAR ENVVAR(MQCCSID) VALUE(37)
```

Bu değiştirge tanıtımda ayarlanırsa, tüm MQI verilerinin 850 kod sayfasında olduğu varsayılır.

Not: Kod sayfası 850 ile ilgili varsayım, iletideki uygulama verileri için geçerli değildir.

Uygulamanız ileti tanımlayıcısından (MQMD) sonra IBM MQ üstbilgilerini içeren birden çok PUT gerçekleştiriyorsa, ilk PUT tamamlandıktan sonra MQMD ' nin CCSID ve kodlama alanlarının üzerine yazıldığını unutmayın.

İlk PUT işleminden sonra bu alanlar, IBM MQ üstbilgilerini dönüştürmek için bağlı kuyruk yöneticisi tarafından kullanılan değeri içerir. Uygulamanızın değerleri, gerektirdiği değerlere geri koyduğundan emin olun.

İstemci somutlamasında MQINQ kullanılması

MQINQ kullanılarak sorgulanan bazı değerler istemci kodu tarafından değiştirilir.

CCSID
Kuyruk yöneticisinin değil, istemci CCSID değerine ayarlanır.

MaxMsgUzunluğu
kanal tanımı tarafından sınırlandıysa azaltılır. Bu, aşağıda belirtilenlerin en aşağısı olacaktır:

- Kuyruk tanımlamasında tanımlanan değer ya da
- Kanal tanımında tanımlanan değer

Daha fazla bilgi için bkz. MQINQ.

İstemci uygulamasında eşitleme noktası koordinasyonunu kullanma

Temel istemcide çalışan bir uygulama MQCMIT ve MQBACK verebilir, ancak eşitleme noktası denetiminin kapsamı MQI kaynaklarıyla sınırlıdır. Bir dış hareket yöneticisini genişletilmiş bir işlem istemcisiyle kullanabilirsiniz.

IBM MQiçinde, kuyruk yöneticisinin rollerinden biri, bir uygulama içindeki eşitleme noktası denetimine sahiptir. Bir uygulama IBM MQ temel istemcisinde çalışıyorsa, MQCMIT ve MQBACK verebilir, ancak eşitleme noktası denetiminin kapsamı MQI kaynaklarıyla sınırlıdır. IBM MQ komutu MQBEGIN, temel istemci ortamında geçerli değil.

Sunucudaki tam kuyruk yöneticisi ortamında çalışan uygulamalar, bir hareket izleme programı aracılığıyla birden çok kaynağı (örneğin, veritabanları) koordine edebilir. Sunucuda, IBM MQ ürünleriyle verilen Transaction Monitor ya da CICSgibi başka bir hareket izleyicisini kullanabilirsiniz. Temel istemci uygulamasıyla hareket izleyici kullanamazsınız.

IBM MQ genişletilmiş hareket istemcisi ile bir dış hareket yöneticisi kullanabilirsiniz. Bkz. Genişletilmiş işlem istemcisi nedir? detaylar için.

İstemci uygulamasında önden okuma özelliğinin kullanılması

İstemci uygulamasının iletileri istemesi gerekmeden, kalıcı olmayan iletilerin bir istemciye gönderilmesine izin vermek için istemcide önden okuma özelliğini kullanabilirsiniz.

Bir istemci sunucudan gelen bir iletiyi gerektiriyorsa, sunucuya bir istek gönderir. Kullandığı her ileti için ayrı bir istek gönderir. Bu istek iletilerini göndermek zorunda kalmaktan kaçınarak kalıcı olmayan iletileri kullanan bir istemcinin performansını artırmak için, bir istemci önden okuma özelliğini kullanacak şekilde

yapılandırılabilir. Önden okuma, iletilerin bir uygulama tarafından istenmeden istemciye gönderilmesini sağlar.

Önden okuma özelliğinin kullanılması, bir istemci uygulamasındaki kalıcı olmayan iletileri kullanırken başarımı artırabilir. Bu performans iyileştirmesi hem MQI hem de JMS uygulamaları tarafından kullanılabilir. MQGET ya da zamanuyumsuz tüketim kullanan istemci uygulamaları, kalıcı olmayan iletileri kullanırken performans geliştirmelerinden yararlanır.

MQOO_READ_ÖNCEDEN ile MQOPEN ' i çağırdığınızda, IBM MQ istemcisi belirli koşullar karşılandığında yalnızca önden okuma seçeneğini etkinleştirir. Bu koşullar şunlardır:

- İstemci uygulaması derlenmeli ve iş parçacıklı IBM MQ MQI istemcisi kitaplıklarına bağlanmalıdır.
- İstemci kanalı TCP/IP protokolünü kullanıyor olmalıdır
- Kanal, istemci ve sunucu kanalı tanımlarında sıfır dışında bir SharingConversations (SHARECNV) ayarı içermelidir.

Önden okuma etkinleştirildiğinde, istemcide önden okuma arabelleği adı verilen bir arabelleğe iletiler gönderilir. İstemcinin, önden okuma özelliği etkinleştirilmiş olarak açmış olduğu her kuyruk için bir önden okuma arabelleği vardır. Önden okuma arabelleğindeki iletiler kalıcı olarak saklanmaz. İstemci, belirli aralıklarla sunucuyu tükettiği veri miktarına ilişkin bilgilerle günceller.

Tüm seçenekler kullanım için desteklenmediğinden, tüm istemci uygulaması tasarımları önden okuma özelliğini kullanmaya uygun değildir. Önden okuma etkinleştirildiğinde bazı seçeneklerin MQGET çağrıları arasında tutarlı olması gerekir. Bir istemci MQGET çağrıları arasında seçim ölçütlerini değiştirirse, önden okuma arabelleğinde saklanan iletiler istemcinin önden okuma arabelleğinde kalır. Daha fazla bilgi için bkz. “Kalıcı olmayan iletilerin performansını artırma” sayfa 754

Önden okuma yapılandırması, IBM MQ istemci yapılandırma dosyasının MessageBuffer bölümünde belirtilen üç öznitelik (MaximumSize, PurgeTimeve UpdatePercentage) tarafından denetlenir.

İstemci uygulamasına zamanuyumsuz koyma özelliğinin kullanılması

Bir uygulama, zamanuyumsuz koyma özelliğini kullanarak, kuyruk yöneticisinden yanıt beklemeden bir iletiyi kuyruğa yerleştirebilir. Bazı durumlarda ileti sistemi performansını artırmak için bunu kullanabilirsiniz.

Olağan durumda, bir uygulama MQPUT ya da MQPUT1kullanan bir kuyruğa ileti ya da ileti koyduğunda, uygulamanın kuyruk yöneticisinin MQI isteğini işlediğini doğrulamasını beklemesi gerekir. İleti sistemi başarımını, özellikle istemci bağ tanımlarını kullanan uygulamalar ve çok sayıda küçük iletiyi kuyruğa yerleştiren uygulamalar için, iletileri zamanuyumsuz olarak yerleştirmeyi seçerek artırabilirsiniz. Bir uygulama bir iletiyi zamanuyumsuz olarak yerleştirdiğinde, kuyruk yöneticisi her çağrıya ilişkin başarıyı ya da başarısızlığı döndürmez, bunun yerine belirli aralıklarla hata olup olmadığını denetleyebilirsiniz.

Bir iletiyi zamanuyumsuz olarak kuyruğa koymak için, MQPMO yapısının *Options* alanında MQPMO_ASYNC_RESPONSE seçeneğini kullanın.

Bir ileti zamanuyumsuz yerleştirme için uygun değilse, zamanuyumlu olarak kuyruğa konur.

MQPUT ya da MQPUT1için zamanuyumsuz koyma yanıtı istenirken, CompCode ve MQCC_OK ve MQRC_NONE nedenleri iletinin başarıyla kuyruğa konduğunu göstermez. Her bir MQPUT ya da MQPUT1 çağrısının başarılı ya da başarısız olması hemen döndürülmese de, zamanuyumsuz bir çağrı altında oluşan ilk hata daha sonra MQSTAT çağrısıyla saptanabilir.

MQPMO_ASYNC_RESPONSE ile ilgili daha fazla ayrıntı için bkz. MQPMO seçenekleri.

Zamanuyumsuz Put örnek programı, kullanılabilir özelliklerden bazılarını gösterir. Programın özelliklerine ve tasarımına ilişkin ayrıntılar ve programın nasıl çalıştırılacağı hakkında bilgi için bkz. “Zamanuyumsuz Put örnek programı” sayfa 1031.

İstemci uygulamasında paylaşım sohbetlerini kullanma

Sohbet paylaşımına izin verilen bir ortamda, sohbetler bir MQI kanalı eşgörünümünü paylaşabilir.

Paylaşım sohbetleri, biri kanal tanımlaması (MQCD) yapısının bir parçası ve biri kanal çıkış parametresi (MQCXP) yapısının bir parçası olan SharingConversationsadlı iki alan tarafından denetlenir. MQCD ' deki

SharingConversations alanı, kanalla ilişkili bir kanal eşgörünümünü paylaşabilecek etkileşim sayısı üst sınırını belirleyen bir tamsayı değeridir. MQCXP ' deki SharingConversations alanı, kanal eşgörünümünün paylaşılıp paylaşılmadığını gösteren bir Boole değeridir.

Paylaşım etkileşimine izin verilmeyen bir ortamda, aynı MQCD ' leri belirten yeni istemci bağlantıları bir kanal yönetim ortamını paylaşmaz.

Aşağıdaki koşullar geçerli olduğunda yeni bir istemci uygulaması bağlantısı kanal örneğini paylaşacaktır:

- Kanal örneğinin hem istemci bağlantısı hem de sunucu bağlantısı uçları, etkileşimleri paylaşmak üzere yapılandırılır ve bu değerler kanal çıkışları tarafından geçersiz kılınmaz.
- İstemci bağlantısı MQCD değeri (istemci MQCONNX çağrısında ya da istemci kanal tanımlama çizelgesinden (CCDT) sağlanır), istemci MQCONNX çağrısında ya da CCDT ' den sağlanan istemci bağlantısı MQCD değeriyle tam olarak eşleşir. Özgün MQCD ' nin daha sonra çıkışlar ya da kanal anlaşması tarafından değiştirilmiş olabileceğini, ancak bu değişiklikler yapılmadan önce istemci sistemine sağlanan değerle eşleşmenin yapıldığını unutmayın.
- Sunucu tarafındaki paylaşım etkileşimleri sınırı aşılmadı.

Yeni bir istemci uygulaması bağlantısı, bir kanal örneğini diğer etkileşimlerle paylaşarak çalıştırma ölçütleriyle eşleşirse, bu karar, o etkileşimle ilgili herhangi bir çıkış çağrılmadan önce verilir. Böyle bir etkileşimin çıkışları, kanal örneğini diğer sohbetlerle paylaştığı gerçeğini değiştiremez. Yeni kanal tanımıyla eşleşen kanal eşgörünümü yoksa, yeni bir kanal eşgörünümü bağlanır.

Kanal kararlaştırması yalnızca bir kanal örneğinde ilk etkileşim için gerçekleşir; kanal örneğine ilişkin kararlaştırılan değerler o aşamada düzeltilmiştir ve sonraki etkileşimler başladığında değiştirilemez. TLS kimlik doğrulaması yalnızca ilk etkileşim için de gerçekleşir.

MQCD SharingConversations değeri, herhangi bir güvenlik kullanıma hazırlanırken değiştirilirse, istemci-bağlantıda ya da kanal örneğinin sunucu bağlantısı ucunda yuvadaki ilk etkileşim için çıkış gönder ya da al seçeneği değiştirilirse, kanal örneğine ilişkin paylaşım etkileşimleri değerini belirlemek için bu çıkışlardan sonra yeni değer kullanılır (en düşük değer önceliklidir).

Paylaşım etkileşimleri için kararlaştırılan değer sıfırsa, kanal örneği hiçbir zaman paylaşılmaz. Bu alanı sıfır değerine ayarlanan diğer çıkış programları da kendi kanal örneklerinde benzer şekilde çalışır.

Paylaşım etkileşimleri için kararlaştırılan değer sıfırdan büyükse, sonraki çıkışlara yönelik çağrılar için MQCXP SharingConversations TRUE olarak ayarlanır ve bu, bu kanal örneğindeki diğer çıkış programlarının bununla aynı anda girilebileceğini gösterir.

Bir kanal çıkış programı yazdığınızda, programın paylaşım etkileşimleri içerebilecek bir kanal örneğinde çalışıp çalışmayacağını göz önünde bulundurun. Kanal eşgörünümü paylaşım etkileşimleri içerebilirse, değişen MQCD alanlarının kanal çıkışının diğer eşgörünümleri üzerindeki etkiyi göz önünde bulundurun; tüm MQCD alanlarının tüm paylaşım etkileşimleri boyunca ortak değerleri vardır. Kanal yönetim ortamı oluşturulduktan sonra, çıkış programları MQCD alanlarını değiştirmeye çalışırsa, kanal yönetim ortamında çalışan çıkış programlarının diğer yönetim ortamları aynı anda aynı alanları değiştirme girişiminde bulunabildiği için sorunlarla karşılaşabilirler. Bu durum çıkış programlarınızda ortaya çıkacaksa, çıkış kodunuzda MQCD ' ye erişimi diziselleştirmeniz gerekir.

Etkileşimleri paylaşmak üzere tanımlanmış bir kanalla çalışıyorsanız, ancak belirli bir kanal eşgörünümünde paylaşımın gerçekleşmesini istemiyorsanız, kanal eşgörünümündeki ilk etkileşimde bir kanal çıkışını başlattığınızda SharingConversations MQCD değerini 1 ya da 0 olarak ayarlayın. SharingConversationsdeğerlerinin açıklaması için bkz. SharingConversations .

Örnek

Paylaşım sohbetleri etkinleştirildi.

Çıkış programını belirleyen bir istemci-bağlantı kanalı tanımı kullanıyorsunuz.

Bu kanal ilk kez başlatıldığında, çıkış programı ilk kullanıma hazırlandığında bazı MQCD parametrelerini değiştirir. Bunlar kanal tarafından çalıştırılır, bu nedenle kanalın çalıştığı tanım artık başlangıçta sağlanandan farklı. MQCXP SharingConversations parametresi TRUE olarak ayarlandı.

Uygulama bu kanalı kullanarak bir sonraki kez bağlandığında, etkileşim, aynı özgün kanal tanımına sahip olduğu için daha önce başlatılan kanal örneğinde çalışır. Uygulamanın ikinci kez bağlandığı kanal örneği, ilk kez bağlandığı kanal örneğiyle aynı. Sonuç olarak, çıkış programı tarafından değiştirilen tanımları kullanır. Çıkış programı ikinci etkileşim için kullanıma hazırlandığında, MQCD alanlarını değiştirebilse de, bunlar kanal tarafından gerçekleştirilmez. Bu aynı özellikler, kanal örneğini paylaşan sonraki etkileşimler için de geçerlidir.

MQCONNX kullanılıyor

MQCNO yapısında bir kanal tanımlaması (MQCD) yapısı belirtmek için MQCONNX çağrıyı kullanabilirsiniz.

Bu, çağıran istemci uygulamasının yürütme sırasında istemci bağlantısı kanalının tanımını belirtmesini sağlar. Daha fazla bilgi için bakınız: Creating a client-connection channel on the IBM MQ MQI client using MQCNO. MQCONNX kullandığınızda, sunucuda verilen çağrı sunucu düzeyine ve dinleyici yapılandırmasına bağlıdır.

Bir istemciden MQCONNX kullandığınızda aşağıdaki seçenekler yoksayılır:

- MQCNO_STANDARD_BINDING
- MQCNO_FASTPATH_BINDING

Kullanabileceğiniz MQCD yapısı, kullandığınız MQCD sürüm numarasına bağlıdır. MQCD sürümleriyle (MQCD_VERSION) ilgili bilgi için MQCD Sürümübaşlıklı konuya bakın. Örneğin, kanal çıkış programlarını sunucuya geçirmek için MQCD yapısını kullanabilirsiniz. MQCD Sürüm 3 ya da üstünü kullanıyorsanız, yapıyı kullanarak sunucuya bir çıkış dizisi geçirebilirsiniz. Var olan bir çıkışı değiştirmek yerine, her işlem için bir çıkış ekleyerek şifreleme ve sıkıştırma gibi aynı ileti üzerinde birden çok işlem gerçekleştirmek için bu işlevi kullanabilirsiniz. MQCD yapısında bir dizi belirtmezseniz, tek çıkış alanları denetlenir. Kanal çıkış programlarına ilişkin ek bilgi için bkz. “İleti sistemi kanalları için kanal çıkış programları” sayfa 920.

MQCONNX üzerinde paylaşılan bağlantı tanıtıcıları

Paylaşılan bağlantı tanıtıcılarını kullanarak, aynı işlem içindeki farklı iş parçacıkları arasında tanıtıcıları paylaşabilirsiniz.

Paylaşılan bir bağlantı tanıtıcısı belirttiğinizde, MQCONNX çağrısından döndürülen bağlantı tanıtıcısı, işlemdeki herhangi bir iş parçacığına ilişkin sonraki MQI çağrılarında iletilebilir.

Not: Paylaşılan bağlantı tanıtıcılarını desteklemeyen bir sunucu kuyruk yöneticisine bağlanmak için IBM MQ MQI client üzerinde paylaşılan bağlantı tanıtıcısını kullanabilirsiniz.

Daha fazla bilgi için “MQCONNX kullanılıyor” sayfa 880 başlıklı konuya bakın.

IBM MQ MQI clients için uygulama oluşturma

Uygulamalar IBM MQ MQI client ortamında oluşturulabilir ve çalıştırılabilir. Uygulama oluşturulmalı ve kullanılan IBM MQ MQI client ile bağlantılandırılmalıdır. Uygulamaların oluşturulma ve bağlanma şekli, kullanılan platforma ve programlama diline göre değişir.

Bir uygulama istemci ortamında çalışacaksa, bu uygulamayı aşağıdaki çizelgede gösterilen dillerde yazabilirsiniz:

Çizelge 133. İstemci ortamlarında desteklenen programlama dilleri

İstemci platformu	C	C++	COBOL	pTAL	RPG	Visual Basic
AIX AIX	Evet	Evet	Evet			
IBM i IBM i	Evet		Evet		Evet	
Linux Linux	Evet	Evet	Evet			
Windows Windows	Evet	Evet	Evet			Evet

Multi *C uygulamalarının IBM MQ MQI client koduyla bağlanması*

IBM MQ MQI clientüzerinde çalıştırmak istediğiniz IBM MQ uygulamanızı yazdıktan sonra, bunu IBM MQ MQI client koduna bağlamanız gerekir.

Uygulamanızı IBM MQ MQI client koduna iki şekilde bağlayabilirsiniz:

1. Uygulamanızı doğrudan bir kuyruk yöneticisine bağlayarak, bu durumda kuyruk yöneticisinin uygulamanızla aynı makinede olması gerekir.
2. Aynı ya da farklı bir makinedeki kuyruk yöneticilerine erişmenize olanak veren istemci kitaplığı dosyasına.

IBM MQ , her ortam için bir istemci kitaplık dosyası sağlar:

AIX **AIX**

İş parçacığı kullanmayan uygulamalar için libmqic.a kitaplığı ya da iş parçacıklı uygulamalar için libmqic_r.a kitaplığı.

Linux **Linux**

İş parçacığı kullanmayan uygulamalar için libmqic.so kitaplığı ya da iş parçacıklı uygulamalar için libmqic_r.so kitaplığı.

IBM i **IBM i**

İş parçacıklı olmayan uygulamalar için LIBMQIC istemci hizmet programı ya da iş parçacıklı uygulamalar için LIBMQIC_R hizmet programı ile istemci uygulamasını bağlayın.

Windows **Windows**

MQIC32.LIB.

ALW *C++ uygulamalarının IBM MQ MQI client koduyla bağlanması*

C + +. Oluşturma yöntemleri, ortama göre değişir.

C++ uygulamalarınızın nasıl bağlanacağına ilişkin bilgi için IBM MQ C++ programları oluşturmabaşlıklı konuya bakın.

C + + kullanmanın tüm yönleriyle ilgili tüm ayrıntılar için C++ kullanarak başlıklı konuya bakın.

Multi *COBOL uygulamalarını IBM MQ MQI client koduyla bağlama*

IBM MQ MQI clientüzerinde çalıştırmak istediğiniz bir COBOL uygulamasını yazdıktan sonra, bunu uygun bir kitaplığa bağlamanız gerekir.

IBM MQ , her ortam için bir istemci kitaplık dosyası sağlar:

AIX **AIX**

İş parçacıklı olmayan COBOL uygulamanızı libmqicb.a kitaplığına ya da iş parçacıklı COBOL uygulamasına libmqicb_r.aile bağlayın.

IBM i **IBM i**

İş parçacığı kullanmayan uygulamalar için COBOL istemci uygulamasını AMQCSTUB hizmet programıyla ya da iş parçacığı kullanan uygulamalar için AMQCSTUB_R hizmet programıyla bağlayın.

Windows **Windows**

32 bit COBOL için uygulama kodunuzu MQICCBB kitaplığına bağlayın. IBM MQ MQI client for Windows , 16 bit COBOL ' unu desteklemez.

Windows *Visual Basic uygulamalarının IBM MQ MQI client koduyla bağlanması*

Microsoft Visual Basic uygulamalarını Windowsüzerindeki IBM MQ MQI client koduyla bağlayabilirsiniz.

Deprecated

IBM MQ 9.0' den Microsoft Visual Basic 6.0 desteği kullanımdan kaldırılmıştır. .NET için IBM MQ sınıfları önerilen değiştirme teknolojisidir. Daha fazla bilgi için bkz .NET uygulamaları geliştirilmesi.

Visual Basic uygulamanızı aşağıdaki içerme dosyalarıyla bağlantılandırın:

CMQB.bas
MQI

CMQBB.bas
MQAI

CMQCFB.bas
PCF komutları

CMQXB.bas
Kanallar

İstemci dll 'in doğru otomatik seçildiğinden emin olmak için, Visual Basic derleyicisinde istemci için `mqtype=2` ayarını yapın:

MQIC32.dll
Windows 7, Windows 8, Windows 2008 ve Windows 2012

İlgili kavramlar

“Visual Basic içinde kodlama” sayfa 1008
Microsoft Visual Basiciçinde IBM MQ programlarını kodlarken göz önünde bulundurulması gereken bilgiler. Visual Basic yalnızca Windowsüzerinde desteklenir.

“Windows ' da Visual Basic programlarının hazırlanması” sayfa 977
Windowsüzerinde Microsoft Visual Basic programlarını kullanırken göz önünde bulundurulması gereken bilgiler.

IBM MQ MQI client ortamında uygulamaları çalıştırma

Belirli koşulların yerine getirilmesi koşuluyla, bir IBM MQ uygulamasını hem tam IBM MQ ortamında hem de kodunuzu değiştirmeden IBM MQ MQI client ortamında çalıştırabilirsiniz.

Bu koşullar şunlardır:

- Uygulamanın eşzamanlı olarak birden çok kuyruk yöneticisine bağlanması gerekmez.
- Bir `MQCONN` ya da `MQCONNX` çağrısında kuyruk yöneticisi adının başına yıldız işareti (*) eklenmez.
- Uygulamanın Bir IBM MQ MQI clientüzerinde hangi uygulamalar çalışır? içinde listelenen kural dışı durumlardan herhangi birini kullanması gerekmez.

Not: Bağlantı düzenleme sırasında kullandığınız kitaplıklar, uygulamanızın çalışması gereken ortamı belirler.

IBM MQ MQI client ortamında çalışırken şunu unutmayın:

- IBM MQ MQI client ortamında çalışan her uygulamanın sunuculara kendi bağlantıları vardır. Bir uygulama, her `MQCONN` ya da `MQCONNX` çağrısında bir sunucuyla bağlantı kurar.
- Bir uygulama iletileri zamanuyumlu olarak gönderir ve alır. Bu, aramanın istemcide verildiği zaman ile ağda tamamlanma kodunun ve neden kodunun iadesi arasında bir bekleme anlamına gelir.
- Tüm veri dönüştürme işlemi sunucu tarafından gerçekleştirilir, ancak makinenin konfigürasyonu tanımlanmış CCSID değerini geçersiz kılmaya ilişkin bilgi edinmek için MQCCSID ' ye bakın.

IBM MQ MQI client uygulamalarının kuyruk yöneticilerine bağlanması

IBM MQ MQI client ortamında çalışan bir uygulama, bir kuyruk yöneticisine çeşitli yollarla bağlanabilir. Ortam değişkenlerini, MQCNO yapısını ya da bir istemci tanımlama çizelgesini kullanabilirsiniz.

IBM MQ istemci ortamında çalışan bir uygulama bir MQCONN ya da MQCONNX çağrısı verdiğinde, istemci bağlantıyı nasıl kuracağını belirler. Bir IBM MQ istemcisinde bir uygulama tarafından MQCONNX çağrısı yayınlandığında, MQI istemcisi kitaplığı istemci kanal bilgilerini aşağıdaki sırayla arar:

1. MQCNO yapısının `ClientConnOffset` ya da `ClientConnPtr` alanlarının içeriklerini kullanarak (sağlandıysa). Bu alanlar, istemci bağlantı kanalının tanımlaması olarak kullanılacak kanal tanımlama yapısını (MQCD) tanıtır. Bağlantı ayrıntıları, bağlantı öncesi çıkış kullanılarak geçersiz kılınabilir. Daha fazla bilgi için bkz "Havuzdan ön bağlantı çıkışı kullanarak bağlantı tanımlamalarına gönderme yapma" sayfa 950.
2. **MQSERVER** ortam değişkeni ayarlanırsa, tanımladığı kanal kullanılır.
3. Bir `mqclient.ini` dosyası tanımlanmışsa ve Kanallar kısmı bir **ServerConnectionParms** özniteliği içeriyorsa, tanımladığı kanal kullanılır. Daha fazla bilgi için bkz. IBM MQ MQI client yapılandırma dosyası, `mqclient.ini` ve İstemci yapılandırma dosyasının kanal kısmı.
4. **MQCHLLIB** ve **MQCHLTAB** ortam değişkenleri ayarlanırsa, işaret ettikleri istemci kanal tanımlama çizelgesi kullanılır. Alternatif olarak, IBM MQ 9.0' dan **MQCCDTURL** ortam değişkeni, **MQCHLLIB** ve **MQCHLTAB** ortam değişkenlerinin bir birleşimini ayarlamak için eşdeğer bir yetenek sağlar. **MQCCDTURL** ayarlanırsa, gösterdiği istemci kanal tanımlama çizelgesi kullanılır. Daha fazla bilgi için bkz. URL CCDT erişimi.
5. Bir `mqclient.ini` dosyası tanımlandıysa ve Kanallar kısmı **ChannelDefinitionDirectory** ve **ChannelDefinitionFile** özniteliklerini içeriyorsa, bu öznitelikler istemci kanal tanımlama çizelgesinin yerini belirlemek için kullanılır. Daha fazla bilgi için bkz. IBM MQ MQI client yapılandırma dosyası, `mqclient.ini` ve İstemci yapılandırma dosyasının kanal kısmı.
6. Son olarak, ortam değişkenleri ayarlanmazsa, istemci `mqs.ini` dosyasındaki AllQueueManager olanağının **DefaultPrefix** özniteliğinden oluşturulan bir yol ve ada sahip bir istemci kanal tanımlama çizelgesini arar. Daha fazla bilgi için bkz. AllQueue mqs.ini dosyasının Yöneticiler kısmı.

   İstemci kanal tanımlama çizelgesinin aranması başarısız olursa, istemci aşağıdaki yolları kullanır:

   - Linux AIX AIX and Linuxişletim tarihinde: `/var/mqm/AMQCLCHL.TAB`
   - Windows Windowsişletim tarihinde: `C:\Program Files\IBM\MQ\amqclchl.tab`
   - IBM i IBM iişletim tarihinde: `/QIBM/UserData/mqm/@ipcc`
   - MQ Appliance IBM MQ Applianceüzerinde: *QMname*`_AMQCLCHL.TAB`. Bunlar `mqbackup://URI`altında görünür.

Önceki listede açıklanan seçeneklerin ilki (MQCNO ' nun `ClientConnOffset` ya da `ClientConnPtr` alanları kullanılarak) yalnızca MQCONNX çağrısı tarafından desteklenir. Uygulama MQCONNX yerine MQCONN kullanıyorsa, kanal bilgileri listede gösterilen sırada kalan beş şekilde aranır. İstemci kanal bilgilerini bulamazsa, MQCONN ya da MQCONNX çağrısı başarısız olur.

Kanal adının (istemci bağlantısı için), MQCONN ya da MQCONNX çağrısıyla başarılı olması için sunucuda tanımlanan sunucu bağlantısı kanal adıyla eşleşmesi gerekir.

İlgili kavramlar

İstemci kanal tanımlama çizelgesine Web üzerinden adreslenebilir erişim

İlgili görevler

Sunucu ve istemci arasındaki bağlantıların yapılandırılması

İlgili başvurular

İstemci kanal tanımlama çizelgesi

MQCNO-Bağlantı seçenekleri

Ortam değişkenlerini kullanarak istemci uygulamalarının kuyruk yöneticilerine bağlanması
İstemci kanal bilgileri, ortam değişkenleri tarafından istemci ortamında çalışan bir uygulamaya sağlanabilir.

IBM MQ MQI client ortamında çalışan bir uygulama, aşağıdaki ortam değişkenlerini kullanarak bir kuyruk yöneticisine bağlanabilir:

MQSERVER

MQSERVER ortam değişkeni, en küçük bir kanal tanımlamak için kullanılır. **MQSERVER** , IBM MQ sunucusunun yerini ve kullanılacak iletişim yöntemini belirtir.

MQCHLLIB
MQCHLLIB ortam değişkeni, istemci kanal tanımlama çizelgesini (CCDT) içeren dosyanın dizin yolunu belirtir. Dosya sunucuda yaratılır, ancak IBM MQ MQI client iş istasyonuna kopyalanabilir.

MQCHLTAB
MQCHLTAB ortam değişkeni, istemci kanal tanımlama çizelgesini (CCDT) içeren kütüğün adını belirtir.

IBM MQ 9.0' den **MQCCDTURL** ortam değişkeni, **MQCHLLIB** ve **MQCHLTAB** ortam değişkenlerinin bir birleşimini ayarlamak için eşdeğer bir yetenek sağlar. **MQCCDTURL** , istemci kanal tanımlama çizelgesinin elde edilebileceği tek bir değer olarak bir dosya, ftp ya da http URL belirtmenizi sağlar. Daha fazla bilgi için bkz. İstemci kanal tanımlama çizelgesine adreslenebilir Web erişimi.

MQCNO yapısını kullanarak istemci uygulamalarının kuyruk yöneticilerine bağlanması
Kanal tanımlamasını, MQCONNX çağrısının MQCNO yapısı kullanılarak sağlanan bir kanal tanımlama yapısında (MQCD) belirtebilirsiniz.

Daha fazla bilgi için bakınız: Creating a client-connection channel on the IBM MQ MQI client using MQCNO.

İstemci kanal tanımlama çizelgesini kullanarak istemci uygulamalarının kuyruk yöneticilerine bağlanması
MQSC DEFINE CHANNEL komutunu kullanırsanız, sağladığınız ayrıntılar istemci kanal tanımlama çizelgesine (ccdt) yerleştirilir. MQCONN ya da MQCONNX çağrısının **QMgrName** değiştirgesinin içeriği, istemcinin hangi kuyruk yöneticisine bağlanacağını belirler.

Bir uygulamanın kullanacağı kanalı belirlemek için istemci bu dosyaya erişir. Birden fazla uygun kanal tanımı varsa, kanal seçimi istemci kanal ağırlığı (CLNTWGHT) ve bağlantı benzerliği (AFFINITY) kanal özniteliklerinden etkilenir.

Otomatik istemci yeniden bağlantısının kullanılması
Birkaç bileşen yapılandırarak, istemci uygulamalarınızın herhangi bir ek kod yazmadan otomatik olarak yeniden bağlanmasını sağlayabilirsiniz.

Otomatik istemci yeniden bağlantısı *yerleşik*. Bağlantı, istemci uygulama programının herhangi bir noktasında otomatik olarak geri yüklenir ve nesneleri açma tanıtıcılarının tümü geri yüklenir.

Buna karşılık, el ile yeniden bağlantı, istemci uygulamasının `MQCONN` ya da `MQCONNX`kullanarak yeniden bağlantı oluşturmasını ve nesneleri yeniden açmasını gerektirir. Otomatik istemci yeniden bağlantısı, tüm istemci uygulamaları için değil, pek ya: Sayısız uygulamalar için uygundur.

Daha fazla bilgi için bkz. Otomatik istemci yeniden bağlantısı.

İstemci kanal tanımlama çizelgesinin rolü
İstemci kanal tanımlama çizelgesi (CCDT), istemci bağlantı kanallarının tanımlarını içerir. Özellikle istemci uygulamalarınızın bir dizi alternatif kuyruk yöneticisine bağlanması gerekebilirse bu yararlıdır.

Bir kuyruk yöneticisi tanımladığınızda istemci kanal tanımlama çizelgesi yaratılır. Aynı dosya birden çok IBM MQ istemcisi tarafından kullanılabilir.

Bir istemci uygulamasının CCDT kullanması için çeşitli yollar vardır. CCDT istemci bilgisayara kopyalanabilir. CCDT ' yi birden çok istemci tarafından paylaşılan bir konuma kopyalayabilirsiniz. CCDT ' yi sunucuda bulunmaya devam ederken, istemcinin paylaşılan bir dosya olarak erişmesine olanak verebilirsiniz.

IBM MQ 9.0'den CCDT, bir URI aracılığıyla erişilebilen merkezi bir konumda barındırılabilir ve devreye alınan her istemci için CCDT' yi ayrı ayrı güncelleme gereksinmesi ortadan kaldırılabilir.

İlgili kavramlar
İstemci kanal tanımlama çizelgesine Web üzerinden adreslenebilir erişim
İlgili görevler
İstemci bağlantısı kanal tanımlarına erişilmesi
İlgili başvurular
İstemci kanal tanımlama çizelgesi

CCDT ' deki kuyruk yöneticisi grupları
İstemci kanal tanımlama çizelgesindeki (CCDT) bir bağlantı kümesini *kuyruk yöneticisi grubu*olarak tanımlayabilirsiniz. Bir uygulamayı, kuyruk yöneticisi grubunun bir parçası olan bir kuyruk yöneticisine bağlayabilirsiniz. Bu, bir MQCONN ya da MQCONNX çağrısında kuyruk yöneticisi adının yıldız işaretiyle önceden düzeltilmesiyle gerçekleştirilebilir.

Aşağıdaki nedenlerden ötürü birden çok sunucu makinesine bağlantı tanımlamayı seçebilirsiniz:

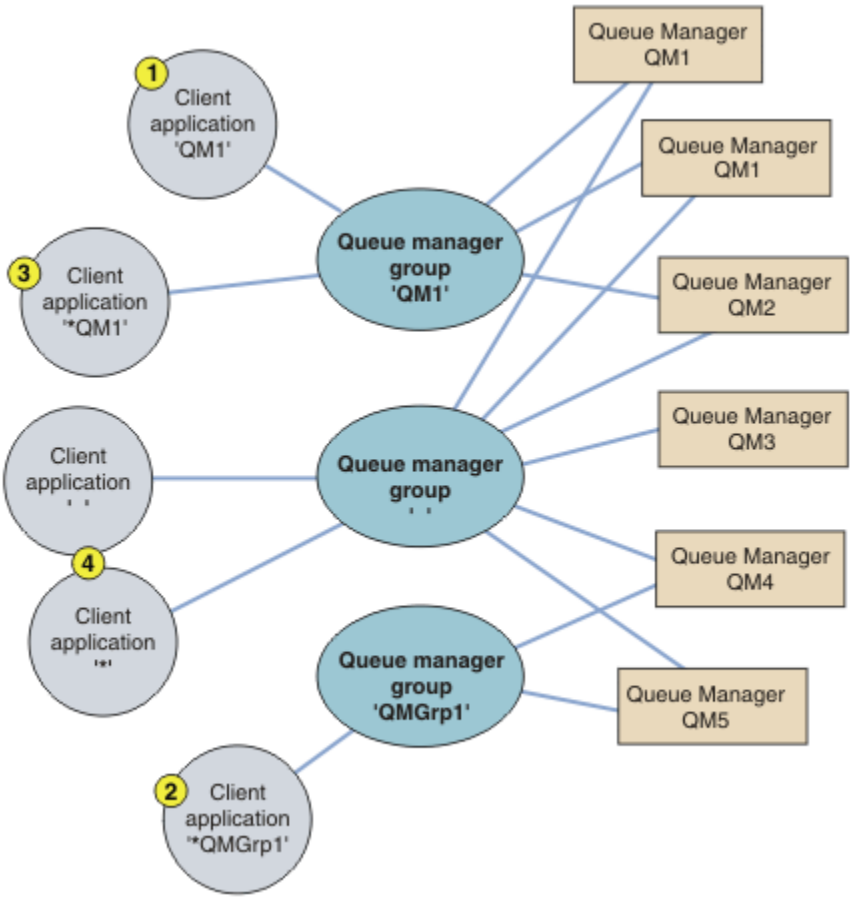
- Kullanılabilirliği artırmak için, bir istemciyi çalışmakta olan kuyruk yöneticileri kümesinden herhangi birine bağlamak istiyorsunuz.
- Bir istemciyi en son bağlandığı kuyruk yöneticisine yeniden bağlamak, ancak bağlantı başarısız olursa farklı bir kuyruk yöneticisine bağlanmak istiyorsunuz.
- Bağlantı başarısız olursa, istemci programında MQCONN komutunu yeniden çalıştırarak farklı bir kuyruk yöneticisine istemci bağlantısını yeniden denemek istiyorsunuz.
- Bağlantı başarısız olursa, istemci kodunu yazmadan başka bir kuyruk yöneticisine istemci bağlantısını otomatik olarak yeniden bağlamak istiyorsunuz.
- Yedek bir yönetim ortamı devralırsa, istemci bağlantısını, istemci kodu yazmadan, çok eşgörünümlü bir kuyruk yöneticisinin farklı bir yönetim ortamına otomatik olarak yeniden bağlamak istiyorsunuz.
- Bazı kuyruk yöneticilerine bağlanan istemcilerin sayısı diğerlerinden daha fazla olacak şekilde, bir dizi kuyruk yöneticisi arasındaki istemci bağlantılarını dengelemek istiyorsunuz.
- Yüksek hacimli bağlantıların bir hataya neden olması durumunda, birçok istemci bağlantısının birden çok kuyruk yöneticisine ve zaman içinde yeniden bağlanmasını dağıtmak istiyorsunuz.
- Herhangi bir istemci uygulama kodunu değiştirmeden kuyruk yöneticilerinizi taşımak istiyorsunuz.
- Kuyruk yöneticisi adlarını bilmesine gerek olmayan istemci uygulama programları yazmak istiyorsunuz.

Farklı kuyruk yöneticilerine bağlanmak her zaman uygun değildir. Örneğin, WebSphere Application Serveriçindeki bir genişletilmiş hareket istemcisi ya da Java istemcisi, tahmin edilebilir bir kuyruk yöneticisi yönetim ortamına bağlanmalıdır. Otomatik istemci yeniden bağlantısı IBM MQ classes for Javatarafından desteklenmez.

Kuyruk yöneticisi grubu, istemci kanal tanımlama çizelgesinde (CCDT) tanımlanan bir bağlantı takımıdır. Küme, kanal tanımlamalarında **QMNAME** özniteliğiyle aynı değere sahip olan üyeleri tarafından tanımlanır.

Şekil 97 sayfa 886 , üç kuyruk yöneticisi grubunu, CCDT 'de **QMNAME** (QM1) ve **QMNAME** (QMGrp1)olarak yazılan iki adlandırılmış kuyruk yöneticisi grubunu ve **QMNAME** (')olarak yazılan bir boş ya da varsayılan grubu gösteren bir istemci bağlantı çizelgesinin grafik gösterimidir.

1. Kuyruk yöneticisi grubu QM1 , kuyruk yöneticilerine QM1 ve QM2bağlayan üç istemci bağlantı kanalına sahiptir. QM1 , iki farklı sunucuda bulunan çok eşgörünümlü bir kuyruk yöneticisi olabilir.
2. Varsayılan kuyruk yöneticisi grubunun, bunu tüm kuyruk yöneticilerine bağlayan altı istemci bağlantısı kanalı vardır.
3. QMGrp1 , iki kuyruk yöneticisine (QM4 ve QM5) istemci bağlantısı kanallarına sahiptir.



Şekil 97. Kuyruk yöneticisi grupları

Bu istemci bağlantı çizelgesinin kullanılmasına ilişkin dört örnek, Şekil 97 sayfa 886içinde numaralandırılmış istemci uygulamalarının yardımıyla açıklanmıştır.

1. İlk örnekte istemci uygulaması, **QmgrName** değiştirgesi olarak QM1kuyruk yöneticisi adını MQCONN ya da MQCONNX MQI çağrısına geçirir. IBM MQ istemci kodu, eşleşen kuyruk yöneticisi grubunu (QM1) seçer. Grup üç bağlantı kanalı içerir ve IBM MQ MQI client , QM1adlı çalışan bir kuyruk yöneticisine bağlı bağlantı için bir IBM MQ dinleyicisi buluncaya kadar bu kanalların her birini kullanarak QM1 ' e bağlanmaya çalışır.

   Bağlantı girişimlerinin sırası, istemci bağlantısı AFFINITY özniteliğinin değerine ve istemci kanalı ağırlıklandırmalarına bağlıdır. Bu kısıtlamalar içinde, bağlantı kurma yükünü dağıtmak için bağlantı girişimlerinin sırası hem olası üç bağlantı üzerinden hem de zaman içinde rasgele hale getirilir.

   İstemci uygulaması tarafından verilen MQCONN ya da MQCONNX çağrısı, çalışmakta olan bir QM1yönetim ortamıyla bağlantı kurulduğunda başarılı olur.
2. İkinci örnekte istemci uygulaması, MQCONN ya da MQCONNX MQI çağrısına **QmgrName** parametresi olarak başına yıldız işareti (*QMGrp1) eklenmiş bir kuyruk yöneticisi adını geçirir. IBM MQ istemcisi, eşleşen kuyruk yöneticisi grubunu (QMGrp1) seçer. Bu grup iki istemci bağlantı kanalı içerir ve IBM MQ MQI client , sırayla her kanalı kullanarak *herhangi bir* kuyruk yöneticisine bağlanmayı dener. Bu örnekte, IBM MQ MQI client ' in başarılı bir bağlantı kurması gerekir; bağlandığı kuyruk yöneticisinin adı önemli değildir.

   Bağlantı girişimi sırasına ilişkin kural öncekiyle aynıdır. Tek fark, kuyruk yöneticisi adının başına yıldız işareti konarak, istemcinin kuyruk yöneticisinin adının ilgili olmadığını göstermesi.

   İstemci uygulaması tarafından verilen MQCONN ya da MQCONNX çağrısı, QMGrp1 kuyruk yöneticisi grubundaki kanallar tarafından bağlanan herhangi bir kuyruk yöneticisinin çalışan bir eşgörünümüyle bağlantı kurulduğunda başarılı olur.

3. Üçüncü örnek, **QmgrName** parametresinin başına bir yıldız işareti (*QM1) eklendiği için ikinciyle aynıdır. Bu örnek, tek bir kanal tanımlamasında QMNAME özniteliğini inceleyerek bir istemci kanal bağlantısının hangi kuyruk yöneticisine bağlanacağını saptayamayacağınızı gösterir. Kanal tanımlamasının **QMNAME** özniteliğinin QM1olması, QM1adlı bir kuyruk yöneticisiyle bağlantı kurulmasını zorunlu kılmak için yeterli değildir. İstemci uygulamanız **QmgrName** değiştirgesini yıldız işaretiyle öneklerse, herhangi bir kuyruk yöneticisi bağlantı hedefi olabilir.

   Bu durumda, çalışmakta olan bir QM1 ya da QM2örneğiyle bağlantı kurulduğunda istemci uygulaması tarafından verilen MQCONN ya da MQCONNX çağrıları başarılı olur.
4. Dördüncü örnek, varsayılan grubun kullanımını gösterir. Bu durumda istemci uygulaması, MQCONN ya da MQCONNX MQI çağrısına **QmgrName** parametresi olarak bir yıldız işareti ('*') ya da boşluk ' 'iletir. İstemci kanalı tanımlamasında kural olarak, boş bir **QMNAME** özniteliği varsayılan kuyruk yöneticisi grubunu belirtir ve boş ya da yıldız işareti **QmgrName** parametresi boş bir **QMNAME** özniteliğiyle eşleşir.

   Bu örnekte, varsayılan kuyruk yöneticisi grubunun tüm kuyruk yöneticilerine istemci kanal bağlantıları vardır. Varsayılan kuyruk yöneticisi grubu seçildiğinde, uygulama gruptaki herhangi bir kuyruk yöneticisine bağlı olabilir.

   İstemci uygulaması tarafından verilen MQCONN ya da MQCONNX çağrısı, herhangi bir kuyruk yöneticisinin çalışan bir yönetim ortamıyla bağlantı kurulduğunda başarılı olur.

   Not: Varsayılan grup, varsayılan kuyruk yöneticisinden farklıdır, ancak bir uygulama varsayılan kuyruk yöneticisi grubuna ya da varsayılan kuyruk yöneticisine bağlanmak için boş bir **QmgrName** parametresi kullanır. Varsayılan kuyruk yöneticisi grubu kavramı yalnızca bir istemci uygulaması ve bir sunucu uygulaması için varsayılan kuyruk yöneticisi ile ilgilidir.

İkinci ya da üçüncü bir kuyruk yöneticisine bağlanan kanallar da içinde olmak üzere, istemci bağlantı kanallarınızı yalnızca bir kuyruk yöneticisinde tanımlayın. Bunları iki kuyruk yöneticisinde tanımlamayın ve iki istemci kanal tanımlama çizelgelerini birleştirmeyi deneyin. İstemci yalnızca bir istemci kanal tanımlama çizelgesine erişebilir.

Örnekler

Konunun başında kuyruk yöneticisi gruplarını kullanma nedenlerinin listesine yeniden bakın. Bir kuyruk yöneticisi grubunun kullanılması bu yetenekleri nasıl sağlar?

Bir kuyruk yöneticisi kümesinden herhangi birine bağlanın.
Kümedeki tüm kuyruk yöneticilerine bağlantıları olan bir kuyruk yöneticisi grubu tanımlayın ve öneki yıldız işareti olan **QmgrName** değiştirgesini kullanarak gruba bağlanın.

Aynı kuyruk yöneticisine yeniden bağlanın, ancak son kez bağlantı kurulan kuyruk yöneticisi kullanılamıyorsa farklı bir kuyruk yöneticisine bağlanın.
Önceden olduğu gibi bir kuyruk yöneticisi grubu tanımlayın, ancak her istemci kanal tanımlamasında **AFFINITY** (PREFERRED) özniteliğini ayarlayın.

Bağlantı başarısız olursa başka bir kuyruk yöneticisiyle bağlantıyı yeniden deneyin.
Bir kuyruk yöneticisi grubuna bağlanın ve bağlantı kesilirse ya da kuyruk yöneticisi başarısız olursa MQCONN ya da MQCONNX MQI çağrıyı yeniden yayınlayın.

Bağlantı başarısız olursa başka bir kuyruk yöneticisine otomatik olarak yeniden bağlanın.
MQCONNX **MQCNO** seçeneğini MQCNO_RECONNECTkullanarak bir kuyruk yöneticisi grubuna bağlanın.

Çok eşgörünümlü bir kuyruk yöneticisinin farklı bir yönetim ortamına otomatik olarak yeniden bağlanın.
Önceki örnekle aynı şekilde yapın. Bu durumda, kuyruk yöneticisi grubunu belirli bir çok eşgörünümlü kuyruk yöneticisinin yönetim ortamlarına bağlanacak şekilde sınırlamak istiyorsanız, bağlantıları olan grubu yalnızca çok eşgörünümlü kuyruk yöneticisi yönetim ortamlarıyla tanımlayın.

İstemci uygulamasından MQCONN ya da MQCONNX MQI çağrısını **QmgrName** değiştirgesinin başına yıldız işareti eklenmeden yayınlamasını da isteyebilirsiniz. Bu şekilde istemci uygulaması yalnızca adı belirtilen kuyruk yöneticisine bağlanabilir. Son olarak, **MQCNO** seçeneğini MQCNO_RECONNECT_Q_MGRolarak ayarlayabilirsiniz. Bu seçenek, önceden bağlanmış olan aynı kuyruk yöneticisine yapılan yeniden bağlantıları kabul eder. Bu değeri, yeniden bağlantıları normal bir kuyruk yöneticisinin aynı örneğiyle sınırlamak için de kullanabilirsiniz.

Bazı kuyruk yöneticilerine diğerlerinden daha fazla bağlanan istemcilerle kuyruk yöneticileri arasındaki istemci bağlantılarını dengeleme.
Bir kuyruk yöneticisi grubu tanımlayın ve bağlantıları eşit olmayan bir şekilde dağıtmak için her istemci kanalı tanımlamasında **CLNTWGHT** özniteliğini ayarlayın.

Bir bağlantı ya da kuyruk yöneticisi hatasından sonra istemcinin yeniden bağlanma yükünü eşit olmayan bir şekilde dağıt ve zaman içinde dağıt.
Önceki örnekle aynı şekilde yapın. IBM MQ MQI client , kuyruk yöneticileri arasında yeniden bağlantıları rasgele hale getirir ve yeniden bağlantıları zaman içinde yayar.

Kuyruk yöneticilerinizi herhangi bir istemci kodunu değiştirmeden taşıyın.
CCDT, istemci uygulamanızı kuyruk yöneticisinin yerinden yalıtır. CCDT, istemcide tanımlanabilen, paylaşılan bir konumdan okunabilen ya da bir web sunucusundan alınabilen bir veri dosyasıdır. Daha fazla bilgi için bkz. İstemci kanal tanımlama çizelgesi.

Kuyruk yöneticisi adlarını bilmeyen bir istemci uygulaması yazın.
Kuyruk yöneticisi grup adlarını kullanın ve kuruluşunuzdaki istemci uygulamalarınızla ilgili kuyruk yöneticisi grup adları için bir adlandırma kuralı oluşturun ve kuyruk yöneticilerinin adlandırılması yerine çözümlerinizin mimarisini yansıtır.

z/OS *Kuyruk paylaşım gruplarıyla bağlantı kuruluyor*

Uygulamanızı, bir kuyruk paylaşım grubunun parçası olan bir kuyruk yöneticisine bağlayabilirsiniz. Bunu, MQCONN ya da MQCONNX çağrısında kuyruk yöneticisi adı yerine kuyruk paylaşım grubu adı kullanılarak yapabilirsiniz.

Kuyruk paylaşım gruplarının adı en çok dört karakterdir. Ad ağınızda benzersiz olmalı ve herhangi bir kuyruk yöneticisi adından farklı olmalıdır.

İstemci kanal tanımlaması, gruptaki kullanılabilir bir kuyruk yöneticisine bağlanmak için kuyruk paylaşım grubu soysal arabirimini kullanmalıdır. Daha fazla bilgi için İstemcinin kuyruk paylaşım grubuna bağlanmasıbaşlıklı konuya bakın. Dinleyicinin bağlandığı kuyruk yöneticisinin kuyruk paylaşım grubunun bir üyesi olduğundan emin olmak için bir denetim yapılır.

Paylaşılan kuyruklara ilişkin ek bilgi için Paylaşılan kuyruklar ve kuyruk paylaşım gruplarıbaşlıklı konuya bakın.

Kanal ağırlıklandırma ve benzerliği örnekleri

Bu örnekler, sıfır olmayan `ClientChannelAğırlıkları` kullanıldığında istemci-bağlantı kanallarının nasıl seçildiğini gösterir.

`ClientChannelAğırlık` ve `ConnectionAffinity` kanal öznitelikleri, bir bağlantı için birden çok uygun kanal olduğunda istemci-bağlantı kanallarının nasıl seçileceğini denetler. Bu kanallar, daha yüksek kullanılabilirlik ve/veya iş yükü dengeleme sağlamak için farklı kuyruk yöneticilerine bağlanacak şekilde yapılandırılır. Birden çok kuyruk yöneticisinden biriyle bağlantı kurulmasına neden olabilecek MQCONN çağrıları, kuyruk yöneticisi adına şu konuda açıklandığı gibi bir yıldız işareti eklemelidir: MQCONN çağrılarına örnekler: Örnek 1. Kuyruk yöneticisi adı bir yıldız işareti (*)içeriyor.

Bir bağlantı için geçerli aday kanallar, QMNAME özniteliğinin MQCONN çağrısında belirtilen kuyruk yöneticisi adıyla eşleştiği kanallardır. Bir bağlantıya ilişkin tüm uygulanabilir kanalların `ClientChannelAğırlığı` sıfırsa (varsayılan), bunlar şu örnekteki gibi alfabetik sırayla seçilir: MQCONN çağrıları örnekleri: Örnek 1. Kuyruk yöneticisi adı bir yıldız işareti (*)içeriyor.

Aşağıdaki örnekler, sıfır olmayan `ClientChannelAğırlıkları` kullanıldığında ne olduğunu göstermektedir. Bu özellik sözde rasgele kanal seçimini içerdiğinden, örneklerin kesinlikle ne olacağını değil, gerçekleşebilecek bir işlem dizisi gösterdiğini unutmayın.

Örnek 1. ConnectionAffinity ayarı PREFERRED değerine ayarlandığında kanalları seçme

Bu örnek, IBM MQ MQI client 'nin ConnectionAffinity öğesinin PREFERRED olarak ayarlandığı bir CCDT' den bir kanalı nasıl seçtiğini gösterir.

Bu örnekte, bir dizi istemci makinesi kuyruk yöneticisi tarafından sağlanan bir İstemci Kanal Tanımlama Çizelgesi (CCDT) kullanır. CCDT, aşağıdaki özniteliklere sahip istemci bağlantı kanallarını içerir (DEFINE CHANNEL komutunun sözdizimi kullanılarak gösterilir):

```
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)
```

Uygulama MQCONN (*CORE) komutu verir

QMNAME özniteliği eşleşmediğinden Kanal A bu bağlantı için aday değil. B, C ve D kanalları adaylar olarak tanımlanır ve ağırlıklandırmalarına göre tercih sırasına göre yerleştirilir. Bu örnekte sıra C, B, D olabilir. İstemci core2.ops.company.exampleadresindeki kuyruk yöneticisine bağlanmayı dener. MQCONN çağrısı kuyruk yöneticisi adında bir yıldız işareti içerdiğinden, bu adresteki kuyruk yöneticisinin adı denetlenmiyor.

AFFINITY(PREFERRED) ile bu belirli istemci makinesinin her bağlanmasında kanalların aynı ilk tercih sırasına yerleştirileceğini göz önünde bulundurmanız önemlidir. Bu, bağlantılar farklı süreçlerden ya da farklı zamanlarda olsa da geçerlidir.

Bu örnekte, core.2.ops.company.example adresindeki kuyruk yöneticisine ulaşılamıyor. Kanal B tercih sırasına göre sırada olduğundan istemci core1.ops.company.example seçeneğine bağlanmayı dener. Buna ek olarak, C kanalı en az tercih edilen kanal olacak şekilde indirgenir.

Aynı uygulama tarafından ikinci bir MQCONN (*CORE) çağrısı yayınlandı. Kanal C önceki bağlantı tarafından indirgendi, bu nedenle en çok tercih edilen kanal şimdi B. Bu bağlantı core1.ops.company.exampleile yapılır.

Aynı İstemci Kanal Tanımlama Tablosunu paylaşan ikinci bir makine, kanalları farklı bir ilk tercih sırasına yerleştirir. Örneğin, D, B, C. Normal koşullarda, tüm kanallar çalışırken, bu makinedeki uygulamalar core3.ops.company.example 'a, ilk makinedeki uygulamalar core2.ops.company.example' a bağlanır. Bu, her bir istemcinin aynı kuyruk yöneticisine (varsa) bağlanmasına izin verirken, birden çok kuyruk yöneticisinde çok sayıda istemcinin iş yükü dengelemesini sağlar.

Örnek 2. ConnectionAffinity seçeneği NONE olarak ayarlandığında kanalları seçme
Bu örnek, IBM MQ MQI client 'in ConnectionAffinity öğesinin NONE olarak ayarlandığı bir CCDT' den bir kanalı nasıl seçtiğini gösterir.

Bu örnekte, bir dizi istemci kuyruk yöneticisi tarafından sağlanan bir İstemci Kanal Tanımlama Çizelgesi (CCDT) kullanır. CCDT, aşağıdaki özniteliklere sahip istemci bağlantı kanallarını içerir (DEFINE CHANNEL komutunun sözdizimi kullanılarak gösterilir):

```
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)
```

Uygulama MQCONN (*CORE) komutunu verir. Önceki örnekte olduğu gibi, QMNAME eşleşmediği için A kanalı dikkate alınmaz. Kanal B, C ya da D, ağırlıklandırmalarına göre seçilir ve %50, %30 ya da %20 olasılıkları vardır. Bu örnekte, B kanalı seçilebilir. Kalıcı bir tercih sırası oluşturulmadı.

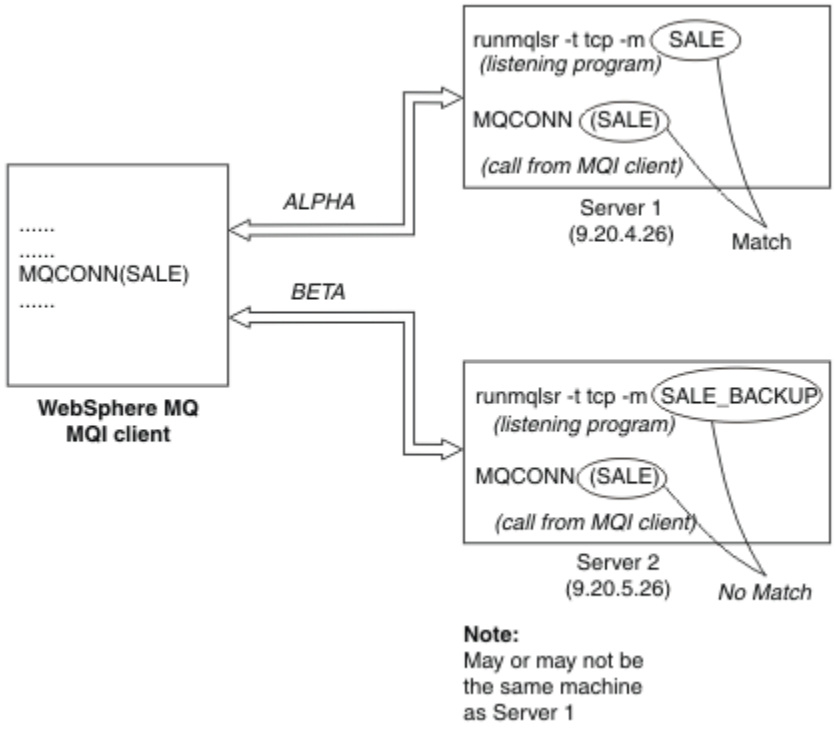
İkinci bir MQCONN (*CORE) çağrısı yapılır. Yine, aynı olasılıklara sahip üç geçerli kanaldan biri seçilir. Bu örnekte, C kanalı seçilir. Ancak core2.ops.company.example yanıt vermediği için, geri kalan aday kanallar arasında başka bir seçim yapılır. Kanal B seçildi ve uygulama core1.ops.company.exampleseçeneğine bağlandı.

AFFINITY (NONE) ile, her MQCONN çağrısı diğerlerinden bağımsızdır. Bu nedenle, bu örnek uygulama üçüncü bir MQCONN (*CORE) yaptığında, B ya da D ' den birini seçmeden önce bozuk kanal C üzerinden bir kez daha bağlanma girişiminde bulunabilir.

MQCONN çağrılarına örnekler
Belirli bir kuyruk yöneticisine ya da bir kuyruk yöneticisi grubuna bağlanmak için MQCONN kullanımı örnekleri.

Aşağıdaki örneklerin her birinde ağ aynıdır; aynı IBM MQ MQI client' dan iki sunucuya tanımlanan bir bağlantı vardır. (Bu örneklerde, MQCONNX çağrısı MQCONN çağrısı yerine kullanılabilir.)

Biri SALE , diğeri SALE_BACKUPadlı sunucu makinelerinde çalışan iki kuyruk yöneticisi vardır.



Şekil 98. MQCONN örneği

Bu örneklerdeki kanallara ilişkin tanımlar şunlardır:

Satış tanımları:

```
DEFINE CHANNEL(ALPHA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to IBM MQ MQI client')

DEFINE CHANNEL(ALPHA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.4.26) DESCR('IBM MQ MQI client connection to server 1') +
QMNAME(SALE)

DEFINE CHANNEL(BETA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +
CONNAME(9.20.5.26) DESCR('IBM MQ MQI client connection to server 2') +
QMNAME(SALE)
```

SALE_BACKUP tanımlaması:

```
DEFINE CHANNEL(BETA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +
DESCR('Server connection to IBM MQ MQI client')
```

İstemci kanalı tanımları aşağıdaki gibi özetlenebilir:

Ad	CHLTYPE	İZLEMA TIPI	KONADı	QMNAME
Alfa	CLNTCONN	TCP	9.20.4.26	SATILIK
BETA	CLNTCONN	TCP	9.20.5.26	SATILIK

MQCONN örneklerinin gösterdiği
Bu örnekler, birden çok kuyruk yöneticisinin yedekleme sistemi olarak kullanılmasını gösterir.

Sunucu 1 'e olan iletişim bağlantısının geçici olarak bozuk olduğunu varsayın. Yedekleme sistemi olarak birden çok kuyruk yöneticisinin kullanılması gösterilmiştir.

Her örnek, farklı bir MQCONN çağrısının kapsamasını ve aşağıdaki kuralları uygulayarak, sunulan örnekte ne olduğuna ilişkin bir açıklama sağlar:

1. İstemci kanal tanımlama çizelgesi (CCDT), MQCONN çağrısında verilenin karşılığı olan bir kuyruk yöneticisi adı (QMNAME alanı) için alfabetik kanal adı sırasıyla taranır.
2. Bir eşleşme bulunursa, kanal tanımı kullanılır.
3. Bağlantı adı (CONNAME) ile tanıtılan makineye kanalı başlatma girişiminde bulunuldu. Bu başarılı olursa, uygulama devam eder. Aşağıdakileri gerektirir:
   - Sunucuda çalıştırılacak bir dinleyici.
   - İstemcinin bağlanmak istediği (belirtildiyse) kuyruk yöneticisine bağlanacak dinleyici.
4. Kanalı başlatma girişimi başarısız olursa ve istemci kanal tanımlama çizelgesinde birden çok giriş varsa (bu örnekte iki giriş vardır), dosya daha fazla eşleşme için aranır. Bir eşleşme bulunursa, işlem 1. adımda devam eder.
5. Eşleşme bulunamazsa ya da istemci kanal tanımlama çizelgesinde başka giriş yoksa ve kanal başlatılamadıysa, uygulama bağlanamıyor. MQCONN çağrısında uygun bir neden kodu ve tamamlanma kodu döndürüldü. Uygulama, döndürülen neden ve tamamlanma kodlarına dayalı olarak işlem yapabilir.

Örnek 1. Kuyruk yöneticisi adı yıldız işareti (*) içeriyor
Bu örnekte uygulama, hangi kuyruk yöneticisine bağlandığı konusunda endişeli değildir. Uygulama, yıldız işareti de içinde olmak üzere bir kuyruk yöneticisi adı için MQCONN çağrısı yayınlar. Uygun bir kanal seçilir.

Uygulama sorunları:

```
MQCONN (*SALE)
```

Kurallara uyulması, bu durumda şöyle olur:

1. İstemci kanal tanımlama çizelgesi (CCDT), uygulama MQCONN çağrısıyla eşleşen SALEkuyruk yöneticisi adı için taranır.
2. ALPHA ve BETA için kanal tanımları bulunur.
3. Bir kanalın CLNTWGHT değeri 0 ise, bu kanal seçilir. Her ikisi de 0 CLNTWGHT değerine sahipse, ALPHA kanalı alfabetik sırada birinci olduğu için seçilir. Her iki kanal da sıfır olmayan bir CLNTWGHT değerine sahipse, ağırlıklandırmasına göre bir kanal rasgele seçilir.
4. Kanalı başlatma girişiminde bulunuldu.
5. Kanal BETA seçildiyse, başlatma girişimi başarılı olur.
6. ALPHA kanalı seçildiyse, iletişim bağlantısı kesildiği için kanal başlatma girişimi başarılı DEĞİLDİR. Daha sonra aşağıdaki adımlar uygulanır:
   a. Kuyruk yöneticisi adı SALE için diğer tek kanal şudur: BETA.
   b. Bu kanalı başlatma girişiminde bulunuldu-bu başarılı oldu.
7. Bir dinleyicinin çalıştığını gösteren bir denetim, çalışan bir dinleyici olduğunu gösterir. SALE kuyruk yöneticisine bağlı değil, ancak MQI çağrı değiştirgesinin içinde bir yıldız işareti (*) olduğu için denetim yapılmadı. Uygulama SALE_BACKUP kuyruk yöneticisine bağlıdır ve işlemeye devam eder.

Örnek 2. Kuyruk yöneticisi adı belirtildi
Bu örnekte uygulama belirli bir kuyruk yöneticisine bağlanmalıdır. Uygulama, o kuyruk yöneticisi adı için bir MQCONN çağrısı yayınlar. Uygun bir kanal seçilir.

Uygulama, MQI çağrısında görüldüğü gibi SALEadlı belirli bir kuyruk yöneticisine bağlantı gerektirir:

```
MQCONN (SALE)
```

Kurallara uyulması, bu durumda şöyle olur:

1. İstemci kanal tanımlama çizelgesi (CCDT), uygulama MQCONN çağrısıyla eşleşen SALEkuyruk yöneticisi adı için alfabetik kanal adı sırasıyla taranır.
2. Eşleşen ilk kanal tanımı: ALPHA.
3. Kanalı başlatma girişiminde bulunuldu-iletişim bağlantısı koptuğundan bu başarısız oldu.
4. İstemci kanal tanımlama çizelgesi SALE kuyruk yöneticisi adı için yeniden taranır ve BETA kanal adı bulunur.
5. Kanalı başlatma girişiminde bulunuldu-bu başarılı oldu.
6. Bir dinleyicinin çalışıp çalışmadığını görme denetimi, çalışan bir dinleyici olduğunu, ancak bunun SALE kuyruk yöneticisine bağlı olmadığını gösterir.
7. İstemci kanal tanımlama çizelgesinde başka giriş yok. Uygulama devam edemiyor ve MQRC_Q_MGR_NOT_VAR dönüş kodunu alıyor.

Örnek 3. Kuyruk yöneticisi adı boş ya da yıldız işareti (*)
Bu örnekte uygulama, hangi kuyruk yöneticisine bağlandığı konusunda endişeli değildir. Uygulama, boş bir kuyruk yöneticisi adı ya da yıldız işareti belirterek bir MQCONN verir. Uygun bir kanal seçilir.

Bu, “Örnek 1. Kuyruk yöneticisi adı yıldız işareti (*) içeriyor” sayfa 891ile aynı şekilde işlenir.

Not: Bu uygulama IBM MQ MQI clientdışında bir ortamda çalışıyorsa ve ad boşsa, varsayılan kuyruk yöneticisine bağlanmayı dener. Bir istemci ortamından çalıştırıldığında bu durum söz konusu değildir; erişilen kuyruk yöneticisi, kanalın bağlandığı dinleyiciyle ilişkilendirilmiş olandır.

Uygulama sorunları:

```
MQCONN ("")
```

veya

```
MQCONN (*)
```

Kurallara uyulması, bu durumda şöyle olur:

1. İstemci kanal tanımlama çizelgesi (CCDT), uygulama MQCONN çağrısıyla eşleşen boş bir kuyruk yöneticisi adı için alfabetik kanal adı sırasında taranır.
2. The entry for the channel name ALPHA has a queue manager name in the definition of SALE. Bu, kuyruk yöneticisi adının boş olmasını gerektiren MQCONN çağrı parametresiyle eşleşmiyor.
3. Sonraki girdi, BETAkanal adı içindir.
4. Tanımdaki `queue manager name` , SALE' dir. Bu, kuyruk yöneticisi adının boş olmasını gerektiren MQCONN çağrı parametresiyle bir kez daha eşleşmiyor.
5. İstemci kanal tanımlama çizelgesinde başka giriş yok. Uygulama devam edemiyor ve MQRC_Q_MGR_NOT_VAR dönüş kodunu alıyor.

İstemci ortamında tetikleme

IBM MQ MQI clients üzerinde çalışan IBM MQ uygulamaları tarafından gönderilen iletiler, diğer iletilerle aynı şekilde tetiklenmeye katkıda bulunur ve hem sunucuda hem de istemcide programları tetiklemek için kullanılabilir.

Tetikleme, Tetikleme kanallarındaayrıntılı olarak açıklanır.

Tetikleyici izleme programı ve başlatılacak uygulama aynı sistemde olmalıdır.

Tetiklenen kuyruğun varsayılan özellikleri, sunucu ortamındaki özelliklerle aynıdır. Özellikle, bir istemci uygulamasında iletileri bir z/OS kuyruk yöneticisi için yerel olan tetiklenmiş bir kuyruğa koyan bir MQPMO eşitleme noktası denetimi seçeneği belirtilmezse, iletiler bir iş birimine yerleştirilir. Tetikleme koşulu yerine getirilirse, tetikleyici iletisi aynı iş birimi içindeki başlatma kuyruğuna konur ve iş birimi sona erinceye kadar tetikleyici izleme programı tarafından alınamaz. Tetiklenecek süreç, iş birimi sona erinceye kadar başlatılamaz.

Süreç tanımlaması
Tetikleme ayarı açık olan kuyrukla ilişkilendirilmiş olduğundan, süreç tanımlamasını sunucuda tanımlamanız gerekir.

Süreç nesnesi, neyin tetikleneceğini tanımlar. İstemci ve sunucu aynı altyapıda çalışmıyorsa, tetikleyici izleme programı tarafından başlatılan her işlem *ApplType*değerini tanımlamalıdır; tersi durumda, sunucu varsayılan tanımlamalarını (sunucu makinesiyle olağan olarak ilişkilendirilen uygulama tipi) alır ve bir hataya neden olur.

Örneğin, tetikleyici bir IBM MQ MQI client üzerinde çalışıyorsa ve başka bir işletim sistemindeki bir sunucuya istek göndermek istiyorsa, MQAT_WINDOWS_NT tanımlanmalıdır; tersi durumda, diğer işletim sistemi varsayılan tanımlamalarını kullanır ve işlem başarısız olur.

Tetikleyici

z/OS IBM MQ dışı ürünler tarafından sağlanan tetikleyici izleyicisi, IBM i IBM i, AIX, Linux, and Windows sistemleri için istemci ortamlarında çalışır.

Tetikleyici izleme programını çalıştırmak için aşağıdaki komutlardan birini verin:

- IBM i IBM i'ta:

```
CALL PGM(QMQM/RUNMQTMC) PARM('-m' QmgrName '-q' InitQ)
```

- ALW AIX, Linux, and Windows platformlarında:

```
runmqtmc [-m QMgrName] [-q InitQ]
```

Varsayılan başlatma kuyruğu SYSTEM.DEFAULT.INITIATION.QUEUE QUEUE değerini belirleyin. Başlatma kuyruğu, tetikleyici izleme programının tetikleyici iletileri aradığı yerdir. Daha sonra, uygun tetikleyici iletileri için programları çağırır. Bu tetikleyici izleme programı varsayılan uygulama tipini destekler ve istemci kitaplıklarını birbirine bağladığı dışında `runmqtrm` ile aynıdır.

Tetikleyici izleme programı tarafından oluşturulan komut dizilimi aşağıdaki gibidir:

1. İlgili süreç tanımlamasındaki *ApplicId* . *ApplicId* , komut satırına girileceği gibi, çalıştırılacak programın adıdır.
2. Başlatma kuyruğundan alınan, tırnak işareti içine alınmış MQTMC2 yapısı. Sistem komutunun bu dizgiyi tek bir parametre olarak kabul etmesi için, tam olarak sağlandığı gibi, tırnak işareti içinde bu dizgiyi içeren bir komut dizgisi başlatılır.
3. İlgili süreç tanımlamasındaki *EnvrData* .

Tetikleyici izleme programı, başlattığı uygulama tamamlanıncaya kadar, başlatma kuyruğunda başka bir ileti olup olmadığını görmez. Uygulamanın yapacak çok işi varsa, tetikleyici izleme programı gelen tetikleyici iletilerinin sayısına ayak uydurmayabilir. Bu durumla başa çıkmanın iki yolu vardır:

1. Çalışan daha fazla tetikleyici izleme programı var

   Daha fazla tetikleyici izleme programı çalıştırmayı seçerseniz, bir kerede çalışabilecek uygulama sayısı üst sınırını denetleyebilirsiniz.

2. Başlatılan uygulamaları arka planda çalıştır

   Uygulamaları arka planda çalıştırmayı seçerseniz, IBM MQ çalışabilecek uygulama sayısı üzerinde herhangi bir kısıtlama uygulamaz.

Başlatılan uygulamayı AIX and Linux sistemlerinde artalanda çalıştırmak için, süreç tanımlamasının *EnvrData* sonuna bir & (ve işareti) koymanız gerekir.

CICS uygulamaları (z/OSdışı)
Bir MQCONN ya da MQCONNX çağrısı verenz/OS CICS dışı bir uygulama programı, CEDA ' ya RESIDENT olarak tanımlanmalıdır. Bir CICS sunucu uygulamasını istemci olarak yeniden bağlantılandırırsanız, eşitleme noktası desteğini kaybetme riski taşıyabilirsiniz.

Bir MQCONN ya da MQCONNX çağrısı verenz/OS CICS dışı bir uygulama programı, CEDA ' ya RESIDENT olarak tanımlanmalıdır. Yerleşik kodu mümkün olduğunca küçük yapmak için, MQCONN ya da MQCONNX çağrılarını yayınlamak üzere ayrı bir programa bağlanabilirsiniz.

İstemci bağlantısını tanımlamak için MQSERVER ortam değişkeni kullanılıyorsa, CICSENV.CMD dosyası.

IBM MQ uygulamaları, kodu değiştirmeden bir IBM MQ sunucu ortamında ya da bir IBM MQ istemcisinde çalıştırılabilir. Ancak, bir IBM MQ sunucu ortamında CICS , eşitleme noktası koordinatörü olarak işlev görür ve siz **MQCMIT** ve **MQBACK**yerine EXEC CICS SYNCPOINT ve EXEC CICS SYNCPOINT ROLLBACK komutunu kullanırsınız. Bir CICS uygulaması istemci olarak yeniden bağlantılandırılırsa, eşitleme noktası desteği kaybolur. Bir IBM MQ MQI clientüzerinde çalışan uygulama için **MQCMIT** ve **MQBACK** kullanılmalıdır.

ALW CICS ve Tuxedo uygulamalarının hazırlanması ve çalıştırılması

CICS ve Tuxedo uygulamalarını istemci uygulamaları olarak çalıştırmak için, sunucu uygulamalarıyla kullananlardan farklı kitaplıklar kullanırsınız. Uygulamanın çalıştırıldığı kullanıcı kimliği de farklı.

CICS ve Tuxedo uygulamalarını IBM MQ MQI client uygulamaları olarak çalışacak şekilde hazırlamak için Genişletilmiş bir işlem istemcisinin yapılandırılmasıbaşlıklı konudaki yönergeleri izleyin.

Ancak, özellikle CICS ve Tuxedo uygulamalarının (IBM MQile birlikte sağlanan örnek programlar da içinde olmak üzere) hazırlanmasıyla ilgili bilgilerin, uygulamaları bir IBM MQ sunucu sisteminde çalışacak şekilde hazırladığınızı varsaydığını unutmayın. Sonuç olarak, bilgiler yalnızca sunucu sisteminde kullanılması amaçlanan IBM MQ kitaplıklarını gösterir. İstemci uygulamalarınızı hazırlarken aşağıdaki işlemleri gerçekleştirmeniz gerekir:

- Uygulamanızın kullandığı dil bağ tanımları için uygun istemci sistem kitaplığını kullanın. Örneğin:
  - Linux AIX AIX and Linuxüzerinde C dilinde yazılan uygulamalar için libmqm yerine libmqic kitaplığını kullanın.
  - Windows Windows sistemlerinde, mqm.libyerine mqic.lib kitaplığını kullanın.
- Çizelge 134 sayfa 894 ve Çizelge 135 sayfa 894içinde gösterilen sunucu sistem kitaplıkları yerine, eşdeğer istemci sistem kitaplıklarını kullanın. Bu çizelgelerde bir sunucu sistem kitaplığı listelenmiyorsa, istemci sisteminde aynı kitaplığı kullanın.

Çizelge 134. AIX and Linux üzerinde istemci sistemi kitaplıkları

IBM MQ sunucu sistemine ilişkin kitaplık	**IBM MQ istemci sisteminde kullanılacak eşdeğer kitaplık**
libmqmxa	libmqcxa

Çizelge 135. Windows sistemlerinde istemci sistem kitaplıkları

IBM MQ sunucu sistemine ilişkin kitaplık	**IBM MQ istemci sisteminde kullanılacak eşdeğer kitaplık**
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib
mqmenc.lib	mqcxa.lib
mqmcics4.lib	mqccics4.lib

Bir istemci uygulaması tarafından kullanılan kullanıcı kimliği

Bir IBM MQ sunucu uygulamasını CICSaltında çalıştırdığınızda, normalde CICS kullanıcısından hareketin kullanıcı kimliğine geçer. Ancak, CICSaltında bir IBM MQ MQI client uygulamasını çalıştırdığınızda, bu uygulama CICS ayrıcalıklı yetkisini korur.

ALW *CICS ve Tuxedo örnek programları*

AIX, Linux, and Windows sistemlerinde kullanılmak üzere CICS ve Tuxedo örnek programları.

Çizelge 136 sayfa 895 içinde, AIX and Linux istemci sistemlerinde kullanılmak üzere sağlanan CICS ve Tuxedo örnek programları listelenir. Çizelge 137 sayfa 895 içinde Windows istemci sistemlerine ilişkin eşdeğer bilgiler listelenir. Çizelgelerde, programları hazırlamak ve çalıştırmak için kullanılan kütükler de listelenir. Örnek programlara ilişkin açıklamalar için bkz. “CICS hareket örneği” sayfa 1034 ve “AIX, Linux, and Windows üzerinde TUXEDO örneklerinin kullanılması” sayfa 1077.

Çizelge 136. AIX and Linux istemci sistemleri için örnek programlar

Açıklama	Kaynak	Yürütülebilir modül
CICS PROGRAM	amqscic0.ccs	amqscicc
CICS programına ilişkin üstbilgi dosyası	amqscih0.h	-
İletileri koymak için tuxedo istemci programı	amqstxpx.c	-
İletileri almak için tuxedo istemci programı	amqstxgx.c	-
İki istemci programı için tuxedo sunucu programı	amqstxsx.c	-
Smokin programları için UBBCONFIG dosyası	ubbstxcx.cfg	-
Tuxedo programlarına ilişkin alan tablosu dosyası	amqstxvx.flds	-
Tuxedo programlarına ilişkin tanımlama dosyasını görüntüle	amqstxvx.v	-

Çizelge 137. Windows istemci sistemleri için örnek programlar

Açıklama	Kaynak	Yürütülebilir modül
CICS Hareket	amqscic0.ccs	amqscicc
CICS işlemine ilişkin üstbilgi dosyası	amqscih0.h	-
İletileri koymak için tuxedo istemci programı	amqstxpx.c	-
İletileri almak için tuxedo istemci programı	amqstxgx.c	-
İki istemci programı için tuxedo sunucu programı	amqstxsx.c	-
Smokin programları için UBBCONFIG dosyası	ubbstxcx.cfg	-
Tuxedo programlarına ilişkin alan tablosu dosyası	amqstxvx.fld	-
Tuxedo programlarına ilişkin tanımlama dosyasını görüntüle	amqstxvx.v	-
Smokin programları için makefile	amqstxmc.mak	-
Tuxedo programlarına ilişkin ENVFILE dosyası	amqstxen.env	-

ALW *CICS ve Tuxedo uygulamaları için değiştirildiği şekliyle AMQ5203hata iletisi*

Genişletilmiş bir hareket istemcisi kullanan CICS ya da Tuxedo uygulamalarını çalıştırdığınızda, standart tanılama iletilerini görebilirsiniz. Bunlardan biri, genişletilmiş bir işlem istemcisiyle kullanılmak üzere değiştirildi

IBM MQ hata günlüğü dosyalarında görebileceğiniz iletiler Tanılama iletileri: AMQ4000-9999. AMQ5203 iletisi, genişletilmiş bir hareket istemcisiyle kullanılmak üzere değiştirildi. Değiştirilen iletinin metni:

AMQ5203: XA arabirimi çağrılırken bir hata oluştu.

Açıklama
Hata numarası & 2; burada 1 değeri, sağlanan & 1 işaret değerinin geçersiz olduğunu, 2 değeri, aynı işlemde iş parçacıklı ve iş parçacıklı olmayan kitaplıkları kullanma girişiminde bulunulduğunu, 3 değeri, sağlanan '& 3' kuyruk yöneticisi adında bir hata olduğunu, 4 değeri & 1 kaynak yöneticisi tanıtıcısının geçersiz olduğunu, 5 değeri, çağrılan ikinci bir kuyruk yöneticisini kullanma girişiminde bulunulduğunu gösterir '& 3 'başka bir kuyruk yöneticisi bağlandığında, 6, uygulama bir kuyruk yöneticisine bağlı olmadığında Hareket Yöneticisi 'nin çağrıldığını, 7, başka bir çağrı devam ederken XA çağrıldığını, 8, xa_open çağrısındaki' & 4 'xa_info dizgisinin' & 5 'parametre adı için geçersiz bir parametre değeri içerdiğini ve 9, xa_open çağrısındaki' & 4 'xa_info dizgisinin gerekli bir parametre olduğunu gösterir, parametre adı' & 5 '.

Kullanıcı yanıtı
Hatayı düzeltip işlemi yeniden deneyin.

Windows Microsoft Transaction Server uygulamalarının hazırlanması ve çalıştırılması

Bir MTS uygulamasını IBM MQ MQI client uygulaması olarak çalışacak şekilde hazırlamak için, ortamınız için uygun olan bu yönergeleri izleyin.

IBM MQ kaynaklarına erişen Microsoft Transaction Server (MTS) uygulamalarının geliştirilmesine ilişkin genel bilgi için IBM MQ Help Center olanağındaki MTS bölümüne bakın.

Bir MTS uygulamasını IBM MQ MQI client uygulaması olarak çalışacak şekilde hazırlamak için uygulamanın her bileşeni için aşağıdakilerden birini yapın:

- Bileşen MQI için C dili bağ tanımlarını kullanıyorsa, “Windows ' da C programlarını hazırlama” sayfa 973 içindeki yönergeleri izleyin, ancak bileşeni mqic.libyerine mqicxa.lib kitaplığına bağlayın.
- Bileşen IBM MQ C++ sınıflarını kullanıyorsa, “Windows üzerinde C++ programları oluşturma” sayfa 528 içindeki yönergeleri izleyin, ancak bileşeni imqc23vn.libyerine imqx23vn.lib kitaplığına bağlayın.
- Bileşen MQI için Visual Basic dil bağ tanımlarını kullanıyorsa, “Windows ' da Visual Basic programlarının hazırlanması” sayfa 977 içindeki yönergeleri izleyin, ancak Visual Basic projesini tanımladığınızda **Koşullu Derleme Bağımsız Değişkenleri** alanına `MqType=3` yazın.

IBM MQ JMS uygulamalarının hazırlanması ve çalıştırılması

Hareket yöneticiniz olarak WebSphere Application Server ile IBM MQ JMS uygulamalarını istemci kipinde çalıştırabilirsiniz. Belirli uyarı iletilerini görebilirsiniz.

IBM MQ JMS uygulamalarını istemci kipinde WebSphere Application Server hareket yöneticiniz olarak hazırlamak ve çalıştırmak için “IBM MQ classes for JMS/Jakarta Messaging ' yi kullanma” sayfa 78içindeki yönergeleri izleyin.

Bir IBM MQ JMS istemci uygulamasını çalıştırdığınızda aşağıdaki uyarı iletilerini görebilirsiniz:

MQJE080
Yetersiz lisans birimleri-setmqcap komutunu çalıştırın

MQJE081
Lisans birimi bilgilerini içeren dosya yanlış biçimde-setmqcap çalıştırılıyor

MQJE082
Lisans birimi bilgilerini içeren dosya bulunamadı-setmqcap çalıştırılır

Kullanıcı çıkışları, API çıkışları ve IBM MQ kurulabilir hizmetleri

Bu konuda, bu programların kullanılmasına ve geliştirilmesine ilişkin bilgiler yer alır.

Kuyruk yöneticisi olanaklarını genişletmek için kullanıcı çıkışlarını, API çıkışlarını ve kurulabilir hizmetleri nasıl kullanabileceğinize ilişkin bilgi için Kuyruk yöneticisi olanaklarını genişletmebaşlıklı konuya bakın.

Çıkışların ve kurulabilir hizmetlerin yazılması ve derlenmesiyle ilgili bilgi için alt konulara bakın.

İlgili kavramlar

MQI kanalları için kanal çıkış programları

İlgili başvurular

API çıkış başvurusu

Kurulabilir hizmetler arabirimi başvuru bilgileri

IBM i IBM i üzerinde kurulabilir hizmetler arabirimi başvuru bilgileri

ALW AIX, Linux, and Windows üzerinde çıkışlar ve kurulabilir hizmetler yazılıyor

AIX, Linux, and Windowsüzerindeki herhangi bir IBM MQ kitaplığını bağlamadan çıkışları yazabilir ve derleyebilirsiniz.

Bu görev hakkında

Bu konu yalnızca AIX, Linux, and Windows sistemleri için geçerlidir. Diğer platformlara ilişkin çıkışların ve kurulabilir hizmetlerin yazılmasıyla ilgili ayrıntılar için ilgili platforma özgü konulara bakın.

IBM MQ varsayılan olmayan bir yere kurulduysa, herhangi bir IBM MQ kitaplığını bağlamadan çıkışlarınızı yazmanız ve derlemeniz gerekir.

Bu IBM MQ kitaplıklarını bağlamadan AIX, Linux, and Windows sistemlerinde çıkışları yazabilir ve derleyebilirsiniz:

- mqmzf
- mqm
- mqmvx
- mqmvxd
- mqic
- mqutl

Bu kitaplıklara bağlı var olan çıkışlar çalışmaya devam eder; bu, AIX and Linux sistemler IBM MQ üzerinde varsayılan yerin kurulu olması koşuluyla.

Yordam

1. cmqec.h üstbilgi dosyasını ekleyin.

   Bu üstbilgi dosyası otomatik olarak cmqc.h, cmqxc.h ve cmqzc.h üstbilgi dosyalarını içerir.
2. MQI ve DCI çağrılarının MQIEP yapısı aracılığıyla yapılabilmesi için çıkışı yazın. MQIEP yapısıyla ilgili ek bilgi için MQIEP yapısıbaşlıklı konuya bakın.
   - Kurulabilir hizmetler
     - MQZEP çağrısına işaret etmek için **Hconfig** değiştirgesini kullanın.
     - **Hconfig** değiştirgesini kullanmadan önce, **Hconfig** değerinin ilk 4 byte 'ının MQIEP yapısının **StrucId** ile eşleştiğini denetlemeniz gerekir.
     - Kurulabilir hizmet bileşenlerinin yazılması hakkında daha fazla bilgi için bkz. MQIEP.
   - API çıkışları
     - MQXEP çağrısına işaret etmek için **Hconfig** değiştirgesini kullanın.
     - **Hconfig** değiştirgesini kullanmadan önce, **Hconfig** değerinin ilk 4 byte 'ının MQIEP yapısının **StrucId** ile eşleştiğini denetlemeniz gerekir.

  - API çıkışlarını yazma hakkında daha fazla bilgi için bkz. “API çıkışları yazılıyor” sayfa 913.
- Kanal çıkışları
  - MQI ve DCI çağrılarını göstermek için MQCXP yapısının **pEntryPoints** değiştirgesini kullanın.
  - **pEntryPoints**komutunu kullanmadan önce MQCXP sürüm numarasının sürüm 8 'de ya da daha sonraki bir sürümde olup olmadığını denetlemeniz gerekir.
  - Kanal çıkışlarının yazılması hakkında daha fazla bilgi için bkz. “Kanal çıkış programları yazılıyor” sayfa 923.
- Veri dönüştürme çıkışları
  - MQI ve DCI çağrılarını göstermek için MQDXP yapısının **pEntryPoints** değiştirgesini kullanın.
  - **pEntryPoints**komutunu kullanmadan önce MQDXP sürüm numarasının sürüm 2 'de ya da daha sonraki bir sürümde olup olmadığını denetlemeniz gerekir.
  - **pEntryPoints** parametresini kullanan veri dönüştürme kodu oluşturmak için **crtmqcvx** komutunu ve amqsvfc0.c kaynak dosyasını kullanabilirsiniz. Bkz. “IBM MQ for Windows için veri dönüştürme çıkışı yazılması” sayfa 948 ve “IBM MQ for AIX or Linux sistemleri için veri dönüştürme çıkışı yazılması” sayfa 946.
  - **crtmqcvx** komutu kullanılarak oluşturulan veri dönüştürme çıkışları varsa, güncellenen komutu kullanarak çıkışı yeniden oluşturmanız gerekir.
  - Veri dönüştürme çıkışları yazma hakkında daha fazla bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941.
- Ön bağlantı çıkışları
  - MQI ve DCI çağrılarını göstermek için MQNXP yapısının **pEntryPoints** değiştirgesini kullanın.
  - **pEntryPoints**komutunu kullanmadan önce MQNXP sürüm numarasının sürüm 2 'de ya da daha sonraki bir sürümde olup olmadığını denetlemeniz gerekir.
  - Bağlantı öncesi çıkışları yazma hakkında daha fazla bilgi için bkz. “Havuzdan ön bağlantı çıkışı kullanarak bağlantı tanımlamalarına gönderme yapma” sayfa 950.
- Çıkışları yayınla
  - MQI ve DCI çağrılarını göstermek için MQPSXP yapısının **pEntryPoints** değiştirgesini kullanın.
  - **pEntryPoints**komutunu kullanmadan önce MQPSXP sürüm numarasının sürüm 2 ya da sonraki bir sürümde olup olmadığını denetlemeniz gerekir.
  - Yayınlama çıkışları yazma hakkında daha fazla bilgi için bkz. “Yayınlama çıkışları yazılıyor ve derleniyor” sayfa 952.
- Küme iş yükü çıkışları
  - MQXCLWLN çağrılarını göstermek için MQWXP yapısının **pEntryPoints** değiştirgesini kullanın.
  - **pEntryPoints**komutunu kullanmadan önce MQWXP sürüm numarasının 4 ya da daha yüksek bir sürümde olup olmadığını denetlemeniz gerekir.
  - Küme iş yükü çıkışları yazma hakkında daha fazla bilgi için bkz. “Küme iş yükü çıkışlarının yazılması ve derlenmesi” sayfa 954.

Örneğin, MQPUT ' yi çağıran bir kanal çıkışında:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,
                                                Hobj,
                                                &md,
                                                &pmo,
                                                messlen,
                                                buffer,
                                                &CompCode,
                                                &Reason);
```

“IBM MQ örnek yordam programlarının kullanılması” sayfa 1013içinde daha fazla örnek görülebilir.

3. Çıkışı derle:

- IBM MQ kitaplıklarına bağlantı vermeyin.
- Çıkışınızdaki IBM MQ kitaplıklarına gömülü bir RPath eklemeyin.
- Çıkışınızı derlemeyle ilgili daha fazla bilgi için aşağıdaki konulardan birine bakın:
  - API çıkışları: “API çıkışları derleniyor” sayfa 915.
  - Kanal çıkışları, yayınlama çıkışları, Küme iş yükü çıkışları: “AIX, Linux, and Windows sistemlerinde kanal çıkış programlarının derlenmesi” sayfa 940.
  - Veri dönüştürme işlemlerinden çıkılıyor: “Veri dönüştürme çıkışları yazılıyor” sayfa 941.

4. Çıkışı aşağıdaki yerlerden birine koyun:
   - Çıkışı yapılandırırken tam olarak nitelenmeyi seçtiğiniz bir yol
   - Belirli bir kuruluş dizinindeki varsayılan çıkış yolu. Örneğin, *MQ_DATA_PATH*/exits/installation2.
   - Varsayılan çıkış yolu

     Varsayılan çıkış yolu, 32 bit çıkışlar için *MQ_DATA_PATH*/exits ve 64 bit çıkışlar için *MQ_DATA_PATH*/exits64 ' dir. Bu yolları qm.ini ya da mqclient.ini dosyasında değiştirebilirsiniz. Daha fazla bilgi için bkz. Çıkış yolu. Windows ve Linuxsistemlerinde, yolu değiştirmek için IBM MQ Gezgini 'ni kullanabilirsiniz:

     a. Kuyruk yöneticisi adını sağ tıklatın

     b. **Özellikler ...** düğmesini tıklatın.

     c. **Çıkışları** seçeneğini tıklatın.

     d. Çıkış varsayılan yolu alanında, çıkış programını tutan dizinin yol adını belirtin.

   Çıkış hem belirli bir kuruluş dizinine, hem de varsayılan yol dizinine yerleştirilirse, yolda adı belirtilen IBM MQ kuruluşu tarafından belirli bir kuruluş dizini çıkışı kullanılır. Örneğin, çıkış /exits/installation2 ve /exitsiçine yerleştirilir, ancak /exits/installation1içine yerleştirilmez. IBM MQ kuruluş installation2 , /exits/installation2' den çıkışı kullanır. IBM MQ kuruluş installation1 , /exits dizinindeki çıkışı kullanır.
5. Gerekiyorsa, çıkışı yapılandırın:
   - Kurulabilir hizmetler: “Hizmetlerin ve bileşenlerin yapılandırılması” sayfa 907.
   - API çıkışları: “API çıkışlarını yapılandırma” sayfa 918.
   - Kanal çıkışları: “Kanal çıkışlarının yapılandırılması” sayfa 941.
   - Çıkışları yayınla: “Yayınlama çıkışlarının yapılandırılması” sayfa 953.
   - Bağlantı öncesi çıkışlar: İstemci yapılanış kütüğünün PreConnect kısmı.

ALW *API çıkışları bir MQI kitaplığıyla bağlantılı değil*

Belirli koşullar altında, MQIEP işlev işaretçileriyle IBM MQ API kitaplığını kullanmak için yeniden kodlanamayan var olan API çıkışınızı bağlamanız gerekir.

Var olan API çıkışınızın, sisteminizin çalıştırma zamanı bağlayıcısı tarafından, işlev göstergeleri yüklü olmayan programlara başarıyla yüklenebilmesi için bu gereklidir.

Not: Bu bilgiler, MQI çağrılarını doğrudan yapan var olan API çıkışlarıyla sınırlıdır. Yani, MQIEPkullanmayan çıkışlar. Mümkünse, çıkışı MQIEP giriş noktalarını kullanacak şekilde yeniden kodlamanız gerekir.

IBM MQ 8.0' den **runmqsc** , bir MQI kitaplığına doğrudan bağlanmayan bir program örneğidir.

Bu nedenle, gerekli IBM MQ API kitaplığına bağlanmamış ya da MQIEP ' yi kullanmak üzere yeniden kodlanmış bir API çıkışı **runmqsc**içine yüklenemedi.

Kuyruk yöneticisi hata günlüğünde hatalar görürsünüz; örneğin, AMQ6175: Sistem paylaşılan kitaplığı dinamik olarak yükleyemedi, undefined symbol: MQCONNgibi niteleyici metinle birlikte.

ve AMQ7214: `API Çıkışı 'myexitname' modülü yüklenemedi.`

İlgili görevler

“AIX, Linux, and Windows üzerinde çıkışlar ve kurulabilir hizmetler yazılıyor” sayfa 897
AIX, Linux, and Windowsüzerindeki herhangi bir IBM MQ kitaplığını bağlamadan çıkışları yazabilir ve derleyebilirsiniz.

ALW AIX, Linux, and Windows için kurulabilir hizmetler ve bileşenler

Bu bölümde, kurulabilir hizmetler ve bunlarla ilişkili işlevler ve bileşenler tanıtılır. Bu işlevlere ilişkin arabirim, sizin ya da yazılım satıcılarının bileşenleri sağlayabilmesi için belgelenmiştir.

IBM MQ kurulabilir hizmetlerinin sağlanmasının başlıca nedenleri şunlardır:

- IBM MQ ürünleri tarafından sağlanan bileşenlerin kullanılıp kullanılmayacağını ya da diğerleriyle değiştirilip değiştirilmeyeceğini seçme esnekliğini size sağlamak için.
- IBM MQ ürünlerinde iç değişiklikler yapmadan, yeni teknolojileri kullanabilecek bileşenler sağlayarak satıcıların katılmasına izin vermek için.
- IBM MQ ' in yeni teknolojilerden daha hızlı ve daha ucuz yararlanmasını sağlamak ve bu nedenle ürünleri daha erken ve daha düşük fiyatlarla sağlamak.

Kurulabilir hizmetler ve *hizmet bileşenleri* , IBM MQ ürün yapısının bir parçasıdır. Bu yapının merkezinde, kuyruk yöneticisinin İleti Kuyruğu Arabirimi (MQI) ile ilişkili işlevi ve kuralları uygulayan kısmı bulunur. Bu merkezi parça, çalışmasını gerçekleştirmek için *kurulabilir hizmetler*adı verilen bir dizi hizmet işlevini gerektirir. Kurulabilir hizmetler şunlardır:

- Yetkilendirme hizmeti
- Ad hizmeti

Her kurulabilir hizmet, bir ya da daha çok *hizmet bileşeni*kullanılarak gerçekleştirilen ilgili bir işlev kümeleridir. Her bileşen, doğru tasarlanmış, genel kullanıma açık bir arabirim kullanılarak çağrılır. Bu, bağımsız yazılım satıcılarının ve diğer üçüncü kişilerin IBM MQ ürünleri tarafından sağlananları genişletmek ya da değiştirmek için kurulabilir bileşenler sağlamalarına olanak sağlar. Çizelge 138 sayfa 900 , kullanılabilecek hizmetleri ve bileşenleri özetler.

Çizelge 138. Kurulabilir hizmet bileşenleri özeti

Kurulabilir hizmet	Sağlanan bileşen	İşlev	Gereksinimler
Yetkilendirme hizmeti	nesne yetki yöneticisi (OAM)	Komutlar ve MQI çağrıları için yetki denetimi sağlar. Kullanıcılar, OAM ' yi genişletmek ya da değiştirmek için kendi bileşenlerini yazabilir. Örneğin, bir kullanıcı kimliğinin bir kuyruğu açma yetkisi olup olmadığını denetlemek için.	(Uygun platform yetkilendirme olanakları kabul edilir)
Ad hizmeti	Yok	Kuyruk yöneticisine, belirtilen bir kuyruğun iyesi olan kuyruk yöneticisinin adını aramak için destek sağlar. • Kullanıcı tanımlı	• Bir üçüncü taraf ya da kullanıcı tarafından yazılan ad yöneticisi

Kurulabilir hizmetler arabirimi Kurulabilir hizmetler arabirimi başvuru bilgilerikısmında açıklanmıştır.

İlgili görevler

Kurulabilir hizmetlerin yapılandırılması

Hizmet bileşeni yazılması

Bu bölümde, hizmetler, bileşenler, giriş noktaları ve dönüş kodları arasındaki ilişki açıklanmaktadır.

İşlevler ve bileşenler

Her hizmet, bir dizi ilgili işlevden oluşur. Örneğin, ad hizmeti aşağıdakiler için işlev içerir:

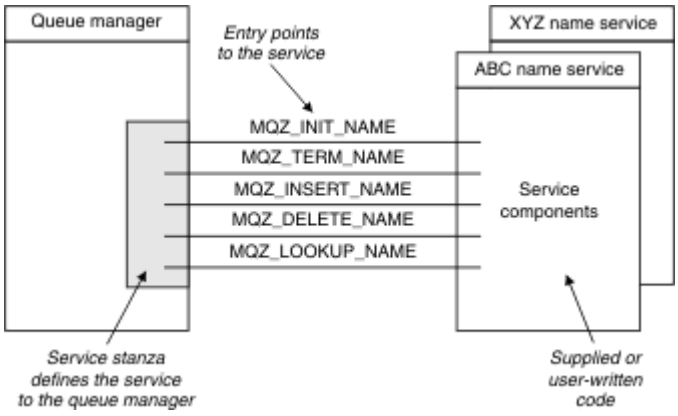
- Kuyruk adı aranıyor ve kuyruğun tanımlandığı kuyruk yöneticisinin adı döndürülüyor
- Hizmet dizinine kuyruk adı eklenmesi
- Hizmet dizininden bir kuyruk adının silinmesi

Ayrıca, başlatma ve sonlandırma işlevlerini de içerir.

Bir ya da daha çok hizmet bileşeni tarafından kurulabilir bir hizmet sağlanır. Her bileşen, o hizmet için tanımlanan işlevlerin bazılarını ya da tümünü gerçekleştirebilir. Örneğin, IBM MQ for AIXiçinde, sağlanan yetkilendirme hizmeti bileşeni OAM, kullanılabilir tüm işlevleri gerçekleştirir. Ek bilgi için bkz. “Yetkilendirme hizmeti arabirimi” sayfa 904 . Bileşen, hizmeti uygulamak için gerekli olan tüm temel kaynakları ya da yazılımları (örneğin, bir LDAP dizini) yönetmekten de sorumludur. Yapılandırma dosyaları, bileşenin yüklenmesinin ve sağladığı işlevsel yordamların adreslerinin belirlenmesinin standart bir yolunu sağlar.

Şekil 99 sayfa 901 içinde, hizmetlerin ve bileşenlerin birbiriyle nasıl ilgili olduğu gösterilmektedir:

- Bir hizmet, yapılanış kütüğündeki stanzas tarafından kuyruk yöneticisine tanımlanır.
- Her hizmet, kuyruk yöneticisinde sağlanan kod tarafından desteklenir. Kullanıcılar bu kodu değiştiremez ve bu nedenle kendi hizmetlerini oluşturamazlar.
- Her hizmet bir ya da daha fazla bileşen tarafından gerçekleştirilir; bunlar ürün ya da kullanıcı tarafından yazılabilir. Bir hizmete ilişkin birden çok bileşen çağrılabilir ve her biri hizmet içindeki farklı olanakları destekler.
- Giriş noktaları, hizmet bileşenlerini kuyruk yöneticisindeki destekleyici koda bağlar.



Şekil 99. Hizmetleri, bileşenleri ve giriş noktalarını anlama

Giriş noktaları

Her hizmet bileşeni, belirli bir kurulabilir hizmeti destekleyen yordamların giriş noktası adreslerinin bir listesiyle gösterilir. Kurulabilir hizmet, her yordam tarafından gerçekleştirilecek işlevi tanımlar.

Hizmet bileşenlerinin yapılandırıldığında sıralanması, hizmete ilişkin bir isteği yerine getirmek amacıyla giriş noktalarının çağrıldığı sırayı tanımlar.

Sağlanan `cmqzc.hüstbilgi` dosyasında, her hizmet için sağlanan giriş noktaları bir MQZID_ önekine sahip.

Hizmetler varsa, hizmetler önceden tanımlanmış bir siparişte yüklenir. Aşağıdaki liste, hizmetleri ve bunların kullanıma hazırlandıkları sırayı gösterir.

1. `NameService`

2. AuthorizationService
3. UserIdentifierService

AuthorizationService , varsayılan olarak yapılandırılan tek hizmettir. Kullanmak istiyorsanız, NameService ve UserIdentifierService öğelerini el ile yapılandırın.

Hizmetler ve hizmet bileşenleri bire bir veya birden çoğa eşlemeye sahip. Her hizmet için birden çok hizmet bileşeni tanımlanabilir. AIX and Linux sistemlerinde ServiceComponent kısmı Hizmet değeri, qm.ini dosyasındaki Hizmet kısmı Adı değeriyle eşleşmelidir. Windowssistemlerinde, ServiceComponent ' in Hizmet kayıt defteri anahtarı değeri Ad kayıt defteri anahtarı değeriyle eşleşmelidir ve şu şekilde tanımlanır: HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\Installation*MQ_INSTALLATION_NAME*\Configuration\QueueManager*qmname*\ ; burada *qmname* , kuyruk yöneticisinin adıdır.

AIX and Linux sistemleri için hizmet bileşenleri, qm.ini dosyasında tanımlandıkları sırayla başlatılır. Windowstarihinde, Windows kayıt dosyası kullanıldığından, IBM MQ değerleri alfabetik sırayla döndüren bir **RegEnumKey** çağrısı yayınlar. Bu nedenle, Windows üzerinde hizmetler, kayıtta tanımlandığı şekilde alfabetik sırayla çağrılır.

ServiceComponent tanımlamalarının sıralanması önemlidir. Bu sipariş, belirli bir hizmet için bileşenlerin çalıştırılacağı sırayı belirler. Örneğin, Windows üzerindeki AuthorizationService , MQSeries.WindowsNT.auth.serviceadlı varsayılan OAM bileşeniyle yapılandırılır. Varsayılan OAM ' yi geçersiz kılmak için bu hizmet için ek bileşenler tanımlanabilir. MQCACF_SERVICE_COMPONENT belirtilmedikçe, isteği işlemek için alfabetik sırada karşılaşılan ilk bileşen kullanılır ve o bileşenin adı kullanılır.

Dönüş kodları

Hizmet bileşenleri, çeşitli koşulları raporlamak için kuyruk yöneticisine dönüş kodları sağlar. İşlemin başarılı ya da başarısız olduğunu bildirir ve kuyruk yöneticisinin sonraki hizmet bileşenine ilerleyip ilerlemediğini belirtir. Bu göstergeyi ayrı bir *Devam* değiştirgesi taşır.

Bileşen verileri

Tek bir hizmet bileşeni, verilerin çeşitli işlevleri arasında paylaşılmasını gerektirebilir. Kurulabilir hizmetler, bir hizmet bileşeninin her çağrılışında geçirilecek isteğe bağlı bir veri alanı sağlar. Bu veri alanı, hizmet bileşeninin özel kullanımı içindir. Farklı adres alanlarından ya da işlemlerden yapılmış olsalar da, belirli bir işlevin tüm çağrıları tarafından paylaşılır. Her çağrıldığında hizmet bileşeninden adreslenebilir olması garanti edilir. *ServiceComponent* kısmında bu alanın boyutunu bildirmeniz gerekir.

Bileşenlerin başlatılması ve sonlandırılması
Bileşen kullanıma hazırlama ve sonlandırma seçeneklerinin kullanılması.

Bileşen kullanıma hazırlama yordamı çağrıldığında, bileşen tarafından desteklenen her giriş noktası için kuyruk yöneticisi **MQZEP** işlevini çağırmalıdır. **MQZEP** , hizmete ilişkin bir giriş noktası tanımlar. Tanımlanmamış tüm çıkış noktalarının NULL olduğu varsayılır.

Bir bileşen, başka bir şekilde çağrılmadan önce, birincil kullanıma hazırlama seçeneğiyle her zaman bir kez çağrılır.

Bir bileşen, belirli altyapılarda ikincil kullanıma hazırlama seçeneğiyle çağrılabilir. Örneğin, hizmete erişilen her işletim sistemi işlemi, iş parçacığı ya da görev için bir kez çağrılabilir.

İkincil kullanıma hazırlama kullanılıyorsa:

- Bileşen, ikincil kullanıma hazırlama için bir kereden fazla çağrılabilir. Bu tür her çağrı için, hizmet artık gerekli olmadığında ikincil sona erdirme için eşleşen bir çağrı yayınlanır.

  Adlandırma hizmetleri için bu, MQZ_TERM_NAME çağrısıdır.

  Yetkilendirme hizmetleri için bu, MQZ_TERM_AUTHORITY çağrısıdır.
- Bileşenin birincil ve ikincil kullanıma hazırlanması için her çağrılışında giriş noktaları yeniden belirtilmelidir (MQZEP çağrılarak).

- Bileşen için yalnızca bir bileşen verisi kopyası kullanılır; her ikincil kullanıma hazırlama için farklı bir kopya yoktur.
- Bileşen, ikincil kullanıma hazırlama gerçekleştirilmeden önce hizmete yapılan diğer çağrılar (işletim sistemi işleminden, iş parçacığından ya da görevden uygun şekilde) için çağrılmaz.
- Bileşen, **Version** değiştirgesini birincil ve ikincil kullanıma hazırlama için aynı değere ayarlamalıdır.

Bileşen, artık gerekli olmadığında, her zaman birincil sonlandırma seçeneğiyle bir kez çağrılır. Bu bileşene başka çağrı yapılmadı.

İkincil kullanıma hazırlama için çağrıldıysa, bileşen ikincil sonlandırma seçeneğiyle çağrılır.

Nesne yetkisi yöneticisi (OAM)
IBM MQ ürünleriyle verilen yetkilendirme hizmeti bileşeni, Nesne Yetkilisi Yöneticisi (OAM) olarak adlandırılır.

Varsayılan olarak OAM etkindir ve **dspmqaut** (görüntü yetkisi), **dmpmqaut** (döküm yetkisi) ve **setmqaut** (yetkiyi ayarla ya da sıfırla) denetim komutlarıyla çalışır.

Bu komutların sözdizimi ve bunların nasıl kullanılacağı, Denetim komutlarını kullanarak IBM MQ for Multiplatforms yönetmebaşlıklı konuda açıklanmaktadır.

OAM, bir birincil kullanıcının ya da grubun *varlığıyla* çalışır:

- Linux AIX AIX and Linux sistemlerinde, asıl ad bir kullanıcı kimliği ya da kullanıcı adına çalışan bir uygulama programıyla ilişkilendirilmiş bir kimliktir; grup, sistem tarafından tanımlanan bir birincil kullanıcı derlemidir.
- Windows Windows sistemlerinde, asıl ad bir Windows kullanıcı kimliği ya da kullanıcı adına çalışan bir uygulama programıyla ilişkilendirilmiş bir kimliktir; grup Windows grubudur.

Yetkiler birincil kullanıcı ya da grup düzeyinde verilebilir ya da iptal edilebilir.

Bir MQI isteği yapıldığında ya da bir komut verildiğinde, OAM, işlemle ilişkili varlığın istenen işlemi gerçekleştirme ve belirtilen kuyruk yöneticisi kaynaklarına erişme yetkisine sahip olup olmadığını denetler.

Yetkilendirme hizmeti, kendi yetki hizmeti bileşeninizi yazarak kuyruk yöneticileri için sağlanan yetki denetimini genişletmenizi ya da değiştirmenizi sağlar.

Ad hizmeti
Ad hizmeti, belirlenen bir kuyruğun iyesi olan kuyruk yöneticisinin adını aramak için kuyruk yöneticisine destek sağlayan kurulabilir bir hizmettir. Bir ad hizmetinden başka kuyruk özniteliği alınamaz.

Ad hizmeti, bir uygulamanın çıkış için uzak kuyrukları yerel kuyruklarmış gibi açmasını sağlar. Kuyruk dışındaki nesneler için ad hizmeti çağrılmaz.

Not: Uzak kuyrukların **Scope** özniteliği CELLolarak ayarlanmış olmalıdır.

Bir uygulama bir kuyruğu açtığında, kuyruk yöneticisinin dizininde önce kuyruğun adını arar. Burada bulamazsa, kuyruk adını tanıyan bir ad buluncaya kadar, konfigürasyonu tanımlanmış olduğu kadar çok sayıda ad hizmetinde görünür. Ad tanınmazsa, açma başarısız olur.

Ad hizmeti, o kuyruk için sahip olan kuyruk yöneticisini döndürür. Kuyruk yöneticisi daha sonra, komut özgün istekte kuyruk ve kuyruk yöneticisi adını belirtmiş gibi MQOPEN isteğiyle devam eder.

Ad hizmeti arabirimi (NSI), IBM MQ çerçevesinin bir parçasıdır.

Ad hizmeti nasıl çalışır?

Bir kuyruk tanımlaması kuyruk yöneticisi olarak **Scope** özniteliğini belirtiyorsa (MQSC ' de SCOPE (QMGR), kuyruk tanımlaması (tüm kuyruk öznitelikleriyle birlikte) yalnızca kuyruk yöneticisinin dizininde saklanır. Bu, kurulabilir bir hizmetle değiştirilemez.

Bir kuyruk tanımlaması **Scope** özniteliğini MQSC ' de hücre olarak belirtirse, kuyruk tanımlaması tüm kuyruk öznitelikleriyle birlikte kuyruk yöneticisinin dizininde yeniden saklanır. Ancak, kuyruk ve kuyruk

yöneticisi adı bir ad hizmetinde de saklanır. Bu bilgileri saklayabilecek bir hizmet yoksa, *Scope* hücreli bir kuyruk tanımlanamaz.

Bilgilerin saklandığı dizin hizmet tarafından yönetilebilir ya da hizmet, bu amaçla bir LDAP dizini gibi temel bir hizmeti kullanabilir. Her iki durumda da, bileşen ve kuyruk yöneticisi sona erdikten sonra da, belirtik olarak silininceye kadar, dizinde saklanan tanımlamaların kalıcı olması gerekir.

Not:

1. Uzak anasistemin yerel kuyruk tanımlamasına (CELL kapsamı), adlandırma dizini hücresi içindeki farklı bir kuyruk yöneticisinde ileti göndermek için bir kanal tanımlamanız gerekir.
2. Uzak kuyruğun kapsamı CELL olsa bile, iletileri doğrudan uzak kuyruktan alamazsınız.
3. CELL kapsamı olan bir kuyruğa gönderilirken uzak kuyruk tanımlaması gerekmez.
4. Adlandırma hizmeti, hedef kuyruğu merkezi olarak tanımlar; ancak, hedef kuyruk yöneticisine ve bir çift kanal tanımına ilişkin bir iletim kuyruğuna gereksinim duyarsınız. Buna ek olarak, yerel sistemdeki iletim kuyruğunun, uzak sistemdeki hedef kuyruğun sahibi olan kuyruk yöneticisiyle aynı adı taşıması gerekir.

   Örneğin, uzak kuyruk yöneticisinde QM01adı varsa, yerel sistemdeki iletim kuyruğu da QM01adını içermelidir.

Yetkilendirme hizmeti arabirimi
Yetkilendirme hizmeti, kuyruk yöneticisi tarafından kullanılmak üzere giriş noktaları sağlar.

Giriş noktaları aşağıdaki gibidir:

MQZ_AUTHENTICATE_USER
Bir kullanıcı kimliğini ve parolayı doğrular ve kimlik bağlamı alanlarını ayarlayabilir.

MQZ_CHECK_AUTHORITY
Bir varlığın belirli bir nesne üzerinde bir ya da daha fazla işlem gerçekleştirme yetkisine sahip olup olmadığını denetler.

MQZ_CHECK_PRIVILEGED
Belirtilen bir kullanıcının ayrıcalıklı bir kullanıcı olup olmadığını denetler.

MQZ_COPY_ALL_AUTHORITY
Başvurulan bir nesne için var olan tüm geçerli yetkileri başka bir nesneye kopyalar.

MQZ_DELETE_AUTHORITY
Belirtilen bir nesneyle ilişkili tüm yetkileri siler.

MQZ_ENUMERATE_AUTHORITY_DATA
Belirtilen seçim ölçütleriyle eşleşen tüm yetki verilerini alır.

MQZ_FREE_USER
İlişkili ayrılmış kaynakları serbest bırakır.

MQZ_GET_AUTHORITY
Bir varlığın belirtilen bir nesneye erişmek için sahip olduğu yetkiyi alır.

MQZ_GET_BELIRIT_YETKISI
Adı belirtilen bir grubun belirtilen bir nesneye erişmesi için gereken yetkiyi (ancak **kimse** grubunun ek yetkisi olmadan) ya da adı belirtilen birincil grubun belirtilen bir nesneye erişmesi için gereken yetkiyi alır.

MQZ_INIT_AUTHORITY
Yetkilendirme hizmeti bileşenini başlatır.

MQZ_INQUIRE
Yetkilendirme hizmetinin desteklenen işlevselliğini sorgular.

MQZ_REFRESH_CACHE
Tüm yetkileri yenile.

MQZ_SET_AUTHORITY
Bir varlığın belirtilen bir nesne için sahip olduğu yetkiyi ayarlar.

MQZ_TERM_AUTHORITY
Yetkilendirme hizmeti bileşenini sonlandırır.

Buna ek olarak, IBM MQ for Windowsüzerinde, yetkilendirme hizmeti kuyruk yöneticisi tarafından kullanılmak üzere aşağıdaki giriş noktalarını sağlar:

- **MQZ_CHECK_AUTHORITY_2**
- **MQZ_GET_AUTHORITY_2**
- **MQZ_GET_EXPLICIT_AUTHORITY_2**
- **MQZ_SET_AUTHORITY_2**

Bu giriş noktaları, Windows Security Identifier (NT SID) kullanımını destekler.

Bu adlar, bileşen işlevlerinin prototipini oluşturmak için kullanılabilen `cmqzc.` hüstbilgi dosyasında **typedef** olarak tanımlanır.

Kullanıma hazırlama işlevi (**MQZ_INIT_AUTHORITY**) bileşenin ana giriş noktası olmalıdır. Diğer işlevler, başlatma işlevinin bileşen giriş noktası vektörüne eklediği giriş noktası adresiyle çağrılır.

Ad hizmeti arabirimi
Ad hizmeti, kuyruk yöneticisi tarafından kullanılmak üzere giriş noktaları sağlar.

Aşağıdaki giriş noktaları sağlanır:

MQZ_INIT_ADı
Ad hizmeti bileşenini başlatın.

MQZ_TERM_ADı
Ad hizmeti bileşenini sona erdirin.

MQZ_ARAMA_ADI
Belirlenen kuyruk için kuyruk yöneticisi adını arayın.

MQZ_EKLE_ADı
Belirtilen kuyruk için sahip kuyruk yöneticisi adını içeren bir girişi, hizmet tarafından kullanılan dizine ekleyin.

MQZ_SİL_ADI
Belirtilen kuyruğa ilişkin girişi, hizmet tarafından kullanılan dizinden silin.

Konfigürasyonu tanımlanmış birden çok ad hizmeti varsa:

- Arama için, kuyruk adı çözülünceye kadar (herhangi bir bileşen aramanın durması gerektiğini belirtmedikçe) listedeki her hizmet için MQZ_LOOKUP_NAME işlevi çağrılır.
- Araya ekleme için, bu işlevi destekleyen listedeki ilk hizmet için MQZ_INSERT_NAME işlevi çağrılır.
- Silme için, listede bu işlevi destekleyen ilk hizmet için MQZ_DELETE_NAME işlevi çağrılır.

Araya ekleme ve silme işlevlerini destekleyen birden çok bileşen yoktur. Ancak, yalnızca aramayı destekleyen bir bileşen uygulanabilir ve örneğin, başka bir ad hizmeti bileşeni tarafından bilinmeyen herhangi bir adı, adın tanımlanabileceği bir kuyruk yöneticisine çözmek için listedeki son bileşen olarak kullanılabilir.

C programlama dilinde adlar, typedef deyimi kullanılarak işlev veri tipleri olarak tanımlanır. Bunlar, parametrelerin doğru olduğundan emin olmak için hizmet işlevlerinin prototipini oluşturmak için kullanılabilir.

Kurulabilir hizmetlere özgü tüm malzemeyi içeren üstbilgi dosyası, C dili için `cmqzc.` h ' dir.

Bileşenin ana giriş noktası olması gereken kullanıma hazırlama işlevinin (MQZ_INIT_NAME) yanı sıra, işlevler MQZEP çağrısıyla, kullanıma hazırlama işlevinin eklediği giriş noktası adresiyle çağrılır.

Birden çok hizmet bileşeninin kullanılması
Bir hizmet için birden çok bileşen kurabilirsiniz. Bu, bileşenlerin hizmetin yalnızca kısmi uygulamalarını sağlamalarına ve kalan işlevleri sağlamak için diğer bileşenlere güvenmelerine olanak sağlar.

Birden çok bileşen kullanma örneği

ABC_name_serv ve XYZ_name_servadlı iki ad hizmetleri bileşeni oluşturdığınızı varsayalım.

ABC_name_serv
Bu bileşen, hizmet dizinine ad eklenmesini ya da bu dizinden ad silinmesini destekler, ancak kuyruk adının araştırılmasını desteklemez.

XYZ_name_serv
Bu bileşen bir kuyruk adının araştırılmasını destekler, ancak hizmet dizinine ad eklenmesini ya da hizmet dizininden ad silinmesini desteklemez.

ABC_name_serv bileşeni, kuyruk adlarından oluşan bir veritabanını tutar ve hizmet dizininden bir ad eklemek ya da silmek için iki basit algoritma kullanır.

XYZ_name_serv bileşeni, çağrıldığı herhangi bir kuyruk adı için sabit bir kuyruk yöneticisi adı döndüren basit bir algoritma kullanır. Kuyruk adlarından oluşan bir veritabanını tutmaz ve bu nedenle ekleme ve silme işlevlerini desteklemez.

Bileşenler aynı kuyruk yöneticisine kurulur. Önce ABC_name_serv bileşeninin çağrılması için *ServiceComponent* stanzas sipariş edilir. Bileşen dizinindeki bir kuyruğu ekleme ya da silme çağrıları ABC_name_serv bileşeni tarafından işlenir; Bu işlevleri uygulayan tek kişi o. Ancak, ABC_name_serv bileşeninin çözümleyemediği bir arama çağrısı yalnızca arama bileşenine (XYZ_name_serv) aktarılır. Bu bileşen, basit algoritmasından bir kuyruk yöneticisi adı sağlar.

Birden çok bileşen kullanılırken giriş noktalarının atlanmasıyla

Bir hizmet sağlamak için birden çok bileşen kullanmaya karar verirseniz, belirli işlevleri uygulamayan bir hizmet bileşeni tasarlayabilirsiniz. Kurulabilir hizmetler çerçevesi, atlayabileceğiniz herhangi bir kısıtlama getirmez. Ancak, belirli kurulabilir hizmetler için, bir ya da daha fazla işlevin ihmal edilmesi, hizmetin amacına mantıksal olarak tutarsız olabilir.

Birden çok bileşenle kullanılan giriş noktaları örneği

Çizelge 139 sayfa 906 içinde, iki bileşenin kurulduğu kurulabilir ad hizmeti örneği gösterilmektedir. Her biri, bu belirli kurulabilir hizmetle ilişkili farklı bir işlev kümesini destekler. Araya ekleme işlevi için önce ABC bileşeni giriş noktası çağrılır. Hizmete tanımlanmamış giriş noktaları (**MQZEP** kullanılarak) boş değerli (NULL) olduğu varsayılır. Çizelgede kullanıma hazırlama için bir giriş noktası sağlanmıştır; ancak, kullanıma hazırlama bileşenin ana giriş noktası tarafından gerçekleştirildiğinden, bu gerekli değildir.

Kuyruk yöneticisinin kurulabilir bir hizmeti kullanması gerekirken, o hizmet için tanımlanan giriş noktalarını (Çizelge 139 sayfa 906 içindeki sütunlar) kullanır. Her bileşeni sırayla alan kuyruk yöneticisi, gerekli işlevi gerçekleştiren yordamın adresini belirler. Daha sonra, varsa, rutini çağırır. İşlem başarılı olursa, sonuçlar ve durum bilgileri kuyruk yöneticisi tarafından kullanılır.

Çizelge 139. Kurulabilir bir hizmete ilişkin giriş noktaları örneği

İşlev numarası	**ABC ad hizmeti bileşeni**	**XYZ ad hizmeti bileşeni**
MQZID_INIT_NAME (Kullanıma Hazırla)	ABC_initialize ()	XYZ_initialize ()
MQZID_TERM_NAME (Sonlandır)	ABC_terminate ()	XYZ_terminate ()
MQZID_INSERT_NAME (Araya ekle)	ABC_Insert ()	BOŞ DEĞERLİ
MQZID_DELETE_NAME (Sil)	ABC_Delete ()	BOŞ DEĞERLİ
MQZID_LOOKUP_NAME (Arama)	BOŞ DEĞERLİ	XYZ_Lookup ()

Yordam yoksa, kuyruk yöneticisi listedeki sonraki bileşen için bu işlemi yineler. Ayrıca, yordam varsa, ancak işlemi gerçekleştiremediğini belirten bir kod döndürürse, deneme sonraki kullanılabilir

bileşenle devam eder. Hizmet bileşenlerindeki yordamlar, işlemi gerçekleştirmek için başka girişimde bulunulmaması gerektiğini gösteren bir kod döndürebilir.

Hizmetlerin ve bileşenlerin yapılandırılması

Her kuyruk yöneticisinin Kayıt Defteri 'nde kendi bölümü olduğu Windows sistemleri dışında, kuyruk yöneticisi yapılandırma dosyalarını kullanarak hizmet bileşenlerini yapılandırırsınız.

Yordam

1. Kuyruk yöneticisine hizmeti tanımlamak ve modülün yerini belirtmek için kuyruk yöneticisi yapılanış dosyasına (`qm.ini`) stanzas ekleyin:
   - Kullanılan her hizmetin, kuyruk yöneticisine ilişkin hizmeti tanımlayan bir `Service` kısmı olmalıdır. Daha fazla bilgi için bkz. qm.ini dosyasının hizmet kısmı.
   - Bir hizmet içindeki her bileşen için bir `ServiceComponent` kısmı olmalıdır. Bu kısım, o bileşene ilişkin kodu içeren modülün adını ve yolunu tanımlar. Daha fazla bilgi için, bkz. qm.ini dosyasının ServiceComponent kısmı.

   Nesne Yetkilisi Yöneticisi (OAM) olarak bilinen yetkilendirme hizmeti bileşeni, ürünle birlikte sağlanır. Bir kuyruk yöneticisi yarattığınızda, kuyruk yöneticisi yapılanış dosyası (ya da Windows sistemlerinde kayıt dosyası), yetki hizmeti ve varsayılan bileşen (OAM) için uygun kısmı içerecek şekilde otomatik olarak güncellenir. Diğer bileşenler için kuyruk yöneticisi yapılanış dosyasını el ile yapılandırmanız gerekir.

   Kuyruk yöneticisi başlatıldığında, altyapıda bunun desteklendiği dinamik bağ tanımı kullanılarak, her hizmet bileşeninin kodu kuyruk yöneticisine yüklenir.
2. Bileşeni etkinleştirmek için kuyruk yöneticisini durdurun ve yeniden başlatın.

İlgili başvurular

qm.ini dosyasının hizmet kısmı

qm.ini dosyasının ServiceComponent kısmı

Kullanıcının yetkilendirmesini değiştirdikten sonra OAM ' nin yenilenmesi

IBM MQiçinde, bir kullanıcının yetki grubu üyeliğini değiştirdikten hemen sonra, kuyruk yöneticisini durdurup yeniden başlatmanız gerekmeden işletim sistemi düzeyinde yapılan değişiklikleri yansıtan OAM yetkilendirme grubu bilgilerini yenileyebilirsiniz. Bunu yapmak için **REFRESH SECURITY** komutunu verin.

Not: `setmqaut` komutuyla yetkileri değiştirdiğinizde, OAM bu tür değişiklikleri hemen uygular.

Kuyruk yöneticileri, yetki verilerini SYSTEM.AUTH.DATA.QUEUE. Bu veriler **amqzfuma.exe**tarafından yönetilir.

İlgili başvurular

Güvenliği yenileme

IBM i üzerine kurulabilir hizmetler ve bileşenler

Kurulabilir hizmetler ve bunlarla ilişkili işlevler ve bileşenler hakkında bilgi edinmek için bu bilgileri kullanın. Bu işlevlere ilişkin arabirim, sizin ya da yazılım satıcılarının bileşenleri sağlayabilmesi için belgelenmiştir.

IBM MQ kurulabilir hizmetlerinin sağlanmasının başlıca nedenleri şunlardır:

- IBM MQ for IBM itarafından sağlanan bileşenleri kullanmayı ya da diğerleriyle değiştirmeyi ya da genişletmeyi seçme esnekliğini size sağlamak için.
- IBM MQ for IBM iüzerinde iç değişiklikler yapmadan yeni teknolojileri kullanabilecek bileşenler sağlayarak satıcıların katılmasına izin vermek için.
- IBM MQ ' in yeni teknolojilerden daha hızlı ve daha ucuz yararlanmasını sağlamak ve bu nedenle ürünleri daha erken ve daha düşük fiyatlarla sağlamak.

Kurulabilir hizmetler ve *hizmet bileşenleri* , IBM MQ ürün yapısının bir parçasıdır. Bu yapının merkezinde, kuyruk yöneticisinin İleti Kuyruğu Arabirimi (MQI) ile ilişkili işlevi ve kuralları uygulayan kısmı bulunur. Bu merkezi parça, çalışmasını gerçekleştirmek için *kurulabilir hizmetler*adı verilen bir dizi hizmet işlevini gerektirir. IBM MQ for IBM i içinde kullanılabilen kurulabilir hizmet, yetkilendirme hizmetidir.

Her kurulabilir hizmet, bir ya da daha çok *hizmet bileşeni*kullanılarak gerçekleştirilen ilgili bir işlev kümeleridir. Her bileşen, doğru tasarlanmış, genel kullanıma açık bir arabirim kullanılarak çağrılır. Bu, bağımsız yazılım satıcılarının ve diğer üçüncü kişilerin IBM MQ for IBM itarafından sağlananları genişletmek ya da değiştirmek için kurulabilir bileşenler sağlamalarına olanak sağlar. Çizelge 140 sayfa 908 , yetkilendirme hizmeti için desteği özetler.

Çizelge 140. Yetkilendirme hizmeti bileşenleri özeti

Sağlanan bileşen	**İşlev**	**Gereksinimler**
Nesne yetkisi yöneticisi (OAM)	Komutlar ve MQI çağrıları için yetki denetimi sağlar. Kullanıcılar, OAM ' yi genişletmek ya da değiştirmek için kendi bileşenlerini yazabilir.	(Uygun platform yetkilendirme olanakları kabul edilir)
DCE ad hizmeti bileşeni **Not:** DCE yalnızca V6.0sürümünden önceki IBM MQ sürümlerinde desteklenir.	• Kuyruk yöneticilerinin kuyrukları paylaşmasına izin verir ya da • Kullanıcı tanımlı **Not:** Paylaşılan kuyrukların **Scope** özniteliği CELL olarak ayarlanmış olmalıdır.	• Sağlanan bileşen için DCE gereklidir ya da • Bir üçüncü taraf ya da kullanıcı tarafından yazılan ad yöneticisi

IBM i *IBM i üzerindeki işlevler ve bileşenler*

IBM MQ for IBM iiçinde kullanabileceğiniz işlevleri ve bileşenleri, giriş noktalarını, dönüş kodlarını ve bileşen verilerini anlamak için bu bilgileri kullanın.

Her hizmet, bir dizi ilgili işlevden oluşur. Örneğin, ad hizmeti aşağıdakiler için işlev içerir:

- Kuyruk adı aranıyor ve kuyruğun tanımlandığı kuyruk yöneticisinin adı döndürülüyor
- Hizmet dizinine kuyruk adı eklenmesi
- Hizmet dizininden bir kuyruk adının silinmesi

Ayrıca, başlatma ve sonlandırma işlevlerini de içerir.

Bir ya da daha çok hizmet bileşeni tarafından kurulabilir bir hizmet sağlanır. Her bileşen, o hizmet için tanımlanan işlevlerin bazılarını ya da tümünü gerçekleştirebilir. Bileşen, hizmeti uygulamak için gereksinim duyduğu tüm temel kaynakları veya yazılımları yönetmekten de sorumludur. Yapılandırma dosyaları, bileşenin yüklenmesinin ve sağladığı işlevsel yordamların adreslerinin belirlenmesinin standart bir yolunu sağlar.

Hizmetler ve bileşenler aşağıdaki gibi ilişkilidir:

- Bir hizmet, yapılanış kütüğündeki stanzas tarafından kuyruk yöneticisine tanımlanır.
- Her hizmet, kuyruk yöneticisinde sağlanan kod tarafından desteklenir. Kullanıcılar bu kodu değiştiremez ve bu nedenle kendi hizmetlerini oluşturamazlar.
- Her hizmet bir ya da daha fazla bileşen tarafından gerçekleştirilir; bunlar ürün ya da kullanıcı tarafından yazılabilir. Bir hizmete ilişkin birden çok bileşen çağrılabilir ve her biri hizmet içindeki farklı olanakları destekler.
- Giriş noktaları, hizmet bileşenlerini kuyruk yöneticisindeki destekleyici koda bağlar.

Giriş noktaları

Her hizmet bileşeni, belirli bir kurulabilir hizmeti destekleyen yordamların giriş noktası adreslerinin bir listesiyle gösterilir. Kurulabilir hizmet, her yordam tarafından gerçekleştirilecek işlevi tanımlar. Hizmet

bileşenlerinin yapılandırıldığında sıralanması, hizmete ilişkin bir isteği yerine getirmek amacıyla giriş noktalarının çağrıldığı sırayı tanımlar. Sağlanan `cmqzc.`hüstbilgi dosyasında, her hizmet için sağlanan giriş noktaları bir MQZID_ önekine sahip.

Dönüş kodları

Hizmet bileşenleri, çeşitli koşulları raporlamak için kuyruk yöneticisine dönüş kodları sağlar. İşlemin başarılı ya da başarısız olduğunu bildirir ve kuyruk yöneticisinin sonraki hizmet bileşenine ilerleyip ilerlemediğini belirtir. Bu göstergeyi ayrı bir *Devam* değiştirgesi taşır.

Bileşen verileri

Tek bir hizmet bileşeni, verilerin çeşitli işlevleri arasında paylaşılmasını gerektirebilir. Kurulabilir hizmetler, belirli bir hizmet bileşeninin her çağrılışında geçirilecek isteğe bağlı bir veri alanı sağlar. Bu veri alanı, hizmet bileşeninin özel kullanımı içindir. Farklı adres alanlarından ya da işlemlerden yapılmış olsalar da, belirli bir işlevin tüm çağrıları tarafından paylaşılır. Her çağrıldığında hizmet bileşeninden adreslenebilir olması garanti edilir. *`ServiceComponent`* kısmında bu alanın boyutunu bildirmeniz gerekir.

IBM i *IBM i ' da kullanıma hazırlama*

Bileşen kullanıma hazırlama yordamı çağrıldığında, bileşen tarafından desteklenen her giriş noktası için kuyruk yöneticisi `MQZEP` işlevini çağırmalıdır. `MQZEP` , hizmete ilişkin bir giriş noktasını tanımlar. Tanımlanmamış tüm çıkış noktalarının NULL olduğu varsayılır.

Birincil kullanıma hazırlama
Bir bileşen başka bir şekilde çağrılmadan önce her zaman bu seçenekle çağrılır.

İkincil kullanıma hazırlama
Bir bileşen, belirli altyapılarda bu seçenekle çağrılabilir. Örneğin, hizmete erişilen her işletim sistemi işlemi, iş parçacığı ya da görev için bir kez çağrılabilir.

İkincil kullanıma hazırlama kullanılıyorsa:

- Bileşen, ikincil kullanıma hazırlama için bir kereden fazla çağrılabilir. Bu tür her çağrı için, hizmet artık gerekli olmadığında ikincil sona erdirme için eşleşen bir çağrı yayınlanır.

  Yetkilendirme hizmetleri için bu, MQZ_TERM_AUTHORITY çağrısıdır.
- Bileşenin birincil ve ikincil kullanıma hazırlanması için her çağrılışında giriş noktaları yeniden belirtilmelidir (MQZEP çağrılarak).
- Bileşen için yalnızca bir bileşen verisi kopyası kullanılır; her ikincil kullanıma hazırlama için farklı bir kopya yoktur.
- Bileşen, ikincil kullanıma hazırlama gerçekleştirilmeden önce hizmete yapılan diğer çağrılar (işletim sistemi işleminden, iş parçacığından ya da görevden uygun şekilde) için çağrılmaz.
- Bileşen, **`Version`** değiştirgesini birincil ve ikincil kullanıma hazırlama için aynı değere ayarlamalıdır.

Birincil sona erdirme
Bileşen, artık gerekli olmadığında her zaman bu seçenekle bir kez başlatılır. Bu bileşene başka çağrı yapılmadı.

İkincil sona erdirme
Bileşen, ikincil kullanıma hazırlama için başlatıldıysa, bu seçenekle başlatılır.

IBM i *IBM i üzerinde hizmetlerin ve bileşenlerin yapılandırılması*

Hizmet bileşenlerini kuyruk yöneticisi yapılanış dosyalarını kullanarak yapılandırırsınız.

Yordam

1. Kuyruk yöneticisine hizmeti tanımlamak ve modülün yerini belirtmek için kuyruk yöneticisi yapılanış dosyasına (`qm.ini`) stanzas ekleyin:

- Kullanılan her hizmetin, kuyruk yöneticisine ilişkin hizmeti tanımlayan bir `Service` kısmı olmalıdır. Daha fazla bilgi için bkz. qm.ini dosyasının hizmet kısmı.
- Bir hizmet içindeki her bileşen için bir `ServiceComponent` kısmı olmalıdır. Bu kısım, o bileşene ilişkin kodu içeren modülün adını ve yolunu tanımlar. Daha fazla bilgi için, bkz. qm.ini dosyasının ServiceComponent kısmı.

Nesne Yetkilisi Yöneticisi (OAM) olarak bilinen yetkilendirme hizmeti bileşeni, ürünle birlikte sağlanır. Bir kuyruk yöneticisi yarattığınızda, kuyruk yöneticisi yapılanış dosyası, yetki hizmeti ve varsayılan bileşen (OAM) için uygun kısmı içerecek şekilde otomatik olarak güncellenir. Diğer bileşenler için kuyruk yöneticisi yapılanış dosyasını el ile yapılandırmanız gerekir.

Kuyruk yöneticisi başlatıldığında, altyapıda bunun desteklendiği dinamik bağ tanımı kullanılarak, her hizmet bileşeninin kodu kuyruk yöneticisine yüklenir.

2.

IBM i *IBM i üzerinde kendi hizmet bileşeninizi oluşturma*

IBM MQ for IBM iüzerinde bir hizmet bileşeninin nasıl yaratılacağını öğrenmek için bu bilgileri kullanın.

Kendi hizmet bileşeninizi yaratmak için:

- `cmqzc.h` üstbilgi kütüğünün programınızda bulunduğundan emin olun.
- Programı derleyerek ve paylaşılan kitaplıklara `libmqm*` ve `libmqmzf*`bağlayarak paylaşılan kitaplığı oluşturun.

  Not: Aracı iş parçacıklı bir ortamda çalışabildiğinden, iş parçacıklı bir ortamda çalıştırmak için OAM oluşturmanız gerekir. Bu, `libmqm` ve `libmqmzf`iş parçacıklı sürümlerinin kullanılmasını içerir.
- Kuyruk yöneticisine hizmet tanımlamak ve modülün yerini belirtmek için kuyruk yöneticisi yapılanış dosyasına stanzas ekleyin.
- Bileşeni etkinleştirmek için kuyruk yöneticisini durdurun ve yeniden başlatın.

IBM i *IBM i üzerinde yetkilendirme hizmeti*

Yetkilendirme hizmeti, kuyruk yöneticilerinin yetki olanaklarını çağırmalarını sağlayan, örneğin bir kullanıcı kimliğinin bir kuyruğu açma yetkisi olup olmadığını denetleyen kurulabilir bir hizmettir.

Bu hizmet, IBM MQ çerçevesinin bir parçası olan IBM MQ güvenlik etkinleştirme arabiriminin (SEI) bir bileşenidir. Aşağıdaki konular ele alınmıştır:

- “Nesne yetkisi yöneticisi (OAM)” sayfa 910
- “İşletim sistemine hizmet tanımlanması” sayfa 911
- “Yetkilendirme hizmeti bölmesini yapılandırma” sayfa 911
- “IBM i üzerinde yetkilendirme hizmeti arabirimi” sayfa 911

Nesne yetkisi yöneticisi (OAM)

IBM MQ ürünleriyle sağlanan yetkilendirme hizmeti bileşeni, nesne yetki yöneticisi (OAM) olarak adlandırılır. Varsayılan olarak OAM etkindir ve aşağıdaki denetim komutlarıyla çalışır:

- **WRKMQMAUT** yetki ile çalışma
- **WRKMQMAUTD** yetki verileriyle çalışma
- **DSPMQMAUT** nesne yetkisini görüntüleme
- **GRTMQMAUT** nesne yetkisi ver
- **RVKMQMAUT** nesne yetkisini iptal etme
- **RFRMQMAUT** Güvenliği yenileme

Bu komutların sözdizimi ve bunların nasıl kullanılacağı CL komutu yardımında açıklanır. OAM, bir birincil kullanıcının ya da grubun *varlığı* ile çalışır.

Bir MQI isteği yapıldığında ya da bir komut verildiğinde, OAM işlemle ilişkili varlığın yetkilendirmesini denetleyip aşağıdaki işlemleri yapıp gerçekleştiremeyeceğini denetler:

- İstenen işlemi gerçekleştirin.
- Belirtilen kuyruk yöneticisi kaynaklarına erişin.

Yetkilendirme hizmeti, kendi yetki hizmeti bileşeninizi yazarak kuyruk yöneticileri için sağlanan yetki denetimini genişletmenizi ya da değiştirmenizi sağlar.

İşletim sistemine hizmet tanımlanması

`qm.ini` kuyruk yöneticisi yapılanış kütüğündeki yetkilendirme hizmeti kısmı, kuyruk yöneticisine yetki hizmetini tanımlar. Dörtlük tiplerine ilişkin bilgi için bkz. "IBM i üzerinde hizmetlerin ve bileşenlerin yapılandırılması" sayfa 909 .

Yetkilendirme hizmeti bölmesini yapılandırma

IBM MQ for IBM i'ta:

Müdür
Bir IBM i sistem kullanıcı profilidir.

Grup
Bir IBM i sistem grubu profilidir.

Yetkiler yalnızca grup düzeyinde verilebilir ya da iptal edilebilir. Bir kullanıcının yetkisini verme ya da geri alma isteği, o kullanıcıya ilişkin birincil grubu günceller.

Her kuyruk yöneticisinin kendi kuyruk yöneticisi yapılanış dosyası vardır. Örneğin, kuyruk yöneticisi QMNAME için kuyruk yöneticisi yapılanış dosyasının varsayılan yolu ve dosya adı şudur: `/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini`.

Varsayılan yetkilendirme bileşenine ilişkin *`Service`* kısmı ve *`ServiceComponent`* kısmı otomatik olarak `qm.ini` ' e eklenir, ancak `WRKENVVAR`tarafından geçersiz kılınabilir. Diğer *`ServiceComponent`* stanzas öğeleri el ile eklenmelidir.

Örneğin, kuyruk yöneticisi yapılanış dosyasındaki aşağıdaki kısımda iki yetkilendirme hizmeti bileşeni tanımlanır:

```
Service:
    Name=AuthorizationService
    EntryPoints=7

ServiceComponent:
    Service=AuthorizationService
    Name=MQ.UNIX.authorization.service
    Module=QMQM/AMQZFU
    ComponentDataSize=0

ServiceComponent:
    Service=AuthorizationService
    Name=user.defined.authorization.service
    Module=LIBRARY/SERVICE PROGRAM NAME
    ComponentDataSize=96
```

Şekil 100. IBM i üzerinde qm.ini içindeki yetkilendirme hizmeti kısmı

İlk hizmet bileşeni kısmı `MQ.UNIX.authorization.service` , varsayılan yetkilendirme hizmeti bileşenini (OAM) tanımlar. Bu kısmı kaldırır ve kuyruk yöneticisini yeniden başlatırsanız, OAM devre dışı bırakılır ve yetki denetimi yapılmaz.

IBM i *IBM i üzerinde yetkilendirme hizmeti arabirimi*

Yetkilendirme hizmeti arabirimi, kuyruk yöneticisi tarafından kullanılmak üzere birkaç giriş noktası sağlar.

MQZ_AUTHENTICATE_USER
Bir kullanıcı kimliğini ve parolayı doğrular ve kimlik bağlamı alanlarını ayarlayabilir.

MQZ_CHECK_AUTHORITY
Bir varlığın belirli bir nesne üzerinde bir ya da daha fazla işlem gerçekleştirme yetkisine sahip olup olmadığını denetler.

MQZ_COPY_ALL_AUTHORITY
Başvurulan bir nesne için var olan tüm geçerli yetkileri başka bir nesneye kopyalar.

MQZ_DELETE_AUTHORITY
Belirtilen bir nesneyle ilişkili tüm yetkileri siler.

MQZ_ENUMERATE_AUTHORITY_DATA
Belirtilen seçim ölçütleriyle eşleşen tüm yetki verilerini alır.

MQZ_FREE_USER
İlişkili ayrılmış kaynakları serbest bırakır.

MQZ_GET_AUTHORITY
Bir varlığın belirtilen bir nesneye erişmek için sahip olduğu yetkiyi alır.

MQZ_GET_BELIRIT_YETKISI
Adı belirtilen bir grubun belirtilen bir nesneye erişmesi için gereken yetkiyi (ancak **kimse** grubunun ek yetkisi olmadan) ya da adı belirtilen birincil grubun belirtilen bir nesneye erişmesi için gereken yetkiyi alır.

MQZ_INIT_AUTHORITY
Yetkilendirme hizmeti bileşenini başlatır.

MQZ_INQUIRE
Yetkilendirme hizmetinin desteklenen işlevselliğini sorgular.

MQZ_REFRESH_CACHE
Tüm yetkileri yenile.

MQZ_SET_AUTHORITY
Bir varlığın belirtilen bir nesne için sahip olduğu yetkiyi ayarlar.

MQZ_TERM_AUTHORITY
Yetkilendirme hizmeti bileşenini sonlandırır.

Bu giriş noktaları, Windows Security Identifier (NT SID) kullanımını destekler.

Bu adlar, bileşen işlevlerinin prototipini oluşturmak için kullanılabilen `cmqzc.` hüstbilgi dosyasında **typedef** olarak tanımlanır.

Kullanıma hazırlama işlevi (**MQZ_INIT_AUTHORITY**) bileşenin ana giriş noktası olmalıdır. Diğer işlevler, başlatma işlevinin bileşen giriş noktası vektörüne eklediği giriş noktası adresiyle çağrılır.

Ek bilgi için bkz. “IBM i üzerinde kendi hizmet bileşeninizi oluşturma” sayfa 910 .

Multi Çoklu Platformda API çıkışlarının yazılması ve derlenmesi

API çıkışları, MQPUT ve MQGET gibi IBM MQ API çağrılarının davranışını değiştiren kodu yazmanızı ve ardından bu kodu bu çağrılardan hemen önce ya da hemen sonra eklemenizi sağlar.

Not: z/OS IBM MQ for z/OSüzerinde desteklenmez.

API çıkışları neden kullanılıyor?

Başvurularınızın her birinin belirli bir işi vardır ve kodu bu görevi mümkün olduğunca verimli bir şekilde yapmalıdır. Daha yüksek bir düzeyde, söz konusu kuyruk yöneticisini kullanan tüm uygulamalar için belirli bir kuyruk yöneticisine standartlar ya da iş süreçleri uygulamak isteyebilirsiniz. Bunu bireysel uygulama düzeyinin üzerinde yapmak daha verimlidir, bu nedenle etkilenen her uygulamanın kodunu değiştirmek zorunda kalmadan.

Aşağıda, API çıkışlarının yararlı olabileceği alanlara ilişkin birkaç öneri verilmiştir:

Durumu
Güvenlik için, uygulamaların bir kuyruğa ya da kuyruk yöneticisine erişim yetkisi olup olmadığını denetleyerek kimlik doğrulaması sağlayabilirsiniz. Ayrıca polis uygulamalarının API kullanımını, tek tek API çağrılarının kimliğini ve hatta kullandıkları parametreleri doğrulamasını da yapabilirsiniz.

Esneklik
Esneklik için, iş ortamınızdaki hızlı değişikliklere, o ortamdaki verilere dayanan uygulamaları değiştirmeden yanıt verebilirsiniz. Örneğin, faiz oranlarındaki, para birimi döviz kurlarındaki ya da bir üretim ortamındaki bileşenlerin fiyatındaki değişikliklere yanıt veren API çıkışları olabilir.

Kuyruk ya da kuyruk yöneticisinin kullanımının izlenmesi
Bir kuyruk ya da kuyruk yöneticisinin kullanımını izlemek için, uygulama ve ileti akışını izleyebilir, API çağrılarındaki hataları günlüğe kaydedebilir, denetim izlerini muhasebe amacıyla ayarlayabilir ya da planlama amacıyla kullanım istatistiklerini toplayabilirsiniz.

Bir API çıkışı çalıştırıldığında ne olur?

Bir çıkış programı yazdıktan ve bu programı IBM MQolarak tanımladıktan sonra, kuyruk yöneticisi kayıtlı noktalarda çıkış kodunuzu otomatik olarak çağırır.

Çalıştırılacak API çıkış yordamları, Multiplatforms üzerindeki stanzas 'ta tanımlanır. Bu konu, `mqs.ini` ve `qm.ini`yapılandırma dosyalarındaki kısıları kapsar.

Yordamların tanımı üç yerde olabilir:

1. ApiExitOrtak, mqs.ini dosyasında, kuyruk yöneticileri başladığında uygulanan IBM MQ' un tamamı için yordamları tanımlar. Bunlar, tek tek kuyruk yöneticileri için tanımlanan yordamlar tarafından geçersiz kılınabilir (bu listedeki “3” sayfa 913 öğesine bakın).
2. ApiExit mqs.ini dosyasındaki şablon, yeni bir kuyruk yöneticisi yaratıldığında, ApiExitYerel kümesine (bu listedeki “3” sayfa 913 öğesine bakın) kopyalanan IBM MQtümüne ilişkin yordamları tanımlar.
3. ApiExitLocal, qm.ini dosyasında, belirli bir kuyruk yöneticisi için geçerli olan yordamları tanımlar.

Yeni bir kuyruk yöneticisi yaratıldığında, mqs.ini içindeki ApiExitŞablon tanımlamaları, yeni kuyruk yöneticisi için qm.ini içindeki ApiExitYerel tanımlamalarına kopyalanır. Bir kuyruk yöneticisi başlatıldığında, hem ApiExitCommon hem de ApiExitYerel tanımlamaları kullanılır. ApiExityerel tanımlamaları, her ikisi de aynı adı taşıyan bir yordamı gösteriyorsa, ApiExitortak tanımlamalarının yerine geçer. “API çıkışlarını yapılandırma” sayfa 918 içinde açıklanan `Sequence` özniteliği, stanzas 'ta tanımlanan yordamların çalıştırılacağı sırayı belirler.

Birden çok IBM MQ kuruluşu için API çıkışlarını kullanma

Önceki IBM MQ sürümü için yazılan API çıkışlarının tüm sürümlerle çalışmak için kullanıldığından emin olun; IBM WebSphere MQ 7.1 içindeki çıkışlarda yapılan değişiklikler daha önceki bir sürümle çalışmayabilir. Çıkışlarda yapılan değişikliklerle ilgili daha fazla bilgi için bkz. “AIX, Linux, and Windows üzerinde çıkışlar ve kurulabilir hizmetler yazılıyor” sayfa 897.

API için sağlanan örnekler amqsaem ve amqsaxe çıkışları yazarken gerekli değişiklikleri yansıtır. İstemci uygulaması, uygulamanın başlatılmasından önce, uygulamanın ilişkilendirildiği kuyruk yöneticisinin kuruluşuna karşılık gelen doğru IBM MQ kitaplıklarının bu kitaplığa bağlandığını doğrulamalıdır.

Multi *API çıkışları yazılıyor*

C programlama dilini kullanarak her API çağrısı için çıkışlar yazabilirsiniz.

Kullanılabilir çıkışlar

Her API çağrısı için çıkışlar aşağıdaki gibi sağlanır:

- MQCB, belirtilen nesne tanıtıcısı için bir geri çağırmayı yeniden kaydettirmek ve geri çağırma üzerinde etkinleştirmeyi ve değişiklikleri denetlemek için
- MQCTL, bir bağlantı için açılan nesne tanıtıcıları üzerinde denetleme işlemleri gerçekleştirmek için

- Sonraki API çağrılarında kullanılmak üzere bir kuyruk yöneticisi bağlantı tanıtıcısı sağlamak için MQCONN/MQCONNX
- MQDISC, kuyruk yöneticisiyle bağlantıyı kesmek için
- MQBEGIN, genel iş birimini başlatmak için (UOW)
- MQBACK, bir UOW ' yi geri almak için
- MQCMIT, bir UOW ' yi kesinleştirmek için
- MQOPEN, sonraki erişim için bir IBM MQ kaynağını açmak için
- MQCLOSE, daha önce erişim için açılmış bir IBM MQ kaynağını kapatmak için
- MQGET, daha önce erişim için açılmış bir kuyruktan ileti almak için
- MQPUT1, bir iletiyi kuyruğa yerleştirmek için
- MQPUT, daha önce erişim için açılmış bir kuyruğa ileti yerleştirmek için
- MQINQ, daha önce erişim için açılmış bir IBM MQ kaynağının özniteliklerini sorgulamak için
- MQSET, daha önce erişim için açılmış olan bir kuyruğun özniteliklerini ayarlamak için
- MQSTAT, durum bilgilerini almak için
- MQSUB, belirli bir konuya ilişkin uygulama aboneliğini kaydettirmek için
- MQSUBRQ, abonelik isteğinde bulunmak için

MQ_CALLBACK_EXIT, geri çağırma işlemesinden önce ve sonra gerçekleştirilecek bir çıkış işlevi sağlar. Daha fazla bilgi için bkz. Callback-MQ_CALLBACK_EXIT.

API çıkışları yazılıyor

API çıkışları içinde, çağrılar genel olarak şu formu alır:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

Burada *call* , MQ öneki olmayan MQI çağrısı adıdır; örneğin, PUT, GET. *parameters* , öncelikle çıkış ve dış denetim blokları MQAXP (API çıkış parametresi yapısı) ve MQAXC (API çıkış bağlamı yapısı)arasındaki iletişimi sağlayarak çıkışın işlevini denetler. *context* , API çıkışının çağrıldığı bağlamı tanımlar ve *ApiCallParameters* , MQI çağrısına ilişkin değiştirgeleri gösterir.

API çıkışınızı yazmanıza yardımcı olmak için amqsaxe0.cadlı örnek bir çıkış sağlanır; bu çıkış, belirttiğiniz bir dosyaya izleme girişleri oluşturur. Bu örneği, çıkışları yazarken başlangıç noktası olarak kullanabilirsiniz. Örnek çıkışı kullanma hakkında daha fazla bilgi için bkz. “API çıkış örnek programı” sayfa 1029.

API çıkış çağrıları, dış denetim blokları ve ilişkili konular hakkında daha fazla bilgi için bkz. API çıkış başvurusu.

Bir çıkışın nasıl yazılacağı, derleneceği ve yapılandırılacağı hakkında genel bilgi için bkz. “AIX, Linux, and Windows üzerinde çıkışlar ve kurulabilir hizmetler yazılıyor” sayfa 897.

API çıkışlarında ileti tanıtıcılarının kullanılması

Bir API çıkışının hangi ileti özelliklerine erişimi olduğunu denetleyebilirsiniz. Özellikler bir ExitMsgtanıtıcısı ile ilişkilendirilir. Bir koyma çıkışında ayarlanan özellikler, konmakta olan iletide ayarlanır, ancak bir alma çıkışında alınan özellikler uygulamaya döndürülmez.

MQXEP MQI çağrısıyla **Function** set to MQXF_INIT and **ExitReason** set to MQXR_CONNECTION ile bir MQ_INIT_EXIT çıkış işlevini kaydettirdiğinizde, bir MQXEPO yapısını **ExitOpts** değiştirgesi olarak geçirir. MQXEPO yapısı, çıkışın kullanımına sunulacak özellikler kümesini belirten ExitProperties alanını içerir. Bir MQRFH2 klasör adına karşılık gelen özelliklerin önekini gösteren bir karakter dizgisi olarak belirtilir.

Her API çıkışı, ExitMsgHandle alanı içeren bir MQAXP yapısı alır. Bu alan, IBM MQ tarafından oluşturulan ve bir bağlantıya özgü bir değere ayarlanır. Bu nedenle, aynı bağlantıda aynı ya da farklı tiplerdeki API çıkışları arasında tanıtıcı değiştirilmez.

MQXR_BEFORE **ExitReason** ile bir MQ_PUT_EXIT ya da MQ_PUT1_EXIT içinde, bir ileti koymadan önce bir API çıkışı gerçekleştirildi; ExitMsgile ilişkili özellikler (ileti tanımlayıcı özellikleri dışında), çıkış tamamlandığında ileti tanıtıcısı ayarlanır. Bunun olmasını önlemek için, ExitMsgTanıtıcısı 'nı MQHM_NONE olarak ayarlayın. Farklı bir ileti tanıtıcısı da sağlayabilirsiniz.

Bir MQ_GET_EXIT ve MQ_CALLBACK_EXIT 'te, ExitMsgtanıtıcısı özelliklerden temizlenir ve ileti tanımlayıcısı özellikleri dışında, MQ_INIT_EXIT kaydedildiğinde ExitProperties alanında belirtilen özelliklerle doldurulur. Bu özellikler, alma uygulaması tarafından kullanılamaz. Alma uygulaması MQGMO (İleti seçeneklerini al) alanında bir ileti tanıtıcısı belirtmişse, API çıkışında, ileti tanımlayıcı özellikleri de içinde olmak üzere, o tanıtıcıyla ilişkilendirilmiş tüm özellikler kullanılabilir. ExitMsgHandle 'a özelliklerle veri yerleştirilmesini önlemek için bunu MQHM_NONE olarak ayarlayın.

Not: Şu yerde işlenecek çıkış iletisi özellikleri için:

- MQ_GET_EXIT işlevinden sonra, çıkış için önce bir MQ_GET_EXIT işlevi tanımlamanız gerekir.
- MQ_CALLBACK_EXIT işlevinden önce, çıkış için önce bir MQ_CB_EXIT işlevi tanımlamanız gerekir.

V 9.3.2 IBM MQ 9.3.2' den MQ-GET-EXIT ve MQ_CALLBACK_EXIT ile ilgili önceki deyimler artık geçerli değildir.

API çıkışlarında ileti tanıtıcılarının kullanımını göstermek için amqsaem0.cadlı örnek bir program sağlanır.

İlgili başvurular

Kullanıcı çıkışları, API çıkışları ve kurulabilir hizmetler başvurusu

Multi *API çıkışları derleniyor*

Bir çıkış yazdıktan sonra, aşağıdaki gibi derleyip bağlayabilirsiniz.

Aşağıdaki örneklerde, “API çıkış örnek programı” sayfa 1029içinde açıklanan örnek program için kullanılan komutlar gösterilmektedir. Windows sistemleri dışındaki platformlarda, örnek API çıkış kodunu *MQ_INSTALLATION_PATH*/samp içinde ve derlenmiş ve bağlantılı paylaşılan kitaplığı *MQ_INSTALLATION_PATH*/samp/biniçinde bulabilirsiniz.

Windows Windows sistemleri için örnek API çıkış kodunu *MQ_INSTALLATION_PATH* \Tools\c\Samplesiçinde bulabilirsiniz. *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulduğu dizini gösterir.

Not: 64 bit programlama uygulamalarına ilişkin rehberlik, 64 bit platformlarda Coding standards(Kodlama standartları) içinde listelenmiştir.

Çoklu yayın istemcileri için, bazı iletiler kuyruk yöneticisinden geçmeyebileceğinden, API çıkışlarının ve veri dönüştürme çıkışlarının istemci tarafında çalışabilmeleri gerekir. Aşağıdaki kitaplıklar, sunucu paketlerinin yanı sıra istemci paketlerinin de bir parçasıdır:

Çizelge 141. İstemci ve sunucu paketlerindeki kitaplıklar

İşletim sistemi	**Kütüphaneler**
AIX AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
IBM i IBM i	LIBMQM ve LIBMQM_R
Linux Linux	32 bit & 64 bit: libmqm.so & libmqm_r.so
Windows Windows	32 bit & 64 bit: mqm.dll & mqm.pdb

Linux AIX *AIX and Linux sistemlerinde API çıkışlarının derlenmesi*

AIX and Linux sistemlerinde API çıkışlarının nasıl derleneceğine ilişkin örnekler.

Tüm altyapılarda, modülün giriş noktası MQStart olur.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

AçıkAIX

AIX

Aşağıdaki komutlardan birini vererek API çıkış kaynak kodunu derleyin:

32 bit uygulamalar

İş parçacıklı olmayan

```
cc -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe \
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
xlc_r -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits/amqsaxe_r \
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

64 bit uygulamalar

İş parçacıklı olmayan

```
cc -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe \
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
xlc_r -q64 -e MQStart -bE:amqsaxe.exp -bM:SRE -o /var/mqm/exits64/amqsaxe_r \
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc
```

AçıkLinux

Linux

Aşağıdaki komutlardan birini vererek API çıkış kaynak kodunu derleyin:

31 bit uygulamalar

İş parçacıklı olmayan

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

32 bit uygulamalar

İş parçacıklı olmayan

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/amqsaxe_r amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

64 bit uygulamalar

İş parçacıklı olmayan

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsaxe_r amqsaxe0.c \
    -I MQ_INSTALLATION_PATH/inc
```

Windows *Windows sistemlerinde API çıkışlarının derlenmesi*

Windows üzerinde `amqsaxe0.c` örnek API çıkış programını derleyin ve bağlayın

Bildirge (manifest) dosyası, derlenmiş bir uygulamaya ya da DLL ' ye gömülebilecek sürümü ya da diğer bilgileri içeren isteğe bağlı bir XML belgesidir.

Böyle bir belgeniz yoksa, **mt** komutunda -manifest *manifest.file* parametresini kaldırın.

Windowsüzerinde `amqsaxe0.c` dosyasını derlemek ve bağlamak için Şekil 101 sayfa 917 ya da Şekil 102 sayfa 917 içindeki örneklerdeki komutları uyarlayın. Komutlar Microsoft Visual Studio 2008, 2010 ya da 2012 ile çalışır. Örnekler, `C:\Program Files\IBM\MQ\tools\c\samples` dizininin yürürlükteki dizin olduğunu varsayar.

32 bit

```
cl /c /nologo /MD /Foamqsaxe0.obj amqsaxe0.c

link /nologo /dll /def:amqsaxe.def



amqsaxe0.obj \
   /manifest /out:amqsaxe.dll

mt -nologo -manifest amqsaxe.dll.manifest \
   -outputresource:amqsaxe.dll;2
```

Şekil 101. `amqsaxe0.c` 32 bit Windows üzerinde derle ve bağla

64 bit

```
cl /c /nologo /MD /Foamqsaxe0.obj amqsaxe0.c

link /nologo /dll /def:amqsaxe.def \
     /libpath:..\..\lib64 \



amqsaxe0.obj /manifest /out:amqsaxe.dll

mt -nologo -manifest amqsaxe.dll.manifest \
   -outputresource:amqsaxe.dll;2
```

Şekil 102. `amqsaxe0.c` 64 bit Windows üzerinde derle ve bağla

İlgili kavramlar

“API çıkış örnek programı” sayfa 1029

Örnek API çıkışı, **MQAPI_TRACE_LOGFILE** ortam değişkeninde tanımlı bir önekle kullanıcı tarafından belirtilen bir dosya için MQI izlemesi oluşturur.

IBM i *IBM i üzerinde API çıkışları derleniyor*
IBM iüzerinde API çıkışları derleniyor.

Çıkış aşağıdaki gibi yaratılır (C dili örneği için):

1. CRTCMOD kullanarak bir modül yaratın. TERASPACE(*YES *TSIFC) değiştirgesini ekleyerek teraspace kullanmak için derleyin.
2. CRTSRVPGM kullanarak modülden bir hizmet programı yaratın. Çok iş parçacıklı API çıkışları için bunu QMQM/LIBMQMZF_R hizmet programına bağlamanız gerekir.

API çıkışlarını yapılandırma

Yapılandırma bilgilerini değiştirerek API çıkışlarını etkinleştirmek için IBM MQ ' i yapılandırın.

Yapılandırma bilgilerini değiştirmek için, çıkış yordamlarını ve bunların çalıştırılma sırasını tanımlayan bölmeleri değiştirmeniz gerekir. Bu bilgiler aşağıdaki şekillerde değiştirilebilir:

- Linux Windows IBM MQ Explorer on Windows ve Linux (x86 ve x86-64 platformları) platformlarının kullanılması.
- Windows Windowsüzerinde **amqmdain** komutunu kullanma.
- Multi `mqs.ini` ve `qm.ini` dosyalarını doğrudan Multiplatforms üzerinde kullanma.

  `mqs.ini` dosyası, belirli bir düğümdeki tüm kuyruk yöneticileriyle ilgili bilgileri içerir. Bu bilgileri aşağıdaki yerlerde bulabilirsiniz:

  – Linux AIX AIX and Linuxüzerindeki `/var/mqm` dizininde.

  – Windows Windows sistemlerinde `HKLM\SOFTWARE\IBM\WebSphere MQ` anahtarında belirtilen WorkPath içinde.

  – IBM i IBM iüzerindeki `/QIBM/UserData/mqm` dizininde.

  `qm.ini` dosyası, belirli bir kuyruk yöneticisiyle ilgili bilgileri içerir. Kuyruk yöneticisinin kapladığı dizin ağacının kökünde tutulan her kuyruk yöneticisi için bir kuyruk yöneticisi yapılanış dosyası vardır. Örneğin, QMNAME adlı bir kuyruk yöneticisine ilişkin konfigürasyon dosyasının yolu ve adı:

  IBM i IBM i sistemlerinde:

  ```
  /QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini
  ```

  Linux AIX AIX and Linux sistemlerinde:

  ```
  /var/mqm/qmgrs/QMNAME/qm.ini
  ```

  Windows Windows sistemlerinde:

  ```
  C:\ProgramData\IBM\MQ\qmgrs\QMNAME\qm.ini
  ```

  Bir yapılandırma dosyasını düzenlemeden önce, gereksinim oluşursa geri alabileceğiniz bir kopyaya sahip olmak için dosyayı yedekleyin.

  Yapılandırma dosyalarını aşağıdaki yollardan biriyle düzenleyebilirsiniz:

  – Otomatik olarak, düğümdeki kuyruk yöneticilerinin yapılandırmasını değiştiren komutlar kullanılarak.

  – Standart bir metin düzenleyicisini kullanarak el ile.

Bir yapılanış dosyası özniteliğinde yanlış bir değer belirlerseniz, değer yoksayılır ve sorunu belirtmek için bir işleç iletisi yayınlanır. Etki, özniteliğin tamamen eksik olmasıyla aynıdır.

Yapılandırılacak stanzas

Değiştirilmesi gereken kıta aşağıda belirtilmiştir:

ApiExitOrtak

`mqs.ini` içinde ve IBM MQ özellikler sayfasında, Çıkışlar altında IBM MQ Explorer içinde tanımlanır.

Herhangi bir kuyruk yöneticisi başlatıldığında, bu kıtadaki öznitelikler okunur ve `qm.ini`içinde tanımlanan API çıkışları tarafından geçersiz kılınır.

ApiExitŞablonu

`mqs.ini` içinde ve IBM MQ özellikler sayfasında, Çıkışlar altında IBM MQ Explorer içinde tanımlanır.

Herhangi bir kuyruk yöneticisi yaratıldığında, bu kıtadaki öznitelikler ApiExitLocal kısmı altındaki yeni yaratılan qm.ini dosyasına kopyalanır.

ApiExitYerel

Çıkışlar altında, `qm.ini` içinde ve kuyruk yöneticisi özellikleri sayfasında IBM MQ Explorer içinde tanımlanır.

Kuyruk yöneticisi başlatıldığında, burada tanımlanan API çıkışları `mqs.ini`içinde tanımlanan varsayılanları geçersiz kılar.

Bölmeye ilişkin öznitelikler

- Aşağıdaki özniteliği kullanarak API çıkışını adlandırın:

  Ad=ApiExit_name
  API çıkışının açıklayıcı adı, MQAXP yapısının ExitInfoAd alanında iletildi.

  Bu ad benzersiz olmalı, 48 karakterden uzun olmamalı ve yalnızca IBM MQ nesnelerinin adları (örneğin, kuyruk adları) için geçerli karakterler içermelidir.

- Aşağıdaki öznitelikleri kullanarak çalıştırılacak API çıkış kodunun modül ve giriş noktasını tanımlayın:

  İşlev=işlev_adı
  API çıkış kodunu içeren modüldeki işlev giriş noktasının adı. Bu giriş noktası MQ_INIT_EXIT işlevidir.

  Bu alanın uzunluğu MQ_EXIT_NAME_LENGTH ile sınırlıdır.

  Module=modüla_adı
  API çıkış kodunu içeren modül.

  Bu alan modülün tam yol adını içeriyorsa, olduğu gibi kullanılır.

  Bu alan yalnızca modül adını içeriyorsa, modül, qm.iniiçindeki `ExitPath` içindeki `ExitsDefaultPath` özniteliği kullanılarak bulunur.

  Ayrı iş parçacıklı kitaplıkları destekleyen platformlarda, API çıkış modülünün hem iş parçacıklı olmayan hem de iş parçacıklı bir sürümünü sağlamanız gerekir. İş parçacıklı sürümün bir `_r` soneki olmalıdır. IBM MQ uygulama sınırlı kod öbeğinin iş parçacıklı sürümü, yüklenmeden önce belirtilen modül adının sonuna `_r` örtük olarak eklenir.

  Bu alanın uzunluğu, altyapının desteklediği yol uzunluğu üst sınırıyla sınırlıdır.

- İsteğe bağlı olarak, verileri aşağıdaki özniteliği kullanarak çıkışla birlikte iletin:

  Veri=veri_adı
  MQAXP yapısının ExitData alanında API çıkışına geçirilecek veriler.

  Bu özniteliği eklerse, baştaki ve sondaki boşluklar kaldırılır, kalan dizgi 32 karaktere kesilir ve sonuç çıkışa geçirilir. Bu özniteliği atlarsanız, çıkışa varsayılan değer olan 32 boşluk geçirilir.

  Bu alanın uzunluk üst sınırı 32 karakterdir.

- Bu çıkışın sırasını, aşağıdaki özniteliği kullanarak diğer çıkışlarla göreli olarak tanımlayın:

  Sequence=sequence_number (Sıralama_numarası)
  Bu API çıkışının diğer API çıkışlarıyla göreli olarak çağrıldığı sıra. Düşük sıra numarasına sahip bir çıkış, daha yüksek sıra numarasına sahip bir çıkıştan önce çağrılır. Çıkışların sıra numaralandırmasının bitişik olması gerekmez. 1, 2, 3 dizisinin sonucu 7, 42, 1096 ile aynıdır. İki çıkış aynı sıra numarasına sahipse, kuyruk yöneticisi hangisinin önce aranacağına karar verir. MQAXP ' de ExitChainAreaPtr ile gösterilen ExitChainAlanına zamanı ya da bir imleyiciyi koyarak ya da kendi günlük dosyanızı yazarak, olaydan sonra hangisinin çağrıldığı anlayabilirsiniz.

  Bu öznitelik, işaretsiz bir sayısal değerdir.

Örnek stanzas

Örnek mqs.ini dosyası aşağıdaki kısmı içerir:

ApiExitŞablonu
Bu bölüm, `OurPayrollQueueAuditor`tanımlayıcı adı, modül adı `auditor`ve sıra numarası 2 olan bir çıkışı tanımlar. Çıkışa 123 veri değeri geçirilir.

ApiExitOrtak
Bu bölüm, açıklayıcı adı `MQPoliceman`, modül adı `tmqp`ve sıra numarası 1 olan bir çıkışı tanımlar. Geçirilen veriler bir yönerge (`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
```

Aşağıdaki örnek qm.ini dosyası, tanımlayıcı adı `ClientApplicationAPIchecker`, modül adı `ClientAppChecker`ve sıra numarası 3 olan bir çıkışın ApiExitYerel tanımlamasını içerir.

```
qm.ini

   ApiExitLocal:
      Name=ClientApplicationAPIchecker
      Sequence=3
      Function=EntryPoint
      Module=/usr/Dev/ClientAppChecker
      Data=9.20.176.20
```

İleti sistemi kanalları için kanal çıkış programları

Bu konu derlemi, ileti sistemi kanallarına ilişkin IBM MQ kanal çıkışı programlarına ilişkin bilgileri içerir.

İleti kanalı aracıları (MCA), veri dönüştürme çıkışlarını da çağırabilir. Veri dönüştürme çıkışları yazma hakkında daha fazla bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941.

Bu bilgilerin bazıları, IBM MQ MQI clients ' u kuyruk yöneticilerine bağlayan MQI kanallarındaki çıkışlar için de geçerlidir. Daha fazla bilgi için bkz. MQI kanalları için kanal çıkış programları.

Kanal çıkış programları, MCA programları tarafından gerçekleştirilen işlemlerde tanımlı yerlerde çağrılır.

Bu kullanıcı çıkış programlarından bazıları tamamlayıcı çiftler halinde çalışır. Örneğin, iletime ilişkin iletileri şifrelemek için gönderen MCA tarafından bir kullanıcı çıkış programı çağrılırsa, işlemin tersine çevrilmesi için tamamlayıcı işlemin alıcı ucunda çalışması gerekir.

Çizelge 142 sayfa 921 içinde, her kanal tipi için kullanılabilecek kanal çıkışı tipleri gösterilir.

Çizelge 142. Her kanal tipi için kullanılabilir kanal çıkışları

Kanal Tipi	İleti çıkışı	İleti-çıkışı yeniden dene	Alma çıkışı	Güvenlik Çıkışı	Çıkış gönder	Otomatik tanımlama çıkışı
Gönderen kanalı	Evet		Evet	Evet	Evet	
Sunucu kanalı	Evet		Evet	Evet	Evet	
Küme-gönderen kanalı	Evet		Evet	Evet	Evet	Evet
Alıcı kanalı	Evet	Evet	Evet	Evet	Evet	Evet
İsteyen kanalı	Evet	Evet	Evet	Evet	Evet	
Küme-alıcı kanalı	Evet	Evet	Evet	Evet	Evet	Evet
İstemci-bağlantı kanalı			Evet	Evet	Evet	
Sunucu bağlantısı kanalı			Evet	Evet	Evet	Evet

Notlar: z/OS

1. z/OSişletim sistemi üzerinde, otomatik tanımlama çıkışı yalnızca küme gönderen ve küme alıcı kanalları için geçerlidir.

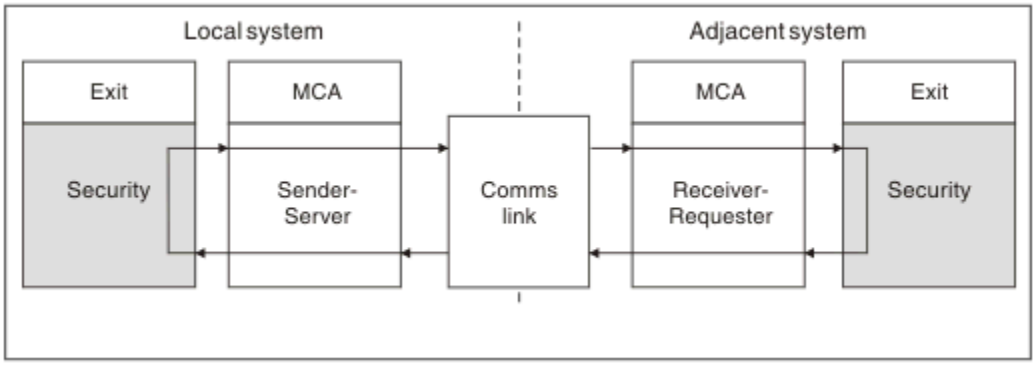
Kanal çıkışlarını bir istemcide çalıştıracaksanız, MQSERVER ortam değişkenini kullanamazsınız. Bunun yerine, İstemci kanal tanımlama çizelgesindeaçıklandığı gibi bir istemci kanal tanımlama çizelgesi (CCDT) yaratın ve bu çizelgeye başvurun.

İşleme-genel bakış

MCA ' ların kanal çıkış programlarını nasıl kullandığına ilişkin bir genel bakış.

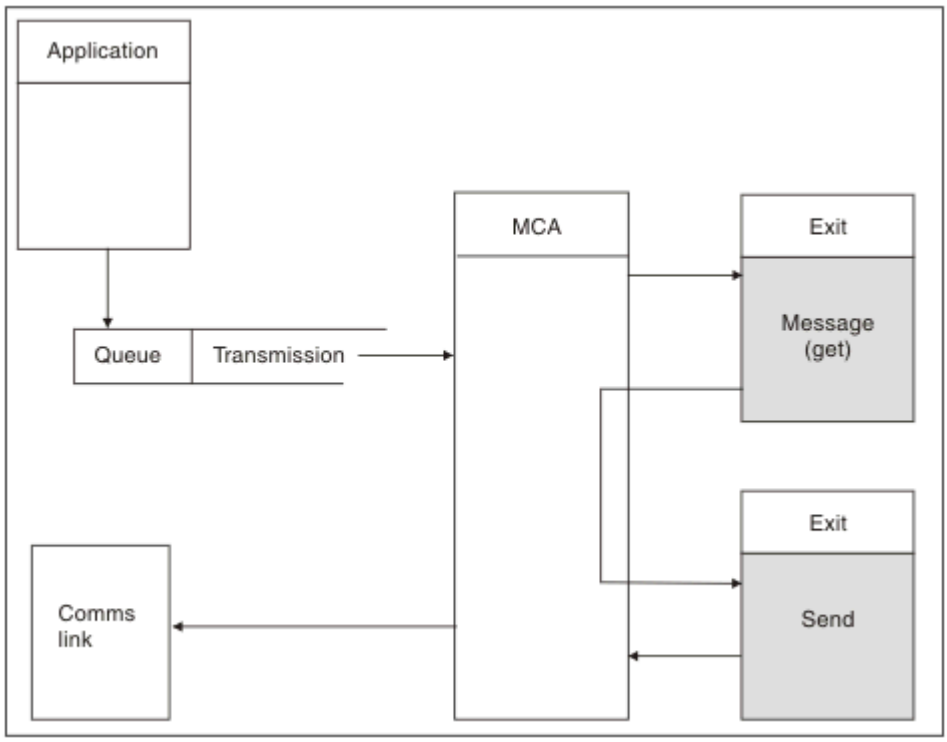
Başlangıçta, MCA ' lar işlemeyi eşitlemek için bir başlangıç iletişim kutusu değiştirir. Daha sonra güvenlik çıkışlarını içeren bir veri değişimine geçerler. Başlatma aşamasının tamamlanması ve iletilerin aktarılmasına izin verilmesi için bu çıkışların başarıyla sona ermesi gerekir.

Güvenlik denetimi aşaması, Şekil 103 sayfa 922içinde gösterildiği gibi bir döngüdür.

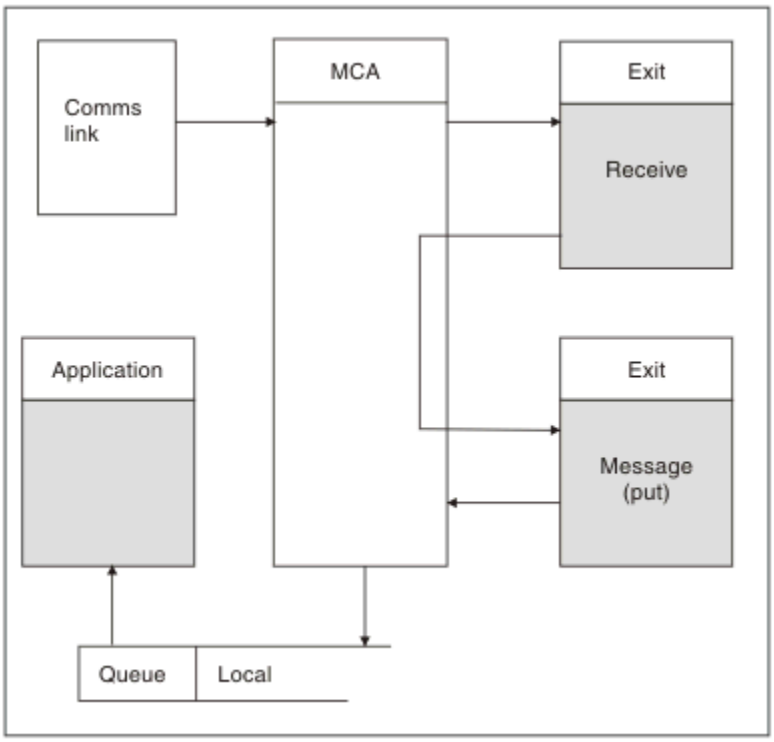


Şekil 103. Güvenlik çıkış döngüsü

İleti aktarma aşamasında, gönderen MCA iletileri bir iletim kuyruğundan alır, ileti çıkışını çağırır, gönderme çıkışını çağırır ve iletiyi alan MCA ' ya gönderir (bkz. Şekil 104 sayfa 922).



Şekil 104. İleti kanalının gönderen ucundaki gönderme çıkışı örneği



Şekil 105. İleti kanalının alıcı ucundaki alma çıkışı örneği

Alan MCA, iletişim bağlantısından bir ileti alır, alma çıkışını çağırır, ileti çıkışını çağırır ve daha sonra, iletiyi Şekil 105 sayfa 923içinde gösterildiği gibi yerel kuyruğa koyar. (Alma çıkışı, ileti çıkışı çağrılmadan önce bir kereden fazla çağrılabilir.)

Kanal çıkış programları yazılıyor

Kanal çıkış programları yazmanıza yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

Kullanıcı çıkışları ve kanal çıkış programları, izleyen kısımlarda belirtilenler dışında, tüm MQI çağrılarını kullanabilir. MQ V7 ve üstü için MQCXP yapısı sürüm 7 ve üstü, MQCONN yayınlamak yerine kullanılabilecek hConnbağlantı tanıtıcısını içerir. Önceki sürümler için, kanal kuyruk yöneticisine bağlı olduğu için MQRC_ALREADY_CONNECTED uyarısı döndürülse de, bağlantı tanıtıcısını elde etmek için bir MQCONN verilmelidir.

Kanal çıkışının iş parçacığı korumalı olması gerektiğini unutmayın.

İstemci-bağlantı kanallarındaki çıkışlar için, çıkışın bağlanmaya çalıştığı kuyruk yöneticisi çıkışın nasıl bağlandığına bağlıdır. Çıkış MQM.LIB (ya da IBM i üzerinde QMQM/LIBMQM) ve MQCONN çağrısında bir kuyruk yöneticisi adı belirtmezseniz, çıkış sisteminizdeki varsayılan kuyruk yöneticisine bağlanmayı dener. Çıkış MQM.LIB (ya da IBM i üzerinde QMQM/LIBMQM) ve MQCD ' nin QMgrName alanı aracılığıyla çıkışa geçirilen kuyruk yöneticisinin adını belirtirseniz, çıkış o kuyruk yöneticisine bağlanmayı dener. Çıkış MQIC.LIB ya da başka bir kitaplık, kuyruk yöneticisi adı belirtseniz de belirtmesseniz de MQCONN çağrısı başarısız olur.

Bir kanal çıkışında iletilen hConn ile ilişkili hareketin durumunu değiştirmekten kaçınmalısınız; MQCMIT, MQBACK ya da MQDISC fiillerini hConnkanalında kullanmamalısınız ve hConnkanalını belirten MQBEGIN komutunu kullanamazsınız.

MQCONNX, yeni bir IBM MQ bağlantısı yaratmak için MQCNO_HANDLE_SHARE_BLOCK ya da MQCNO_HANDLE_SHARE_NO_BLOCK belirtilerek kullanılırsa, bağlantının doğru yönetilmesini ve kuyruk yöneticisiyle bağlantısının doğru biçimde kesilmesini sağlamak sizin sorumluluğunuzdadır. Örneğin, bağlantıyı kesmeden her çağrıda kuyruk yöneticisine yeni bir bağlantı oluşturan bir kanal çıkışı, bağlantı kurulmasına ve aracı iş parçacıklarının sayısında artışa neden olur.

Bir çıkış, MCA ' nın kendisiyle aynı iş parçacığında çalışır ve aynı bağlantı tanıtıcısını kullanır. Bu nedenle, MCA ile aynı UOW içinde çalışır ve senkronizasyon noktası altında yapılan tüm çağrılar, toplu işin sonunda kanal tarafından kesinleştirilir veya yedeklenir.

Bu nedenle, bir kanal ileti çıkışı, yalnızca özgün iletiyi içeren toplu iş kesinleştirildiğinde o kuyrukta kesinleştirilen bildirim iletilerini gönderebilir. Bu nedenle, bir kanal ileti çıkışından eşitleme noktası MQI çağrıları gönderilebilir.

Bir kanal çıkışı MQCD ' deki alanları değiştirebilir. Ancak, listelenen koşullar dışında, bu değişiklikler üzerinde bir değişiklik yapılmaz. Bir kanal çıkış programı MQCD veri yapısındaki bir alanı değiştirirse, yeni değer IBM MQ kanal işlemi tarafından yoksayılır. Ancak, yeni değer MQCD ' de kalır ve bir çıkış zincirinde kalan çıkışlara ve kanal yönetim ortamını paylaşan sohbetlere geçirilir. Daha fazla bilgi için bkz. Kanal çıkışında MQCD alanlarını değiştirme

Ayrıca, C dilinde yazılmış programlar için, yeniden girilemeyen C kitaplığı işlevi bir kanal çıkış programında kullanılmamalıdır.

Linux AIX Aynı anda birden çok kanal çıkışı kitaplığı kullanırsanız, iki farklı çıkışın kodu aynı adı taşıyan işlevler içeriyorsa, bazı UNIX and Linux platformlarında sorunlar ortaya çıkabilir. Bir kanal çıkışı yüklendiğinde, dinamik yükleyici, çıkış kitaplığındaki işlev adlarını kitaplığın yüklendiği adreslere çözümler. İki çıkış kitaplığı aynı adlara sahip ayrı işlevler tanımlarsa, bu çözüm işlemi bir kitaplığın işlev adlarını başka bir kitaplığın işlevlerini kullanacak şekilde yanlış çözebilir. Bu sorun ortaya çıkarsa, bağlantı kesiciye, bu işlevler etkilenmediği için yalnızca gerekli çıkış ve MQStart işlevlerini dışa aktarması gerektiğini belirtin. Diğer işlevlere, kendi çıkış kitaplıkları dışındaki işlevler tarafından kullanılmaması için yerel görünürlük verilmelidir. Daha fazla bilgi için bağlantı bağlantısına ilişkin belgelere bakın.

Tüm çıkışlar bir kanal çıkış değiştirgesi yapısı (MQCXP), kanal tanımlama yapısı (MQCD), hazırlanmış veri arabelleği, veri uzunluğu değiştirgesi ve arabellek uzunluğu değiştirgesiyle çağrılır. Arabellek uzunluğu aşılmamalıdır:

- İleti çıkışları için, kanal boyunca en büyük iletinin gönderilmesine ve MQXQH yapısının uzunluğuna izin vermeniz gerekir.
- Gönderme ve alma çıkışları için, izin vermeniz gereken en büyük arabellek aşağıdaki gibidir:

  LU 6.2
  32 KB

  TCP:
  IBM i IBM i 16 KB

  IBM i Diğerleri 32 KB

  Not: Kullanılabilir uzunluk üst sınırı bu uzunluktan 2 byte daha az olabilir. Ayrıntılar için MaxSegmentUzunluğu içinde döndürülen değeri denetleyin. MaxSegmentLength hakkında daha fazla bilgi için bkz. MaxSegmentLength.

  NetBIOS:
  64 KB

  -Bu ne?
  64 KB

  Not: Alıcı kanallarında gönderen kanalları ve gönderen çıkışlarında TCP için 2 KB arabellekler kullanılır.
- Güvenlik çıkışları için, dağıtılmış kuyruğa alma olanağı 4000 baytlık bir arabellek ayırır.

Çıkışın ilgili parametrelerle birlikte alternatif bir arabellek döndürmesine izin verilebilir. Arama ayrıntıları için bkz. “İleti sistemi kanalları için kanal çıkış programları” sayfa 920 .

z/OS *z/OS üzerinde kanal çıkış programları yazılıyor*
z/OSiçin kanal çıkış programlarını yazmanıza ve derlemenize yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

Çıkışlar z/OS LINK ile başlatılmış gibi başlatılır:

- Yetkisiz sorun programı durumu
- Birincil adres alanı denetim kipi
- Bellek dışı kip
- Erişim dışı kayıt kipi
- 31 bit adresleme kipi

Bağlantı düzenlenen modüller, kanal başlatıcı adres alanı yordamının CSQXLIB DD deyimi tarafından belirtilen veri kümesine yerleştirilmelidir; yükleme modüllerinin adları, kanal tanımlamasında çıkış adları olarak belirtilir.

z/OSiçin kanal çıkışları yazılırken aşağıdaki kurallar geçerlidir:

- Çıkışlar çevirici ya da C olarak yazılmalıdır; C kullanılırsa, sistem çıkışları için C sistemleri programlama ortamına uygun olmalıdır; açıklamalar için bkz. z/OS C/C++ Programming Guide.
- Çıkışlar, CSQXLIB DD deyimi tarafından tanımlanan yetkili olmayan kitaplıklardan yüklenir. CSQXLIB ' de DISP=SHR varsa, kanal başlatıcısı çalışırken çıkışlar güncellenebilir. Kanal yeniden başlatıldığında yeni sürüm kullanılır.
- Çıkışların yeniden girilebilmesi ve sanal saklama alanında herhangi bir yerde çalışabilmesi gerekir.
- Çıkışlar, geri döndüğünüzde, girdide ortamı ilk durumuna getirmelidir.
- Çıkışlar, elde edilen depolama alanını serbest bırakmalı ya da sonraki bir çıkış çağrısıyla serbest bırakıldığından emin olmalıdır.

  Çağrılar arasında kalıcı olarak saklanacak depolama için, z/OS STORAGE hizmetini ya da System Programming C için 4kmalc kitaplık işlevini kullanın.

  Bu işleve ilişkin ek bilgi için bkz. 4kmalc() -- Allocate Page-Aligned Storage.
- MQCMIT ya da CSQBCMT ve MQBACK ya da CSQBBAK dışında tüm IBM MQ MQI çağrıları kullanılabilir. Bunlar MQCONN ' dan sonra (boş bir kuyruk yöneticisi adıyla) bulunmalıdır. Bu çağrılar kullanılırsa, çıkış CSQXSTUB sınırlı kod öbeğiyle düzenlenmelidir.

  Bu kuralın kural dışı durumu, güvenlik kanalı çıkışlarının kesinleştirme ve geriletme MQI çağrıları yayınlayabilmeleri. Bu tür çağrıları yayınlamak için, MQCMIT, CSQBCMT ve MQBACK ya da CSQBBAK yerine CSQXCMT ve CSQXBAK fiillerini kodlayın.
- IBM WebSphere MQ 7.0 ya da daha sonraki bir sürümdeki CSQXSTUB sınırlı kod öbeğini kullanan tüm çıkışlar, PDS-E biçimindeki bir CSQXLIB yükleme kitaplığında düzenlenmelidir.
- Sistem hizmetlerinin kullanılması, diğer kanalların bazılarının ya da tümünün işlenmesini önemli ölçüde etkileyeceği için, çıkışların beklemesine neden olan herhangi bir sistem hizmeti kullanmaması gerekir. Birçok kanal tipik olarak tek bir TCB altında çalışır. Bir çıkışta beklemeye neden olan bir şey yaparsanız ve MQXWAIT kullanmazsanız, bu tüm kanalların beklemesine neden olur. Kanalların beklemesine neden olmak herhangi bir işlevsel sorun yaratmaz, ancak performans üzerinde olumsuz bir etkisi olabilir. Çoğu SVC bekleme işlemleri içerir; bu nedenle, aşağıdaki SVC ' ler dışında bu işlemlerden kaçınmanız gerekir:
  - GETMAIN/FREEMAIN/STORAGE
  - YÜKLEME/SILME

  Bu nedenle, genel olarak SVC 'leri, PC' leri ve G/Ç ' yi önleyin. Bunun yerine MQXWAIT çağrıyı kullanın.
- Hata işlemeleri IBM MQtarafından gerçekleştirilen hata işlemeyi engelleyebileceğinden, bağladıkları alt görevler dışında, çıkışlar ESTAE ' leri ya da SPIE'leri yayınlamaz. Bu, IBM MQ ' in bir hatadan kurtulamayabileceği ya da çıkış programınızın tüm hata bilgilerini alamayabileceği anlamına gelir.
- MQXWAIT çağrısı (bkz. MQXWAIT) G/Ç ve diğer olayları bekleyen bir bekleme hizmeti sağlar; bu hizmet kullanılırsa, çıkışlar bağ yığınını kullanmamalıdır.

  G/Ç ve engelleyici olmayan olanaklar ya da ECB ' nin beklenmesini sağlamayan diğer olanaklar için, ayrı bir alt görev ATTACHed edilmeli ve bunun tamamlanması MQXWAIT tarafından beklenmelidir; bu tekniğin işlemesi nedeniyle, bu olanak yalnızca güvenlik çıkışı tarafından kullanılmalıdır.

- MQDISC MQI çağrısı, çıkış programı içinde örtük bir kesinleştirme oluşmasına neden olmaz. Kanal işlemine ilişkin kesinleştirme, yalnızca kanal protokolü gerektirdiğinde gerçekleştirilir.

IBM MQ for z/OSile birlikte aşağıdaki çıkış örnekleri sağlanır:

CSQ4BAX0
Bu örnek çevirici içinde yazılır ve MQXWAIT kullanımını gösterir.

CSQ4BCX1 ve CSQ4BCX2
Bu örnekler C harfiyle yazılır ve parametrelere nasıl erişileceğini gösterir.

CSQ4BCX3 ve CSQ4BAX3
Bu örnekler sırasıyla C ve çevirici olarak yazılır.

CSQ4BCX3 örneği (SCSQAUTH LOADLIB içinde önceden derlenen), çıkışın kendisinde herhangi bir değişiklik yapılmasına gerek olmadan çalışmalıdır. Bir LOADLIB (örneğin, MY.TEST.LOADLIB) ve SCSQAUTH (CSQ4BCX3) üyesini bu üyeye kopyalayın.

Bir istemci bağlantısında güvenlik çıkışı ayarlamak için aşağıdaki yordamı gerçekleştirin:

1. Kanal başlatıcısının kullandığı kullanıcı kimliği için geçerli bir OMVS bölümü oluşturun.

   Bu, IBM MQ for z/OS kanal başlatıcısının çıkış işlemeyi kolaylaştırmak için z/OS UNIX System Services (z/OS UNIX) yuva arabirimiyle TCP/IP ' yi kullanmasını sağlar. Bağlanan herhangi bir istemcinin kullanıcı kimliği için bir OMVS bölümü tanımlanmasının gereksiz olduğunu unutmayın.
2. Çıkış kodunun kendisinin yalnızca program denetimli bir ortamda çalıştığından emin olun.

   Bu, CHINIT adres alanına yüklenen her şeyin, program denetimli bir kitaplıktan (STEPLIB içindeki tüm kitaplıklar anlamına gelir) ve CSQXLIB ' de adı geçen tüm kitaplıklardan yüklenmesi gerektiği anlamına gelir.

   ```
   ++hlq++.SCSQANLx
   ++hlq++.SCSQMVR1
   ++hlq++.SCSQAUTH
   ```

   Bir yükleme kitaplığını program denetimli olarak ayarlamak için, aşağıdaki örneğe benzer bir komut kullanın:

   ```
   RALTER PROGRAM * ADDMEM('MY.TEST.LOADLIB'//NOPADCHK)
   ```

   Daha sonra, aşağıdaki komutu vererek program denetimli ortamı etkinleştirebilir ya da yenileyebilirsiniz:

   ```
   SETROPTS WHEN(PROGRAM) REFRESH
   ```
3. Aşağıdaki komutu vererek, LOADLIB çıkışını CSQXLIB DD ' ye (CHINIT başlatıldı yordamında) ekleyin:

   ```
   ALTER CHANNEL(xxxx) CHLTYPE(SVRCONN)SCYEXIT(CSQ4BCX3)
   ```

   Bu, adı belirtilen kanal için çıkışı etkinleştirir.
4. Dış güvenlik yöneticiniz (ESM), program denetiminde olacak diğer kitaplıkları listeler, ancak ESM ya da C kitaplıklarının hiçbirinin program denetimi altında olması gerekmediğini unutmayın.

Örnek CSQ4BCX3kullanılarak güvenlik çıkışı ayarlanmasına ilişkin ek bilgi için IBM MQ for z/OS sunucu bağlantı kanalı başlıklı konuya bakın.

CSQ4BCX4
Bu örnek C dilinde yazılır ve MQCXP ' de **RemoteProduct** ve **RemoteVersion** alanlarının kullanılmasını gösterir.

İlgili kavramlar
“IBM i üzerinde kanal çıkış programları yazılıyor” sayfa 927
IBM iiçin kanal çıkış programlarını yazmanıza ve derlemenize yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

“AIX, Linux, and Windows üzerinde kanal çıkış programları yazılıyor” sayfa 927
AIX, Linux, and Windows sistemleri için kanal çıkış programları yazmanıza yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

İlgili başvurular
IBM MQ for z/OS sunucu bağlantısı kanalı

IBM i *IBM i üzerinde kanal çıkış programları yazılıyor*
IBM iiçin kanal çıkış programlarını yazmanıza ve derlemenize yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

Çıkış, ILE C, ILE RPG ya da ILE COBOL dilinde yazılmış bir program nesnesidir. Çıkış programı adları ve kitaplıkları kanal tanımında adlandırılır.

Bir çıkış programı yaratırken ve derlerken aşağıdaki koşulları göz önünde bulundurun:

- Program, iş parçacığı güvenli hale getirilmeli ve ILE C, ILE RPG ya da ILE COBOL derleyicisiyle yaratılmalıdır. ILE RPG için THREAD (*SERIALIZE) denetim belirtimini belirlemeli ve ILE COBOL için PROCESS deyiminin THREAD seçeneği için SERIALIZE belirlemelisiniz. ILE COBOL söz konusu olduğunda, programlar iş parçacıklı IBM MQ kitaplıklarına da bağlanmalıdır: ILE C ve ILE RPG durumunda QMQM/LIBMQM_R ve AMQ0STUB_R . RPG ya da COBOL uygulamalarının iş parçacığı güvenliğine ilişkin ek bilgi için, dile ilişkin uygun Programcı Kılavuzu 'na bakın.
- IBM MQ for IBM i , teraspace desteği için çıkış programlarının etkinleştirilmesini gerektirir. (Teraspace, OS/400 V4R4' te tanıtılan bir paylaşılan bellek biçimidir.) ILE RPG ve COBOL derleyicileri için, OS/400 V4R4 ya da sonraki yayın düzeylerinde derlenen programlar etkinleştirilir. C için, programlar CRTCMOD ya da CRTBNDC komutlarında belirlenen TERASPACE (*YES *TSIFC) seçenekleriyle derlenmelidir.
- Bir göstergeyi kendi arabellek alanına döndüren bir çıkış, gösterilen nesnenin kanal çıkış programının zaman aralığının ötesinde var olduğunu doğrulamalıdır. Gösterge, program yığınındaki bir değişkenin adresi ya da program öbeğindeki bir değişkenin adresi olamaz. Bunun yerine, işaretçi sistemden alınmalıdır. Örnek, kullanıcı çıkışında yaratılan bir kullanıcı alanıdır. Program sona erdiğinde, kanal çıkış programı tarafından ayrılan herhangi bir veri alanının MCA için kullanılabilir olduğundan emin olmak için kanal çıkışının, çağıranın etkinleştirme grubunda ya da adlandırılmış bir etkinleştirme grubunda çalışması gerekir. Bunu, CRTPGM üzerindeki ACTGRP parametresini kullanıcı tanımlı bir değere ya da *CALLER değerine ayarlayarak yapın. Program bu şekilde yaratılırsa, kanal çıkış programı dinamik bellek ayırabilir ve bu belleğe MCA ' ya bir gösterge aktarabilir.

İlgili kavramlar
“AIX, Linux, and Windows üzerinde kanal çıkış programları yazılıyor” sayfa 927
AIX, Linux, and Windows sistemleri için kanal çıkış programları yazmanıza yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

“z/OS üzerinde kanal çıkış programları yazılıyor” sayfa 924
z/OSiçin kanal çıkış programlarını yazmanıza ve derlemenize yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

ALW *AIX, Linux, and Windows üzerinde kanal çıkış programları yazılıyor*
AIX, Linux, and Windows sistemleri için kanal çıkış programları yazmanıza yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

“AIX, Linux, and Windows üzerinde çıkışlar ve kurulabilir hizmetler yazılıyor” sayfa 897içinde açıklanan yönergeleri izleyin. Uygun olduğunda, aşağıdaki kanal çıkışına özgü bilgileri kullanın:

Çıkış C dilinde yazılmalı ve Windowsüzerinde bir DLL olmalıdır.

Çıkışta kukla bir MQStart () yordamı tanımlayın ve kitaplıkta giriş noktası olarak `MQStart` değerini belirtin. Şekil 106 sayfa 928 içinde, programınıza ilişkin bir girişin nasıl ayarlaneceği gösterilmektedir:

```
#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
}
```

Şekil 106. Kanal çıkışına ilişkin örnek kaynak kodu

Windows için Visual C + + kullanarak kanal çıkışlarını yazarken, kendi DEF dosyanızı yazmanız gerekir. Şekil 107 sayfa 928içinde nasıl gösterildiğine bir örnek. Kanal çıkış programlarının yazılması hakkında daha fazla bilgi için bkz. “Kanal çıkış programları yazılıyor” sayfa 923.

```
EXPORTS
ChannelExit
```

Şekil 107. Windows için örnek DEF dosyası

İlgili kavramlar

“IBM i üzerinde kanal çıkış programları yazılıyor” sayfa 927
IBM iiçin kanal çıkış programlarını yazmanıza ve derlemenize yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

“z/OS üzerinde kanal çıkış programları yazılıyor” sayfa 924
z/OSiçin kanal çıkış programlarını yazmanıza ve derlemenize yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

Kanal güvenliği çıkış programları
Bir kanalın diğer ucundaki ortağın gerçek olduğunu doğrulamak için güvenlik çıkış programlarını kullanabilirsiniz. Bu kimlik doğrulama olarak bilinir.

Bir kanalın güvenlik çıkışı kullanması gerektiğini belirtmek için, kanal tanımının **SCYEXIT** alanında çıkış adını belirtin.

Not: Kimlik doğrulaması, kanal kimlik doğrulama kayıtlarıyla da gerçekleştirilebilir. Kanal kimlik doğrulama kayıtları , belirli kullanıcılardan ve kanallardan kuyruk yöneticilerine erişimin önlenmesi ve uzak kullanıcıların IBM MQ kullanıcı tanıtıcılarıyla eşlenmesi konusunda büyük esneklik sağlar. TLS desteği, kullanıcılarınızın kimliğini doğrulamak ve verileriniz için şifreleme ve veri bütünlüğü denetimleri sağlamak için IBM MQ tarafından da sağlanır. TLS hakkında daha fazla bilgi için bkz. IBM MQ içinde TLS güvenlik iletişim kuralları. Ancak, yine de daha gelişmiş (ya da farklı) güvenlik işleme biçimlerine ve diğer denetim ve güvenlik bağlamı oluşturma tiplerine gereksinim duyarsanız, güvenlik çıkışları yazmayı düşünebilirsiniz.

Konu ve Veren DN öznitelikleri aşağıdaki kanal durumu özniteliklerinde görüntülenir:

- SSLPEER (PCF seçici MQCACH_SSL_SHORT_PEER_NAME)
- SSLCERTI (PCF seçici MQCACH_SSL_CERT_ISSUER_NAME)

Bu değerler, kanal durumu komutlarının yanı sıra, aşağıda gösterildiği gibi, kanal güvenliği çıkışlarına geçirilen veriler tarafından döndürülür:

- MQCD SSLPeerNamePtr
- MQCXP SSLRemCertIssNamePtr

Bir güvenlik çıkışı C ya da Javabiçiminde yazılabilir.

Kanal güvenliği çıkış programları, MCA ' nın işleme döngüsünde aşağıdaki yerlerde çağrılır:

- MCA başlatma ve sonlandırma sırasında.
- Kanal başlatıldığında ilk veri anlaşması bittikten hemen sonra. Kanalın alıcı ya da sunucu ucu, uzak uçtaki güvenlik çıkışına teslim edilecek bir ileti sağlayarak uzak uçla bir güvenlik iletisi değiş tokuşu başlatabilir. Bunu yapmayı da reddedebilir. Uzak uçtan alınan güvenlik iletilerini işlemek için çıkış programı yeniden başlatılır.
- Kanal başlatıldığında ilk veri anlaşması bittikten hemen sonra. Kanalın gönderen ya da istekte bulunan ucu, uzak uçtan alınan bir güvenlik iletisini işler ya da uzak uçtan alınamadığında bir güvenlik değiştokuşu başlatır. Alınabilecek sonraki tüm güvenlik iletilerini işlemek için çıkış programı yeniden başlatılır.

Bir istek kanalı hiçbir zaman MQXR_INIT_SEC ile çağrılmaz. Kanal, sunucuya bir güvenlik çıkış programı olduğunu bildirir ve sunucu bir güvenlik çıkışı başlatma fırsatına sahip olur. Bir değeri yoksa, istekte bulunana bilgi verir ve çıkış programına sıfır uzunluklu bir akış döndürülür.

Not: Sıfır uzunluklu güvenlik iletileri göndermekten kaçının.

Güvenlik çıkış programları tarafından değiş tokuş edilen verilere ilişkin örnekler, Şekil 108 sayfa 930 - Şekil 111 sayfa 932arasındaki şekillerde gösterilmiştir. Bu örnekler, alıcının güvenlik çıkışını ve gönderenin güvenlik çıkışını içeren olayların sırasını gösterir. Şekillerdeki ardışık satırlar zamanın geçişini temsil eder. Bazı durumlarda, alıcı ve gönderendeki olaylar ilintili değildir ve bu nedenle aynı anda ya da farklı zamanlarda oluşabilir. Diğer durumlarda, bir çıkış programındaki bir olay, daha sonra diğer çıkış programında gerçekleşen tamamlayıcı bir olayla sonuçlanır. Örneğin, Şekil 108 sayfa 930içinde:

1. Alıcı ve gönderen her biri MQXR_INIT ile çağrılır, ancak bu çağrılar ilintili değildir ve bu nedenle aynı anda ya da farklı zamanlarda oluşabilir.
2. Alıcı daha sonra MQXR_INIT_SEC ile çağrılır, ancak gönderen çıkışında tamamlayıcı olay gerektirmeyen MQXCC_OK değerini döndürür.
3. Gönderen daha sonra MQXR_INIT_SEC ile çağrılır. Bu, alıcının MQXR_INIT_SEC ile çağrılmasıyla ilintilendirilmez. Gönderen, alıcı çıkışında tamamlayıcı bir olaya neden olan MQXCC_SEND_SEC_MSG öğesini döndürür.
4. Alıcı daha sonra MQXR_SEC_MSG ile çağrılır ve gönderen çıkışında tamamlayıcı bir olaya neden olan MQXCC_SEND_SEC_MSG değerini döndürür.
5. Gönderen daha sonra MQXR_SEC_MSG ile çağrılır ve alıcı çıkışında tamamlayıcı olay gerektirmeyen MQXCC_OK döndürür.

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	

Şekil 108. Gönderen tarafından başlatılan sözleşme ile değiş tokuş

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

Şekil 109. Sözleşme olmadan gönderen tarafından başlatılan değiş tokuş

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

Şekil 110. Sözleşmeyle alıcı tarafından başlatılan değiş tokuş

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	

Şekil 111. Sözleşme olmadan alıcı tarafından başlatılan değiş tokuş

Kanal güvenliği çıkış programına, güvenlik çıkışı tarafından oluşturulan iletim üstbilgileri dışında, güvenlik verilerini içeren bir aracı arabelleği geçirilir. Bu veriler, kanalın herhangi bir ucunun güvenlik doğrulaması gerçekleştirebilmesi için uygun veriler olabilir.

İleti kanalının hem gönderen hem de alan ucundaki güvenlik çıkış programı, herhangi bir çağrıya iki yanıt kodundan birini döndürebilir:

- Güvenlik değiş tokuşu hatasız sona erdi
- Kanalı engelle ve kapat

Not:

1. Kanal güvenliği çıkışları genellikle çiftler halinde çalışır. Uygun kanalları tanımlarken, uyumlu çıkış programlarının kanalın her iki ucu için de adlandırıldığından emin olun.
2. IBM i IBM i içinde, `Use adopted authority` (USEADPAUT = *YES) ile derlenen güvenlik çıkış programları QMQM ya da QMQMADM yetkisini kabul edebilir. Çıkışın sisteminiz için bir güvenlik riski oluşturması için bu özelliği kullanmamasına dikkat edin.
3. Kanalın diğer ucunun sertifika sağladığı bir TLS kanalında, güvenlik çıkışı, SSLPeerNamePtr ile erişilen MQCD alanında bu sertifikanın konusunun Ayırt Edici Adını ve SSLRemCertIssNamePtr ile erişilen MQCXP alanındaki sertifika verenin Ayırt Edici Adını alır. Bu adın konabileceği kullanımlar şunlardır:
   - TLS kanalı üzerinden erişimi kısıtlamak için.
   - MQCD.MCAUserIdentifier .

İlgili kavramlar

Aktarım Katmanı Güvenliği (TLS) kavramları

İlgili başvurular

Kanal kimlik doğrulama kayıtları

Güvenlik çıkışı yazılması

Güvenlik çıkışı iskelet kodunu kullanarak bir güvenlik çıkışı yazabilirsiniz.

Şekil 112 sayfa 933 içinde bir güvenlik çıkışına nasıl yazılacağı gösterilir.

```
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 */

}
```

Şekil 112. Güvenlik çıkışı iskelet kodu

Standart IBM MQ Giriş Noktası MQStart var olmalıdır, ancak herhangi bir işlevi gerçekleştirmek için gerekli değildir. İşlevin adı (bu örnekteEntryPoint) değiştirilebilir, ancak kitaplık derlenip bağlandığında işlev dışa aktarılmalıdır. Önceki örnekte olduğu gibi, işaretçiler pChannelExitParms PMQCXP 'ye ve pChannelTanımı PMQCD' ye çevrilmelidir. Kanal çıkışlarının çağrılmasıyla ve değiştirgelerin kullanılmasıyla ilgili genel bilgi için MQ_CHANNEL_EXITbaşlıklı konuya bakın. Bu parametreler bir güvenlik çıkışında aşağıdaki gibi kullanılır:

PMQVOID pChannelExitParms

giriş/çıkış

MQCXP yapısına ilişkin gösterge-alanlara erişmek için PMQCXP ' ye dönüşür. Bu yapı, Çıkış ve MCA arasında iletişim kurmak için kullanılır. MQCXP ' de aşağıdaki alanlar özellikle Güvenlik Çıkışları için ilginizi çekmektedir:

ExitReason
Güvenlik değiştokuşunda geçerli durumu Güvenlik Çıkışı 'ya bildirir ve hangi işlemin yapılacağına karar verilirken kullanılır.

ExitResponse
Güvenlik değiştokuşunda bir sonraki aşamayı belirleyen MCA ' ya verilen yanıt.

ExitResponse2
MCA ' nın Güvenlik Çıkışı yanıtını nasıl yorumladığını yönetmek için ek denetim işaretleri.

ExitUserAlanı
Çağrılar arasındaki durumu korumak için Güvenlik Çıkışı tarafından kullanılabilen 16 bayt (üst sınır) depolama alanı.

ExitData
Kanal tanımının SCYDATA alanında belirlenen verileri içerir (32 bayt sağa boşluklarla doldurulur).

PMQVOID pChannelTanımlaması
giriş/çıkış

MQCD yapısına ilişkin gösterge-alanlara erişmek için PMQCD ' ye dönüşür. Bu parametre, kanalın tanımını içerir. MQCD ' deki aşağıdaki alanlar özellikle Güvenlik Çıkışları için ilginizi çekmektedir:

ChannelName
Kanal adı (20 bayt sağa boşluklarla doldurulur).

ChannelType
Kanal tipini tanımlayan bir kod.

MCA Kullanıcı Tanıtıcısı
Bu üç alandan oluşan grup, kanal tanımında belirtilen MCAUSER alanının değeriyle ilk kullanıma hazırlanır. Bu alanlarda Güvenlik Çıkışı tarafından belirlenen herhangi bir kullanıcı kimliği erişim denetimi için kullanılır (SDR, SVR, CLNTCONN ya da CLUSSDR kanalları için geçerli değildir).

MCAUserIdentifier
Tanıtıcının ilk 12 baytı sağa boşluklarla doldurulduu.

LongMCAUserIdPtr
MCAUserIdentifier' dan daha öncelikli olarak, tam uzunluklu tanıtıcıyı (boş sonlandırılmış olarak garanti edilmez) içeren bir arabellek göstergesi kullanılır.

LongMCAUserIdLength
LongMCAUserIdPtr tarafından gösterilen dizginin uzunluğu- LongMCAUserIdPtr ayarlandıysa ayarlanmalıdır.

Uzak Kullanıcı Tanıtıcısı
Yalnızca CLNTCONN/SVRCONN kanal çiftleri için geçerlidir. Herhangi bir CLNTCONN Güvenlik Çıkışı tanımlanmadıysa, bu üç alan istemci MCA tarafından kullanıma hazırlanır; bu nedenle, kimlik doğrulaması için ve MCA Kullanıcı Tanıtıcısı belirtilirken bir SVRCONN Güvenlik Çıkışı tarafından kullanılabilecek istemci ortamından bir kullanıcı kimliği içerebilir. Bir CLNTCONN Güvenlik Çıkışı tanımlandıysa, bu alanlar ilk kullanıma hazırlanmaz ve CLNTCONN Güvenlik Çıkışı tarafından ayarlanabilir ya da güvenlik iletileri istemden sunucuya kullanıcı kimliği geçirmek için kullanılabilir.

RemoteUserTanıtıcısı
Tanıtıcının ilk 12 Baytı sağa boşluklarla doldurulduu.

LongRemoteUserIdPtr
Tam uzunluklu tanıtıcıyı içeren (boş sonlandırılmış olarak garanti edilmeyen) bir arabelleğe ilişkin gösterge, RemoteUserTanıtıcısı 'na göre önceliklidir.

LongRemoteUserIdLength (Uzun Uzak) Uzunluğu
LongRemoteUserIdPtr belirlendiyse, LongRemoteUserIdPtr ile gösterilen dizginin uzunluğu ayarlanmalıdır.

PMQLONG pDataUzunluğu
giriş/çıkış

MQLONG göstergesi. Güvenlik çıkışının çağrılmasının ardından AgentBuffer içindeki Güvenlik Çıkışının uzunluğunu içerir. Bir Güvenlik Çıkışı tarafından, AgentBuffer ya da ExitBufferiçinde gönderilen herhangi bir iletinin uzunluğuna ayarlanmalıdır.

PMQLONG pAgentBufferLength
Giriş

MQLONG göstergesi. Güvenlik çıkışının çağrılmasına ilişkin AgentBuffer içinde bulunan verilerin uzunluğu.

PMQVOID pAgentArabelleği
giriş/çıkış

Güvenlik Çıkışı çağrıldığında, bu, iş ortağı çıkışından gönderilen herhangi bir iletiyi işaret eder. MQCXP yapısındaki ExitResponse2 için MQXR2_USE_AGENT_BUFFER işareti ayarlandıysa (varsayılan), bir Güvenlik Çıkışı 'nın bu parametreyi gönderilmekte olan ileti verilerini gösterecek şekilde ayarlaması gerekir.

PMQLONG pExitBufferLength
giriş/çıkış

MQLONG göstergesi. Bu parametre, bir Güvenlik Çıkışı 'nın ilk çağrılmasında 0 olarak ilk kullanıma hazırlanır ve döndürülen değer, bir güvenlik değiştokuşu sırasında Güvenlik Çıkışı çağrıları arasında tutulur.

PMQPTR pExitBufferAddr
giriş/çıkış

Bu parametre, bir Güvenlik Çıkışı 'nın ilk çağrılmasında boş değerli bir işaretçiyle kullanıma hazırlanır ve güvenlik değiştokuşu sırasında Güvenlik Çıkışı çağrıları arasında döndürülen değer korunur. MQCXP yapısındaki ExitResponse2 içinde MQXR2_USE_EXIT_BUFFER işareti ayarlandıysa, bir Güvenlik Çıkışının bu parametreyi gönderilmekte olan ileti verilerini gösterecek şekilde ayarlaması gerekir.

CLNTCONN/SVRCONN kanal çiftlerinde ve diğer kanal çiftlerinde tanımlanan güvenlik çıkışları arasındaki davranış farklılıkları
Güvenlik çıkışları her kanal tipinde tanımlanabilir. Ancak, CLNTCONN/SVRCONN kanal çiftlerinde tanımlanan güvenlik çıkışlarının davranışı, diğer kanal çiftlerinde tanımlanan güvenlik çıkışlarından biraz farklıdır.

CLNTCONN kanalındaki bir Güvenlik Çıkışı, ortak bir SVRCONN çıkışı tarafından işlenmek üzere kanal tanımındaki Uzak Kullanıcı Tanıtıcısı 'nı ya da SVRCONN Güvenlik Çıkışı tanımlanmadıysa ve SVRCONN ' un MCAUSER alanı ayarlanmadıysa OAM yetkilendirmesi için ayarlayabilir.

Herhangi bir CLNTCONN Güvenlik Çıkışı tanımlanmadıysa, kanal tanımındaki Uzak Kullanıcı Tanıtıcısı, istemci MCA tarafından istemci ortamından (boş olabilir) bir kullanıcı kimliğine ayarlanır.

SVRCONN Güvenlik Çıkışı MQXCC_OK için bir ExitResponse döndürdüğünde, CLNTCONN ve SVRCONN kanal çiftinde tanımlanan Güvenlik Çıkışları arasındaki güvenlik değiş tokuşu başarıyla tamamlanır. Değiş tokuşu başlatan Güvenlik Çıkışı MQXCC_OK ' un ExitResponse öğesini döndürdüğünde, diğer kanal çiftleri arasındaki bir güvenlik değiş tokuşu başarıyla tamamlanır.

Ancak, MQXCC_SEND_AND_REQUEST_SEC_MSG ExitResponse kodu, güvenlik değişiminin sürdürülmesini zorlamak için kullanılabilir: MQXCC_SEND_AND_REQUEST_SEC_MSG ExitResponse öğesi bir CLNTCONN ya da SVRCONN Güvenlik Çıkışı tarafından döndürülürse, ortak çıkış bir güvenlik iletisi göndererek yanıt vermelidir (MQXCC_OK ya da boş bir yanıt değil) ya da kanal sonlandırılır. Diğer kanal tiplerinde tanımlanan Güvenlik Çıkışları için, ortak Güvenlik Çıkışı 'ndan bir MQXCC_SEND_AND_REQUEST_SEC_MSG ' ye yanıt olarak MQXCC_OK ExitResponse döndürüldü; bu, güvenlik değişiminin, kanal sonlandırılmadan boş bir yanıt döndürülmüş gibi devam edilmesiyle sonuçlanır.

SSPI güvenlik çıkışı
IBM MQ for Windows , SSPI (Security Services Programming Interface; Güvenlik Hizmetleri Programlama Arabirimi) kullanarak IBM MQ kanalları için kimlik doğrulaması sağlayan bir güvenlik çıkışı sağlar. SSPI, Windows' in tümleşik güvenlik olanaklarını sağlar.

Bu güvenlik çıkışı hem IBM MQ istemcisi, hem de IBM MQ sunucusu içindir.

Güvenlik paketleri security.dll ya da secur32.dll' den yüklenir. Bu DLL ' ler işletim sisteminizle birlikte sağlanır.

Tek yönlü kimlik doğrulama, NTLM kimlik doğrulama hizmetleri kullanılarak Windowsüzerinde sağlanır. İki yönlü kimlik doğrulaması, Windows 2000' de Kerberos kimlik doğrulama hizmetleri kullanılarak sağlanır.

Güvenlik çıkış programı kaynak ve nesne biçiminde sağlanır. Nesne kodunu olduğu gibi kullanabilir ya da kaynak kodu başlangıç noktası olarak kullanarak kendi kullanıcı çıkış programlarınızı yaratabilirsiniz. SSPI güvenlik çıkışının nesne ya da kaynak kodunu kullanma hakkında daha fazla bilgi için bkz. “Windows üzerinde SSPI güvenlik çıkışının kullanılması” sayfa 1086

Kanal gönderme ve alma çıkış programları
Veri sıkıştırma ve açma gibi görevleri gerçekleştirmek için gönderme ve alma çıkışlarını kullanabilirsiniz. Art arda çalıştırılacak gönderme ve alma çıkış programlarının listesini belirleyebilirsiniz.

Kanal gönderme ve alma çıkış programları, MCA ' nın işleme döngüsünde aşağıdaki yerlerde çağrılır:

- Gönderme ve alma çıkış programları MCA başlatılırken kullanıma hazırlama ve MCA sonlandırma sırasında sona erdirme için çağrılır.
- Gönderme çıkış programı, bağlantı üzerinden bir iletim gönderilmeden hemen önce, bir ileti aktarımı iletiminin gönderileceği sona bağlı olarak, kanalın bir ucunda ya da başka bir ucunda çağrılır. Not 4, ileti kanalları iletileri tek yönde gönderse de çıkışların neden her iki yönde de kullanılabilir olduğunu açıklar.
- Alma çıkış programı, bağlantıdan bir iletim alındıktan hemen sonra, bir ileti aktarımı iletiminin alındığı sona bağlı olarak, kanalın bir ucunda ya da başka bir ucunda çağrılır. Not 4, ileti kanalları iletileri tek yönde gönderse de çıkışların neden her iki yönde de kullanılabilir olduğunu açıklar.

Bir ileti aktarımı için çok sayıda iletim olabilir ve bir ileti, alma sonunda ileti çıkışına ulaşmadan önce gönderme ve alma çıkış programlarının birçok yinelemesi olabilir.

Kanal gönderme ve alma çıkış programlarına, iletişim bağından gönderilen ya da alınan iletim verilerini içeren bir aracı arabelleği geçirilir. Gönderme çıkış programları için, arabelleğin ilk 8 baytı MCA tarafından kullanılmak üzere ayrılmıştır ve değiştirilmemelidir. Program farklı bir arabellek döndürürse, bu ilk 8 bayt yeni arabellekte bulunmalıdır. Çıkış programlarına sunulan verilerin biçimi tanımlı değil.

Gönderme ve alma çıkış programları tarafından iyi bir yanıt kodu döndürülmelidir. Diğer yanıtlar MCA ' nın olağandışı bitmesine (olağandışı bitmesine) neden olur.

Not: Gönderme ya da alma çıkışından uyumlulaştırma noktası içinde bir MQGET, MQPUT ya da MQPUT1 çağrısı yürütmeyin.

Not:

1. Gönderme ve alma çıkışları genellikle çiftler halinde çalışır. Örneğin, gönderme çıkışı verileri sıkıştırabilir ve alma çıkışı sıkıştırmayı çözebilir ya da gönderme çıkışı verileri şifreleyebilir ve alma çıkışı verilerin şifresini çözebilir. Uygun kanalları tanımlarken, uyumlu çıkış programlarının kanalın her iki ucu için de adlandırıldığından emin olun.
2. Kanal için sıkıştırma açıksa, çıkışlar sıkıştırılmış verilerden geçirilir.
3. Kanal gönderme ve alma çıkışları, uygulama verileri dışındaki ileti bölümleri (örneğin, durum iletileri) için çağrılabilir. Bunlar, başlatma iletişim kutusu sırasında ya da güvenlik denetimi aşamasında çağrılmazlar.
4. İleti kanalları iletileri tek yönlü gönderse de, kalp atımları ve toplu işleme sonu gibi kanal kontrol verileri, her iki yönde de akar ve bu çıkışlar her iki yönde de kullanılabilir. Ancak, ilk kanal başlatma veri akışlarından bazıları, herhangi bir çıkış tarafından işlenmekten muaf tutulmuştur.
5. Gönderme ve alma çıkışlarının sırasız olarak çağrılabileceği durumlar vardır; örneğin, bir dizi çıkış programı çalıştırıyorsanız ya da güvenlik çıkışlarını da çalıştırıyorsanız. Daha sonra, verileri işlemek için alma çıkışı ilk kez çağrıldığında, ilgili gönderme çıkışından geçmemiş verileri alabilir. Alma çıkışı, önce gerekli olup olmadığını denetlemeden, örneğin sıkıştırma açma işlemini gerçekleştirdiyse, sonuçlar beklenmeyen olur.

Gönderme ve alma çıkışlarınızı, alma çıkışının aldığı verilerin ilgili gönderme çıkışı tarafından işlenip işlenmediğini denetleyebileceği şekilde kodlamanız gerekir. Bunu yapmanın önerilen yolu, çıkış programlarınızı aşağıdaki şekilde kodlamak:

- Gönderme çıkışı, dokuzuncu veri baytı değerini 0 olarak ayarlar ve işlemi gerçekleştirmeden önce tüm verileri 1 bayt boyunca kaydırır. (İlk 8 bayt MCA tarafından kullanılmak üzere ayrılmıştır.)
- Alma çıkışı, bayt 9 'da 0 olan verileri alırsa, verilerin gönderme çıkışından geldiğini bilir. 0 'ı kaldırır, tamamlayıcı işlemi gerçekleştirir ve sonuçtaki verileri 1 bayt geriye kaydırır.
- Alma çıkışı, bayt 9 'da 0 dışında bir değer içeren veriler alırsa, gönderme çıkışının çalışmadığını varsayar ve verileri arayanlara değişmeden geri gönderir.

Güvenlik çıkışları kullanılırken, kanal güvenlik çıkışı tarafından sona erdirilirse, ilgili alma çıkışı olmadan bir gönderme çıkışı çağrılabilir. Bu sorunu önlemenin bir yolu, MQCD.SecurityUserData ya da MQCD.SendUserData' da (örneğin, çıkış kanalı sonlandırmaya karar verdiğinde) bir işaret ayarlamak için güvenlik çıkışını kodlamaktır. Daha sonra, gönderme çıkışının bu alanı denetlemesi ve yalnızca işaret ayarlanmamışsa verileri işlemesi gerekir. Bu denetim, gönderme çıkışının gereksiz yere verileri değiştirmesini önler ve güvenlik çıkışının değiştirilmiş veriler alması durumunda oluşabilecek dönüştürme hatalarını önler.

Kanal gönderme çıkış programları-alan ayırma
İletiden önce verileri dönüştürmek için gönderme ve alma çıkışlarını kullanabilirsiniz. Kanal gönderme çıkış programları, iletim arabelleğinde yer ayırarak dönüştürmeye ilişkin kendi verilerini ekleyebilir.

Bu veriler, alma çıkış programı tarafından işlenir ve arabellekten kaldırılır. Örneğin, verileri şifrelemek ve şifre çözme için bir güvenlik anahtarı eklemek isteyebilirsiniz.

Alanı nasıl ayırdığınız ve kullandığınız

Başlatma için gönderme çıkış programı çağrıldığında, MQXCP ' nin *ExitSpace* alanını ayrılacak byte sayısına ayarlayın. Ayrıntılar için bkz. MQCXP . *ExitSpace* yalnızca kullanıma hazırlama sırasında ayarlanabilir; yani, *ExitReason* MQXR_INIT değerine sahip olduğunda. *ExitReason* MQXR_XMIT olarak ayarlanmadan hemen önce gönderme çıkışı çağrıldığında, *ExitSpace* byte 'ları iletim arabelleğinde ayrılır. *ExitSpace* , z/OSüzerinde desteklenmez.

Gönderme çıkışının ayrılmış alanın tümünü kullanması gerekmez. *ExitSpace* bayttan az ya da iletim arabelleği dolu değilse, çıkış ayrılan miktardan daha fazla kullanabilir. *ExitSpace*değerini ayarlarken, iletim arabelleğindeki ileti verileri için en az 1 KB bırakmanız gerekir. Çok miktarda veri için ayrılmış alan kullanılırsa kanal performansı etkilenebilir.

İletim arabelleği normalde 32KB uzunluğudur. Bununla birlikte, kanal TLS kullanıyorsa, iletim arabelleği boyutu, RFC 6101 ve ilgili TLS standartları ailesi tarafından tanımlanan kayıt uzunluğu üst sınırına sığması için 15.352 bayta düşürülür. IBM MQtarafından kullanılmak üzere 1024 bayt daha ayrılmıştır, bu nedenle gönderme çıkışları tarafından kullanılabilen iletim arabelleği alanı üst sınırı 14.328 bayttır.

Kanalın alıcı ucunda ne oluyor?

Kanal alma çıkış programları, ilgili gönderme çıkışlarıyla uyumlu olacak şekilde ayarlanmalıdır. Alma çıkışları, ayrılmış alandaki bayt sayısını bilmeli ve o alandaki verileri kaldırmalıdır.

Çoklu gönderme çıkışları

Art arda çalıştırılacak gönderme ve alma çıkış programlarının listesini belirleyebilirsiniz. IBM MQ , tüm gönderme çıkışları tarafından ayrılan alan için bir toplam tutar. Bu toplam alan, iletim arabelleğinde ileti verileri için en az 1 KB bırakmalıdır.

Aşağıdaki örnek, art arda çağrılan üç gönderme çıkışına nasıl yer ayrıldığını göstermektedir:

1. Başlatma için çağrıldığında:
   - A çıkışını gönder: 1 KB.

- B çıkışını gönder 2 KB ayırıyor.
- C çıkışını gönder 3 KB ayırsın.

2. İletim büyüklüğü üst sınırı 32 KB ve kullanıcı verileri 5 KB uzunluğudur.
3. Çıkış A, 5 KB veri ile çağrılır; 5 KB B ve C çıkışları için ayrıldığından 27 KB ' ye kadar kullanılabilir. Çıkış A, ayırdığı miktara 1 KB ekler.
4. Çıkış B 6 KB veri ile çağrılır; 3 KB C çıkışı için ayrıldığı için 29 KB ' ye kadar kullanılabilir. Çıkış B, ayırdığı 2 KB ' den daha küçük 1 KB ekler.
5. 7 KB veri ile C çıkışı çağrılır; 32 KB ' ye kadar kullanılabilir. Çıkış C, ayırdığı 3 KB ' den daha fazla 10Kekler. Toplam veri miktarı (17 KB) 32 KB üst sınırından az olduğundan, bu tutar geçerlidir.

TLS kullanan bir kanal için iletim arabelleği büyüklüğü üst sınırı, 32KBdeğil, 15,352 bayttır. Bunun nedeni, temeldeki güvenli yuva iletim kesimlerinin 16KB ile sınırlı olması ve TLS kayıt ek giderleri için alanın bir kısmının gerekli olmasıdır. IBM MQtarafından kullanılmak üzere 1024 bayt daha ayrılmıştır, bu nedenle gönderme çıkışları tarafından kullanılabilen iletim arabelleği alanı üst sınırı 14.328 bayttır.

Kanal iletisi çıkış programları
Kanal ileti çıkışını, gelen kullanıcı kimlikleri, ileti verileri dönüştürme, günlük kaydı ve başvuru iletisi işleme gibi bağlantıda şifreleme, doğrulama ya da yerine koyma gibi görevleri gerçekleştirmek için kullanabilirsiniz. Art arda çalıştırılacak ileti çıkış programlarının listesini belirtebilirsiniz.

Kanal iletisi çıkış programları, MCA ' nın işleme döngüsünde aşağıdaki yerlerde çağrılır:

- MCA başlatma ve sonlandırma sırasında
- Gönderen bir MCA bir MQGET çağrısı yayınladıktan hemen sonra
- Alan MCA ' dan önce bir MQPUT çağrısı

İleti çıkışına, iletim kuyruğu üstbilgisi MQXQH ' yi ve kuyruktan alınan uygulama iletisi metnini içeren bir aracı arabelleği geçirilir. MQXQH biçimi MQXQH-İletim kuyruğu üstbilgisindeverilmiştir.

Başvuru iletilerini (yani, gönderilecek başka bir nesneyi gösteren bir üstbilgi içeren iletileri) kullanırsanız, ileti çıkışı MQRMH üstbilgisini tanır. Nesneyi tanımlar, uygun olan şekilde alır ve üstbilginin sonuna ekler ve alan MCA 'ya iletilmesi için MCA' ya iletir. Alan MCA ' da, başka bir ileti çıkışı bu iletinin bir başvuru iletisi olduğunu algılar, nesneyi çıkarır ve üstbilgiyi hedef kuyruğa iletir. Başvuru iletileri ve bunları işleyen bazı örnek ileti çıkışları hakkında daha fazla bilgi için bkz. “Başvuru iletileri” sayfa 762 ve “Başvuru İletisi örneklerinin çalıştırılması” sayfa 1059 .

İleti çıkışları aşağıdaki yanıtları döndürebilir:

- İletiyi gönder (GET exit). İleti çıkış tarafından değiştirilmiş olabilir. (MQXCC_OK değerini döndürür.)
- İletiyi kuyruğa koyun (PUT çıkışı). İleti çıkış tarafından değiştirilmiş olabilir. (MQXCC_OK değerini döndürür.)
- İletiyi işleme. İleti, MCA tarafından teslim edilmemiş ileti kuyruğuna yerleştirilir.
- Kanalı kapat.
- MCA ' nın olağandışı bitmesine neden olan hatalı dönüş kodu.

Not:

1. İleti parçalara bölünse bile, aktarılan her eksiksiz ileti için ileti çıkışları bir kez çağrılır.
2. Linux AIX AIX ya da Linuxüzerinde bir ileti çıkışı sağlarsanız, kullanıcı kimliklerinin küçük harfli karakterlere otomatik olarak dönüştürülmesi (buradaaçıklanmıştır) çalışmaz.
3. Bir çıkış, MCA ' nın kendisiyle aynı iş parçacığında çalışır. Aynı bağlantı tanıtıcısını kullandığı için MCA ile aynı iş birimi (UOW) içinde de çalışır. Bu nedenle, eşitleme noktası altında yapılan tüm çağrılar, toplu işin sonunda kanal tarafından kesinleştirilebilir ya da geri çekilebilir. Örneğin, bir kanal iletisi çıkış programı diğerine bildirim iletileri gönderebilir ve bu iletiler yalnızca özgün iletiyi içeren toplu iş kesinleştirildiğinde kuyruğa gönderilir.

Bu nedenle, bir kanal iletisi çıkış programından eşitleme noktası MQI çağrıları verebilirsiniz.

İleti çıkışının dışında ileti dönüştürme
İleti çıkışını çağırmadan önce, alan MCA, iletide bazı dönüştürmeler gerçekleştirir. Bu konuda, dönüştürmeleri gerçekleştirmek için kullanılan algoritmalar açıklanmaktadır.

Hangi üstbilgilerin işlendiği

İleti çıkışı çağrılmadan önce, alıcının MCA ' sında bir dönüştürme yordamı çalışır. Dönüştürme yordamı, iletinin başında MQXQH üstbilgisiyle başlar. Dönüştürme yordamı daha sonra MQXQH ' yi izleyen zincirli üstbilgiler aracılığıyla işleyerek gerektiğinde dönüştürme gerçekleştirir. Zincirleme üstbilgiler, alıcının ileti çıkışına geçirilen MQCXP verilerinin HeaderLength değiştirgesinde bulunan göreli konumu aşabilir. Aşağıdaki üstbilgiler yerinde dönüştürülür:

- MQXQH (biçim adı " `MQXMIT` ")
- MQMD (bu üstbilgi MQXQH ' nin bir parçasıdır ve biçim adı yoktur)
- MQMDE (biçim adı " `MQHMDE` ")
- MQDH (biçim adı " `MQHDIST` ")
- MQWIH (biçim adı " `MQHWIH` ")

Aşağıdaki üstbilgiler dönüştürülmez, ancak MCA zincirleme üstbilgileri işlemeye devam ettikçe bu üstbilgiler adım adım atilir:

- MQDLH (biçim adı " `MQDEAD` ")
- Üç karakterle başlayan biçim adları olan herhangi bir üstbilgi 'MQH' (örneğin, " `MQHRF` ") aksi belirtilmemiş olanlar

Üstbilgilerin nasıl işlendiği

Her IBM MQ üstbilgisinin Biçim parametresi MCA tarafından okunur. Format parametresi, üstbilgi içinde 8 bayttır ve bir ad içeren 8 tek baytlık karakterdir.

Daha sonra MCA, her üstbilgiyi izleyen verileri adlandırılmış tipte olarak yorumlar. Biçim, IBM MQ veri dönüştürmesi için uygun bir üstbilgi tipinin adıysa, dönüştürülür. MQ olmayan verileri (örneğin, MQFMT_NONE ya da MQFMT_STRING) gösteren başka bir adsa, MCA üstbilgileri işlemeyi durdurur.

MQCXP HeaderLengthnedir?

Bir ileti çıkışına sağlanan MQCXP verilerindeki HeaderLength değiştirgesi, iletinin başındaki MQXQH (MQMD ' yi içerir), MQMDE ve MQDH üstbilgilerinin toplam uzunluğudur. Bu üstbilgiler, 'Biçim' adları ve uzunlukları kullanılarak zincirlenmektedir.

MQWIH

Zincirleme üstbilgiler, HeaderLength uzunluğunun ötesine kullanıcı verileri alanına genişletilebilir. Varsa, MQWIH üstbilgisi, HeaderLengthüstbilgisinin ötesinde görünen üstbilgilerden biridir.

Zincirleme üstbilgilerde bir MQWIH üstbilgisi varsa, alıcının ileti çıkışı çağrılmadan önce bu üstbilgi yerinde dönüştürülür.

Kanal iletisi yeniden deneme çıkış programı
Kanal iletisi-hedef kuyruğu açma girişimi başarısız olduğunda yeniden deneme çıkışı çağrılır. Çıkışı, hangi koşullar altında yeniden deneneceğini, kaç kez yeniden deneneceğini ve ne sıklıkta deneneceğini belirlemek için kullanabilirsiniz.

Bu çıkış, MCA başlatma ve sonlandırma sırasında kanalın alıcı ucunda da çağrılır.

Kanal iletisi-yeniden deneme çıkışına, iletim kuyruğu üstbilgisi, MQXQH ve kuyruktan alınan uygulama iletisi metnini içeren bir aracı arabelleği geçirilir. MQXQH biçimi MQXQH için genel bakışiçinde verilmiştir.

Çıkış tüm neden kodları için çağrılır; çıkış, MCA ' nın kaç kez ve hangi aralıklarla yeniden denemesini istediğini belirler. (Kanal tanımlandığında ayarlanan ileti yeniden deneme sayısı değeri MQCD ' deki çıkışa aktarılır, ancak çıkış bu değeri yoksayabilir.)

MQCXP 'deki MsgRetrySayı alanı, çıkışın her çağrılışında MCA tarafından artırılır ve çıkış, MQCXP' nin MsgRetryAralık alanında ya da MQXCC_SUPPRESS_FUNCTION içinde bulunan bekleme süresi ile MQXCC_OK değerini döndürür. Çıkış, MQCXP ' nin ExitResponse alanında MQXCC_SUPPRESS_FUNCTION öğesini döndürünceye kadar yeniden denemeler süresiz olarak devam eder. Bu tamamlama kodları için MCA tarafından yapılan işlemle ilgili bilgi için bkz. MQCXP .

Tüm yeniden denemeler başarısız olursa, ileti, gönderilmeyen iletiler kuyruğuna yazılır. Kullanılabilir bir gitmeyen iletiler kuyruğu yoksa, kanal durur.

Bir kanal için ileti yeniden deneme çıkışı tanımlamazsanız ve MQRC_Q_FULL gibi geçici bir hata oluşursa, MCA kanal tanımlandığında ayarlanan ileti yeniden deneme sayısını ve ileti yeniden deneme aralıklarını kullanır. Hata daha kalıcı nitelikdeyse ve bunu işlemek için bir çıkış programı tanımlamadıysanız, ileti gitmeyen iletiler kuyruğuna yazılır.

Kanal otomatik tanımlama çıkış programı
Bir alıcı ya da sunucu bağlantısı kanalını başlatma isteği alındığında kanal otomatik tanımlama çıkışı kullanılabilir, ancak o kanal için tanımlama yoktur (IBM MQ for z/OS için değil). Kanalın bir eşgörünümü için tanımlama değişikliğine izin vermek üzere küme gönderen ve küme alıcı kanalları için tüm platformlarda da çağrılabilir.

Kanal otomatik tanımlama çıkışı, bir alıcı ya da sunucu bağlantısı kanalı başlatma isteği alındığında ancak kanal tanımı olmadığında z/OS dışındaki tüm platformlarda çağrılabilir. Otomatik olarak tanımlanan bir günlük nesnesi ya da sunucu bağlantısı kanalı SYSTEM.AUTO.RECEIVERya da SYSTEM.AUTO.SVRCON. Kanal tanımlarının otomatik olarak nasıl oluşturulabileceğine ilişkin açıklamalar için Kanalların hazırlanması başlıklı konuya bakın.

Küme gönderen kanalını başlatma isteği alındığında kanal otomatik tanımlama çıkışı da çağrılabilir. Kanalın bu eşgörünümü için tanımlama değişikliğine izin vermek üzere küme gönderen ve küme alıcı kanalları için çağrılabilir. Bu durumda, çıkış IBM MQ for z/OSiçin de geçerlidir. Kanal otomatik tanımlama çıkışının yaygın bir kullanımı, çıkış adlarının farklı altyapılarda farklı biçimleri olması nedeniyle ileti çıkışlarının (MSGEXIT, RCVEXIT, SCYEXIT ve SENDEXIT) adlarını değiştirmektir. Kanal otomatik tanımlama çıkışı belirtilmezse, z/OS ' un varsayılan davranışı, *[path]/libraryname(function)* biçiminde dağıtılmış bir çıkış adını incelemek ve varsa, sekiz karakter kadar işlev ya da kitaplık adını almaktır. z/OSişletim sisteminde, bir kanal otomatik tanımlama çıkış programı MsgExitPtr, MsgUserDataPtr, SendExitPtr, SendUserDataPtr, ReceiveExitPtr ve ReceiveUserDataPtrtarafından adreslenen alanları, MsgExit, MsgUserData, SendExit, SendUserData, ReceiveExit ve ReceiveUserVeri alanlarının kendileri.

Daha fazla bilgi için bkz. Otomatik tanımlı kanallarla çalışma.

Diğer kanal çıkışları gibi, parametre listesi:

```
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)
```

`ChannelExitParms` , MQCXPiçinde açıklanmıştır. `ChannelDefinition` , MQCD' de açıklanmıştır.

MQCD, çıkışla değiştirilmezse, varsayılan kanal tanımlamasında kullanılan değerleri içerir. Çıkış, alanların yalnızca bir alt kümesini değiştirebilir; bkz. MQ_CHANNEL_AUTO_DEF_EXIT. Ancak, diğer alanları değiştirme girişimi hataya neden olmaz.

Kanal otomatik tanımlama çıkışı, MQXCC_OK ya da MQXCC_SUPPRESS_FUNCTION yanıtını döndürür. Bu yanıtlardan hiçbiri döndürülmezse, MCA, MQXCC_SUPPRESS_FUNCTION döndürülmüş gibi işlemeye devam eder. Yani, otomatik tanımlama bırakılır, yeni bir kanal tanımı yaratılmaz ve kanal başlatılamaz.

ALW *AIX, Linux, and Windows sistemlerinde kanal çıkış programlarının derlenmesi*

AIX, Linux, and Windows sistemlerine ilişkin kanal çıkış programlarını derlemenize yardımcı olması için aşağıdaki örnekleri kullanın.

Windows

Windows

Windowsüzerinde kanal çıkış programları için derleyici ve bağlantı programı komutu:

```
cl.exe /Ic:\mqm\tools\c\include /nologo /c myexit.c
link.exe /nologo /dll myexit.obj /def:myexit.def /out:myexit.dll
```

AIX and Linux sistemleri

Linux AIX

Bu örneklerde, `exit` kitaplık adı ve `ChannelExit` işlev adıdır. AIX üzerinde dışa aktarma dosyası `exit.exp`olarak adlandırılır. Bu adlar, MQCD-kanal tanımlamasıiçinde açıklanan biçimi kullanarak çıkış programına gönderme yapmak için kanal tanımlaması tarafından kullanılır. DEFINE CHANNEL komutunun MSGEXIT parametresine de bakın.

AIXAIXüzerindeki kanal çıkışlarına ilişkin örnek derleyici ve bağlantı oluşturulabilir komutlar:

```
$ xlc_r -q64 -e MQStart -bE:exit.exp -bM:SRE -o /var/mqm/exits64/exit
exit.c -I/usr/mqm/inc
```

LinuxKuyruk yöneticisinin 32 bit olduğu Linux üzerindeki kanal çıkışlarına ilişkin örnek derleyici ve bağlantı komutları:

```
$ gcc -shared -fPIC -o /var/mqm/exits/exit exit.c -I/opt/mqm/inc
```

LinuxKuyruk yöneticisinin 64 bit olduğu Linux üzerindeki kanal çıkışlarına ilişkin örnek derleyici ve bağlantı komutları:

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc
```

İstemcide, 32 bit ya da 64 bit çıkış kullanılabilir. Bu çıkış mqic_r ile bağlantılı olmalıdır.

AIXAIXüzerinde, IBM MQ tarafından çağrılan tüm işlevler dışa aktarılmalıdır. Bu dosya için örnek bir dışa aktarma dosyası:

```
#
!channelExit
MQStart
```

Kanal çıkışlarının yapılandırılması

Kanal çıkışını çağırmak için kanal tanımında adlandırmanız gerekir.

Kanal çıkışları kanal tanımında adlandırılmalıdır. Bu adlandırma işlemini kanalları ilk tanımladığınızda yapabilir ya da bilgileri daha sonra örneğin, ALTER CHANNEL MQSC komutunu kullanarak ekleyebilirsiniz. MQCD kanal veri yapısında kanal çıkış adlarını da verebilirsiniz. Çıkış adının biçimi IBM MQ altyapınıza bağlıdır; bilgi için MQCD ya da MQSC komutları konusuna bakın.

Kanal tanımı bir kullanıcı çıkış programı adı içermiyorsa, kullanıcı çıkışı çağrılmaz.

Kanal otomatik tanımlama çıkışı, tek tek kanal değil, kuyruk yöneticisinin özelliğidir. Bu çıkışın çağrılabilmesi için, çıkışın kuyruk yöneticisi tanımlamasında adlandırılması gerekir. Bir kuyruk yöneticisi tanımlamasını değiştirmek için, ALTER QMGR MQSC komutunu kullanın.

Veri dönüştürme çıkışları yazılıyor

Bu konu derlemi, veri dönüştürme çıkışlarının nasıl yazılacağı hakkında bilgi içerir.

Not: MQSeries VSE/ESAiçin desteklenmez.

Bir MQPUT işlemi yaptığınızda, uygulamanız iletinin ileti tanımlayıcısını (MQMD) yaratır. IBM MQ 'in MQMD' nin içeriğini, yaratıldığı platformdan bağımsız olarak anlayabilmesi gerektiğinden, sistem tarafından otomatik olarak dönüştürülür.

Ancak, uygulama verileri otomatik olarak dönüştürülmez. `CodedCharSetId` ve `Encoding` alanlarının farklı olduğu platformlar arasında (örneğin, ASCII ve EBCDIC arasında) karakter verileri değiş tokuş ediliyorsa, uygulamanın iletinin dönüştürülmesini düzenlemesi gerekir. Uygulama verilerini dönüştürme, kuyruk yöneticisinin kendisi ya da *veri dönüştürme çıkışı*olarak adlandırılan bir kullanıcı çıkış programı tarafından gerçekleştirilebilir. Uygulama verileri yerleşik biçimlerden (MQFMT_STRING gibi) birindeyse, kuyruk yöneticisi yerleşik dönüştürme yordamlarından birini kullanarak veri dönüştürme işlemini kendisi gerçekleştirebilir. Bu konu, uygulama verileri yerleşik biçimde olmadığında IBM MQ ' in sağladığı veri dönüştürme çıkış olanağıyla ilgili bilgileri içerir.

Denetim, MQGET çağrısı sırasında veri dönüştürme çıkışına iletilebilir. Bu, son hedefe ulaşmadan önce farklı platformlar arasında dönüştürmeyi önler. Ancak, son hedef MQGET üzerinde veri dönüştürmeyi desteklemeyen bir altyapıysa, verileri son hedefine gönderen gönderen kanalda CONVERT (YES) belirtmeniz gerekir. Bu, iletim sırasında IBM MQ ' in verileri dönüştürmesini sağlar. Bu durumda, veri dönüştürme çıkışınızın gönderen kanalının tanımlı olduğu sistemde bulunması gerekir.

MQGET çağrısı doğrudan uygulama tarafından yayınlanır. MQMD ' deki `CodedCharSetId` ve `Encoding` alanlarını gereken karakter kümesi ve kodlamaya ayarlayın. Uygulamanız kuyruk yöneticisiyle aynı karakter kümesini ve kodlamayı kullanıyorsa, `CodedCharSetId` değerini MQCCSI_Q_MGR ve `Encoding` değerini MQENC_NATIVE olarak ayarlayın. MQGET çağrısı tamamlandıktan sonra, bu alanlar döndürülen ileti verilerine uygun değerleri içerir. Dönüştürme başarılı olamazsa, bunlar gereken değerlerden farklı olabilir. Uygulamanız, her MQGET çağrıdan önce bu alanları gereken değerlere geri döndürmelidir.

Veri dönüştürme çıkışının çağrılması için gereken koşullar, MQGETiçinde MQGET çağrısı için tanımlanır.

Veri dönüştürme çıkışına geçirilen parametrelerin açıklaması ve ayrıntılı kullanım notları için MQ_DATA_CONV_EXIT çağrısı ve MQDXP yapısı için Veri dönüştürme başlıklı konuya bakın.

Uygulama verilerini farklı makine kodlamaları ve CCSID ' ler arasında dönüştüren programlar, IBM MQ veri dönüştürme arabirimine (DCI) uymalıdır.

Çoklu yayın istemcileri için, bazı iletiler kuyruk yöneticisinden geçmeyebileceğinden, API çıkışlarının ve veri dönüştürme çıkışlarının istemci tarafında çalışabilmeleri gerekir. Aşağıdaki kitaplıklar, sunucu paketlerinin yanı sıra istemci paketlerinin de bir parçasıdır:

Çizelge 143. İstemci ve sunucu paketlerindeki kitaplıklar

İşletim sistemi	**Kütüphaneler**
AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
IBM i	LIBMQM ve LIBMQM_R
Linux	32 bit & 64 bit: libmqm.so & libmqm_r.so
Windows	32 bit & 64 bit: mqm.dll & mqm.pdb

Veri dönüştürme çıkışının çağrılması

Veri dönüştürme çıkışı, bir MQGET çağrısının işlenmesi sırasında denetimi alan, kullanıcı tarafından yazılan bir çıkıştır.

Aşağıdaki deyimler doğruysa çıkış çağrılır:

- MQGET çağrısında MQGMO_CONVERT seçeneği belirtildi.
- İleti verilerinin bazıları ya da tümü istenen karakter kümesinde ya da kodlamada değil.
- İletiyle ilişkilendirilmiş MQMD yapısındaki *`Format`* alanı MQFMT_NONE değil.

- MQGET çağrısında belirtilen *BufferLength* sıfır değil.
- İleti verileri uzunluğu sıfır değil.
- İleti, kullanıcı tanımlı biçime sahip veriler içeriyor. Kullanıcı tanımlı biçim, iletinin tamamını kaplayabilir ya da bir ya da daha çok yerleşik biçimden önce gelir. Örneğin, kullanıcı tanımlı biçimin önünde bir MQFMT_DEAD_LETTER_HEADER biçimi olabilir. Çıkış yalnızca kullanıcı tanımlı biçimi dönüştürmek için çağrılır; kuyruk yöneticisi, kullanıcı tanımlı biçimden önce gelen yerleşik biçimleri dönüştürür.

  Yerleşik biçimi dönüştürmek için kullanıcı tarafından yazılan bir çıkış da çağrılabilir; ancak, yerleşik dönüştürme yordamları yerleşik biçimi başarıyla dönüştüremezse bu durum oluşur.

MQ_DATA_CONV_EXITiçindeki MQ_DATA_CONV_EXIT çağrısının kullanım notlarında tam olarak açıklanan başka koşullar da vardır.

MQGET çağrısının ayrıntıları için MQGET kısmına bakın. Veri dönüştürme çıkışları MQXCNVC dışında MQI çağrılarını kullanamaz.

Uygulama, kuyruk yöneticisine bağlandığından bu yana *Format* ' u kullanan ilk iletiyi almaya çalıştığında çıkışın yeni bir kopyası yüklenir. Kuyruk yöneticisi önceden yüklenmiş bir kopyayı attıysa, başka zamanlarda da yeni bir kopya yüklenebilir.

Veri dönüştürme çıkışı, MQGET çağrısının yayınlandığı programın ortamına benzer bir ortamda çalışır. Bu program, kullanıcı uygulamalarının yanı sıra, ileti dönüştürmeyi desteklemeyen bir hedef kuyruk yöneticisine ileti gönderen bir MCA (ileti kanalı aracısı) olabilir. Ortam, uygun olduğu durumlarda adres alanını ve kullanıcı profilini içerir. Çıkış, kuyruk yöneticisinin ortamında çalışmadığından, kuyruk yöneticisinin bütünlüğünü bozamıyor.

z/OS üzerinde veri dönüştürme

z/OS

z/OSüzerinde, aşağıdakilere dikkat edin:

- Çıkış programları yalnızca birleştirme dilinde yazılabilir.
- Çıkış programları yeniden girmeli ve depolama alanında herhangi bir yerde çalışabilmelidir.
- Çıkış programları, giriş sırasında ortamı geri yüklemeli ve elde edilen saklama alanını serbest bırakmalıdır.
- Çıkış programları BEKLEMEZ ya da ESTAE 'ler ya da SPE' ler yayınlamamalıdır.
- Çıkış programları genellikle aşağıdaki durumlarda z/OS LINK ile çağrılır:
  - Yetkisiz sorun programı durumu
  - Birincil adres alanı denetim kipi
  - Bellek arası olmayan kip
  - Erişim kaydı olmayan kip
  - 31 bit adresleme kipi
  - TCB-PRB kipi
- Bir CICS uygulaması tarafından kullanıldığında, çıkış EXEC CICS LINK tarafından çağrılır ve CICS programlama kurallarına uygun olmalıdır. Parametreler, CICS iletişim alanındaki (COMMAREA) işaretçiler (adresler) tarafından iletilir.

  Önerilmese de, kullanıcı çıkış programları aşağıdaki uyarılarla CICS API çağrılarını da kullanabilir:

  - Sonuçlar MCA tarafından bildirilen iş birimlerini etkileyebileceği için eşitleme noktaları yayınlamayın.
  - CICS Transaction Server tarafından denetlenenler de içinde olmak üzere, IBM MQ for z/OSdışında bir kaynak yöneticisi tarafından denetlenen kaynakları güncellemeyin.

  CONVERT = YES olan kanallar için çıkış, CSQXLIB DD deyimi tarafından başvurulan veri kümesinden yüklenir. MQ IBM MQ CICS Bridge için CSQCBDCI ve CSQCBDCO çıkışları SCSQAUTH ' de yer alır.

IBM i için veri dönüştürme çıkış programı yazılması

IBM iiçin MQ veri dönüştürme çıkış programları yazarken dikkate alınacak adımlarla ilgili bilgiler.

Aşağıdaki adımları izleyin:

1. İleti biçiminizi adlandırın. Ad, MQMD ' nin *Format* alanına sığmalıdır. *Format* adının başında boşluk olmamalıdır; sondaki boşluklar dikkate alınmaz. *Format* yalnızca sekiz karakter uzunluğunda olduğundan, nesnenin adı sekizden fazla boşluk olmayan karakter içermemelidir. Her ileti gönderişinde bu adı kullanmayı unutmayın (örneğimiz Biçim adını kullanır).
2. İletinizi temsil edecek bir yapı oluşturun. Örnek için Geçerli sözdizimi konusuna bakın.
3. Veri dönüştürme çıkışınız için bir kod parçası yaratmak üzere CVTMQMDTA komutuyla bu yapıyı çalıştırın.

   CVTMQMDTA komutu tarafından oluşturulan işlevler, QMQM/H (AMQSVMHA) dosyasında verilen makroları kullanır. Bu makrolar, tüm yapıların paketlenmiş olduğu varsayılarak yazılır; durum böyle değilse, bu makrolar değiştirilir.
4. Sağlanan iskelet kaynak dosyası QMQMSAMP/QCSRC (AMQSVFC4) kopyasını alın ve yeniden adlandırın. (Örneğimiz EXIT_MOD adını kullanır.)
5. Kaynak dosyada aşağıdaki yorum kutularını bulun ve açıklandığı gibi kod ekleyin:
   a. Kaynak dosyanın sonuna doğru, bir yorum kutusu şununla başlar:

      ```
      /* Insert the functions produced by the data-conversion exit */
      ```

      Buraya, “3” sayfa 944. adımda oluşturulan kod parçasını ekleyin.
   b. Kaynak dosyanın ortasına yakın bir yorum kutusu şununla başlar:

      ```
      /* Insert calls to the code fragments to convert the format's */
      ```

      Bunu, `Converttag`STRUCTişlevine yapılan bir açıklama çağrısı izler.

      İşlevin adını, “5.a” sayfa 944. adımda eklediğiniz işlevin adıyla değiştirin. İşlevi etkinleştirmek için açıklama karakterlerini kaldırın. Birden çok işlev varsa, her biri için çağrı oluşturun.
   c. Kaynak dosyanın başlangıcına yakın bir açıklama kutusu şununla başlar:

      ```
      /* Insert the function prototypes for the functions produced by */
      ```

      Buraya, “5.a” sayfa 944. adımda eklenen işlevlere ilişkin işlev prototip deyimlerini ekleyin.

   İleti karakter verileri içeriyorsa, oluşturulan kod MQXCNVC ' yi çağırır; bu, QMQM/LIBMQM hizmet programı için bağ tanımlanarak çözülebilir.
6. EXIT_MOD kaynak modülünü aşağıdaki gibi derleyin:

   ```
   CRTCMOD MODULE(library/EXIT_MOD) +
   SRCFILE(QCSRC) +
   TERASPACE(*YES *TSIFC)
   ```

7. Programı oluşturun/bağlayın.

   İş parçacıklı olmayan uygulamalar için aşağıdakileri kullanın:

   ```
   CRTPGM PGM(library/Format)  +
      MODULE(library/EXIT_MOD)  +
      BNDSRVPGM(QMQM/LIBMQM)  +
      ACTGRP(QMQM)  +
      USRPRF(*USER)
   ```

Temel ortam için veri dönüştürme çıkışının oluşturulmasına ek olarak, iş parçacıklı ortamda başka bir çıkış da gereklidir. Bu yüklenebilir nesneyi _R izlemelidir. MQXCNVC çağrılarını çözmek için LIBMQM_R kitaplığını kullanın. İş parçacıklı bir ortam için her iki yüklenebilir nesne de gereklidir.

```
CRTPGM PGM(library/Format_R)  +
   MODULE(library/EXIT_MOD)  +
   BNDSRVPGM(QMQM/LIBMQM_R)  +
   ACTGRP(QMQM)  +
   USRPRF(*USER)
```

8. Çıkışı IBM MQ işine ilişkin kitaplık listesine yerleştirin. Üretim için, veri dönüştürme çıkış programlarının QSYS ' de saklanması önerilir.

Not:

1. CVTMQMDTA paketlenmiş yapıları kullanıyorsa, tüm IBM MQ uygulamalarının _Packed niteleyicisini kullanması gerekir.
2. Veri dönüştürme çıkış programları yeniden girişli olmalıdır.
3. MQXCNVC, bir veri dönüştürme çıkışından yayınlanabilen tek MQI çağrısıdır.
4. Çıkış programını, kullanıcı tanıtımı derleyicisi seçeneği *USER olarak ayarlanarak derleyin; böylece, çıkış kullanıcının yetkisiyle çalışır.
5. IBM MQ for IBM i ile tüm kullanıcı çıkışları için teraspace bellek etkinleştirmesi gerekir; CRTCMOD ve CRTBNDC komutlarında TERASPACE (*YES *TSIFC) değerini belirleyin.

IBM MQ for z/OS için veri dönüştürme çıkış programı yazılması

IBM MQ for z/OSiçin veri dönüştürme çıkış programları yazarken dikkate alınacak adımlarla ilgili bilgiler.

Aşağıdaki adımları izleyin:

1. Sağlanan kaynak iskeleti CSQ4BAX9 (CICS dışı ortamlar için) ya da CSQ4CAX9 (CICS için) Başlangıç noktan olarak.
2. CSQUCVX yardımcı programını çalıştırın.
3. CSQ4BAX9 ya da CSQ4CAX9 önsözündeki yönergeleri izleyerek, CSQUCVX yardımcı programı tarafından üretilen yordamları, yapıların dönüştürmek istediğiniz iletide gerçekleşmesi sırasıyla birleştirin.
4. Yardımcı program, veri yapılarının paketlenmediğini, verilerin örtük hizalamasının yerine getirildiğini ve yapıların tam sözcük sınırında başladığını ve byte 'ların gerektiği gibi atlandığını varsayar (Geçerli sözdizimi örneğinde tanıtıcı ve VERSION arasında olduğu gibi). Yapılar paketlenmişse, oluşturulan CMQXCALA makrolarını atlayın. Bu nedenle, yapılarınızı tüm alanların adlandırılması ve hiçbir byte atlanmayacak şekilde bildirmeyi düşünün; Geçerli sözdizimiörneğinde, tanıtıcı ve sürüm arasına bir "MQBYTE DUMMY;" alanı ekleyin.
5. Giriş arabelleği dönüştürülecek ileti biçiminden kısaysa, sağlanan çıkış bir hata döndürür. Çıkış, mümkün olduğu kadar çok tam alanı dönüştürse de, hata, uygulamaya dönüştürülmemiş bir iletinin döndürülmesine neden olur. Kısmi alanlar da içinde olmak üzere, kısa giriş arabelleklerinin dönüştürülebilmesine izin vermek istiyorsanız, CSQXCDFA makrosunda TRUNC= değerini YES olarak değiştirin: Hata döndürülmez, böylece uygulama dönüştürülen bir ileti alır. Uygulamanın kesmeyi işlemesi gerekir.
6. Gereksinim duyduğunuz diğer özel işleme kodunu ekleyin.
7. Programı veri biçimi adınızla yeniden adlandırın.
8. Programınızı bir toplu iş uygulama programı gibi derleyin ve düzenleyin (CICS uygulamalarıyla birlikte kullanılmıyorsa). Yardımcı program tarafından oluşturulan koddaki makrolar **thlqual**.SCSQMACS kitaplığında yer alır.

   İleti karakter verileri içeriyorsa, oluşturulan kod MQXCNVC ' yi çağırır. Çıkışınız bu çağrıyı kullanıyorsa, CSQASTUB çıkış sınırlı kod öbeği programıyla bu çağrıyı düzenleyin. Sınırlı kod öbeği, dilden bağımsız ve ortamdan bağımsızdır. Diğer bir yöntem olarak, CSQXCNVC dinamik çağrı adını kullanarak sınırlı kod

öbeğini dinamik olarak yükleyebilirsiniz. Ek bilgi için bkz. “IBM MQ sınırlı kod öbeğini dinamik olarak çağırma” sayfa 986 .

Bağlantı düzenlenen modülü uygulama yükleme kitaplığınıza ve kanal başlatıcınızın başlattığı görev yordamınızın CSQXLIB DD deyimi tarafından başvurulan bir veri kümesine yerleştirin.

9. Çıkış CICS uygulamaları tarafından kullanılacaksa, gerekirse CSQASTUB da içinde olmak üzere, bir CICS uygulama programı gibi derleyin ve düzenleyin. Bunu CICS uygulama programı kitaplığınıza yerleştirin. EXECKEY (CICS) belirtilerek, programı tipik olarak CICS olarak tanımlayın tanımlamada.

Not: LE/370 yürütme ortamı kitaplıkları CSQUCVX yardımcı programını çalıştırmak için gerekli olsa da (bkz. adım “2” sayfa 945), bağlantı düzenlemek ya da veri dönüştürme çıkışının kendisini çalıştırmak için bu kitaplıklara gerek yoktur (“8” sayfa 945 ve “9” sayfa 946 adımlarına bakın).

IBM MQ - IMS köprüsünde veri dönüştürme hakkında bilgi için bkz. “IMS köprü uygulamalarının yazılması” sayfa 70 .

Linux AIX *IBM MQ for AIX or Linux sistemleri için veri dönüştürme çıkışı yazılması*

IBM MQ for AIX or Linux sistemleri için veri dönüştürme çıkış programları yazarken dikkate alınacak adımlarla ilgili bilgiler.

Aşağıdaki adımları izleyin:

1. İleti biçiminizi adlandırın. Ad, MQMD ' nin *Format* alanına sığmalı ve büyük harfli olmalıdır; örneğin, MYFORMAT. *Format* adının başında boşluk olmamalıdır. Sondaki boşluklar dikkate alınmaz. *Format* yalnızca sekiz karakter uzunluğunda olduğundan, nesnenin adı sekizden fazla boşluk olmayan karakter içermemelidir. Her ileti gönderişinde bu adı kullanmayı unutmayın.

   Veri dönüştürme çıkışı iş parçacıklı bir ortamda kullanılıyorsa, bunun iş parçacıklı bir sürüm olduğunu belirtmek için yüklenebilir nesneyi _r izlemelidir.
2. İletinizi temsil edecek bir yapı oluşturun. Örnek için Geçerli sözdizimi konusuna bakın.
3. Veri dönüştürme çıkışınız için bir kod parçası yaratmak üzere bu yapıyı `crtmqcvx` komutuyla çalıştırın.

   `crtmqcvx` komutu tarafından oluşturulan işlevler, tüm yapıların paketlendiğini varsayan makroları kullanır; bu durumda değilse bunları düzeltir.
4. Sağlanan çatı kaynak dosyasını, “1” sayfa 946. adımda ayarladığınız ileti biçiminin adıyla yeniden adlandırın. İskelet kaynak dosyası ve kopyası salt okunur.

   İskelet kaynak dosyası amqsvfc0.colarak adlandırılır.
5. IBM MQ for AIXsistemlerinde amqsvfc.exp adlı bir iskelet dışa aktarma dosyası da sağlanır. Bu dosyayı MYFORMAT.EXP.
6. İskelet, *MQ_INSTALLATION_PATH*/inc dizininde amqsvmha.hadlı örnek bir üstbilgi dosyası içerir; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir. İçerme yolunuzun bu dosyayı almak için bu dizini işaret ettiğinden emin olun.

   amqsvmha.h kütüğü, `crtmqcvx` komutu tarafından üretilen kod tarafından kullanılan makroları içerir. Dönüştürülecek yapı karakter verileri içeriyorsa, bu makrolar MQXCNVC ' yi çağırır.
7. Kaynak dosyada aşağıdaki yorum kutularını bulun ve açıklandığı gibi kod ekleyin:
   a. Kaynak dosyanın sonuna doğru, bir yorum kutusu şununla başlar:

      ```
      /* Insert the functions produced by the data-conversion exit */
      ```

      Buraya, “3” sayfa 946. adımda oluşturulan kod parçasını ekleyin.
   b. Kaynak dosyanın ortasına yakın bir yorum kutusu şununla başlar:

      ```
      /* Insert calls to the code fragments to convert the format's */
      ```

      Bunu, `ConverttagSTRUC`Tişlevine yapılan bir açıklama çağrısı izler.

İşlevin adını, “7.a” sayfa 946. adımda eklediğiniz işlevin adıyla değiştirin. İşlevi etkinleştirmek için açıklama karakterlerini kaldırın. Birden çok işlev varsa, her biri için çağrı oluşturun.

c. Kaynak dosyanın başlangıcına yakın bir açıklama kutusu şununla başlar:

```
/* Insert the function prototypes for the functions produced by */
```

Buraya, “3” sayfa 946. adımda eklenen işlevlere ilişkin işlev prototip deyimlerini ekleyin.

8. Giriş noktası olarak MQStart kullanarak, çıkışınızı paylaşılan kitaplık olarak derleyin. Bunu yapmak için bkz. “AIX and Linux sistemlerinde veri dönüştürme çıkışlarının derlenmesi” sayfa 947.
9. Çıkışı çıkış dizinine yerleştirin. Varsayılan çıkış dizini, 32 bit sistemler için `/var/mqm/exits` ve 64 bit sistemler için `/var/mqm/exits64`dizinidir. Bu dizinleri qm.ini ya da mqclient.ini dosyasında değiştirebilirsiniz. Bu yol her kuyruk yöneticisi için ayarlanabilir ve çıkış yalnızca o yolda ya da yollarda aranabilir.

Not:

1. `crtmqcvx` paketlenmiş yapıları kullanıyorsa, tüm IBM MQ uygulamalarının bu şekilde derlenmesi gerekir.
2. Veri dönüştürme çıkış programları yeniden girişli olmalıdır.
3. MQXCNVC, bir veri dönüştürme çıkışından yayınlanabilen tek MQI çağrısıdır.

Linux AIX *AIX and Linux sistemlerinde veri dönüştürme çıkışlarının derlenmesi*

AIX and Linux sistemlerinde bir veri dönüştürme çıkışının nasıl derleneceğine ilişkin örnekler.

Tüm altyapılarda, modülün giriş noktası MQStart olur.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

AIX

AIX

Aşağıdaki komutlardan birini vererek çıkış kaynak kodunu derleyin:

32 bit uygulamalar

İş parçacıklı olmayan

```
cc -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT \
     MYFORMAT.c -I  MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
xlc_r -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits/MYFORMAT_r \
     MYFORMAT.c -I  MQ_INSTALLATION_PATH/inc
```

64 bit uygulamalar

İş parçacıklı olmayan

```
cc -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT \
     MYFORMAT.c -I MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
xlc_r -q64 -e MQStart -bE:MYFORMAT.exp -bM:SRE -o /var/mqm/exits64/MYFORMAT_r \
     MYFORMAT.c -I  MQ_INSTALLATION_PATH/inc
```

Linux

Linux

Aşağıdaki komutlardan birini vererek çıkış kaynak kodunu derleyin:

31 bit uygulamalar

İş parçacıklı olmayan

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c \
     -I  MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

32 bit uygulamalar

İş parçacıklı olmayan

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/MYFORMAT_r MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

64 bit uygulamalar

İş parçacıklı olmayan

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

İş parçacıklı

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/MYFORMAT_r MYFORMAT.c
     -I  MQ_INSTALLATION_PATH/inc
```

Windows *IBM MQ for Windows için veri dönüştürme çıkışı yazılması*

IBM MQ for Windowsiçin veri dönüştürme çıkış programları yazarken dikkate alınacak adımlarla ilgili bilgiler.

Aşağıdaki adımları izleyin:

1. İleti biçiminizi adlandırın. Ad, MQMD ' nin *Format* alanına sığmalıdır. *Format* adının başında boşluk olmamalıdır. Sondaki boşluklar dikkate alınmaz. *Format* yalnızca sekiz karakter uzunluğunda olduğundan, nesnenin adı sekizden fazla boşluk olmayan karakter içermemelidir.

MQ_INSTALLATION_PATH\Tools\C\Samplesadlı örnekler dizininde amqsvfcn.def adlı bir .DEF dosyası da bulunur. *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizindir. Bu dosyanın bir kopyasını alın ve örneğin, MYFORMAT.DEF. Yaratılmakta olan DLL ' nin adının ve MYFORMAT.DEF aynı. MYFORMAT.DEF içindeki FORMAT1 adının üzerine yeni biçim adını yazın.

Her ileti gönderişinde bu adı kullanmayı unutmayın.

2. İletinizi temsil edecek bir yapı oluşturun. Örnek için Geçerli sözdizimi konusuna bakın.
3. Veri dönüştürme çıkışınız için bir kod parçası yaratmak üzere bu yapıyı `crtmqcvx` komutuyla çalıştırın.

   CRTMQCVX komutu tarafından oluşturulan işlevler, tüm yapıların paketlendiğini varsayarak yazılan makroları kullanır; böyle bir durumda değilse bunları iyileştirin.
4. Sağlanan çatı kaynak dosyasını (amqsvfc0.c), “1” sayfa 948. adımda ayarladığınız ileti biçiminin adıyla yeniden adlandırın.

   amqsvfc0.c , *MQ_INSTALLATION_PATH*\Tools\C\Samples dizininde bulunur; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizindir. (Varsayılan kuruluş dizini: `C:\Program Files\IBM\MQ`.)

   İskelet, *MQ_INSTALLATION_PATH*\Tools\C\include dizininde amqsvmha.h adlı örnek bir üstbilgi dosyası içerir. İçerme yolunuzun bu dosyayı almak için bu dizini işaret ettiğinden emin olun.

   amqsvmha.h dosyası, CRTMQCVX komutu tarafından oluşturulan kod tarafından kullanılan makroları içerir. Dönüştürülecek yapı karakter verileri içeriyorsa, bu makrolar MQXCNVC ' yi çağırır.
5. Kaynak dosyada aşağıdaki yorum kutularını bulun ve açıklandığı gibi kod ekleyin:
   a. Kaynak dosyanın sonuna doğru, bir yorum kutusu şununla başlar:

      ```
      /* Insert the functions produced by the data-conversion exit */
      ```

      Buraya, “3” sayfa 949. adımda oluşturulan kod parçasını ekleyin.
   b. Kaynak dosyanın ortasına yakın bir yorum kutusu şununla başlar:

      ```
      /* Insert calls to the code fragments to convert the format's */
      ```

      Bunu, `ConverttagSTRUC`Tişlevine yapılan bir açıklama çağrısı izler.

      İşlevin adını, “5.a” sayfa 949. adımda eklediğiniz işlevin adıyla değiştirin. İşlevi etkinleştirmek için açıklama karakterlerini kaldırın. Birden çok işlev varsa, her biri için çağrı oluşturun.
   c. Kaynak dosyanın başlangıcına yakın bir açıklama kutusu şununla başlar:

      ```
      /* Insert the function prototypes for the functions produced by */
      ```

      Buraya, “3” sayfa 949. adımda eklenen işlevlere ilişkin işlev prototip deyimlerini ekleyin.
6. Aşağıdaki komut dosyasını oluşturun:

   ```
   cl -I  MQ_INSTALLATION_PATH\Tools\C\Include -Tp   \
   MYFORMAT.C


   MYFORMAT.DEF
   ```

   Burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizindir.
7. Çıkışınızı DLL dosyası olarak derlemek için komut dosyasını çalıştırın.
8. Çıkışı, çıkış altdizinine IBM MQ veri dizininin altına yerleştirin. Çıkışlarınızı 32 bit sistemlere kurmak için varsayılan dizin şudur: *MQ_DATA_PATH*\Exits ve 64 bit sistemler için: *MQ_DATA_PATH*\Exits64

   Veri dönüştürme çıkışlarını aramak için kullanılan yol kayıtta verilir. Kayıt klasörü:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere
MQ\Installation\MQ_INSTALLATION_NAME\Configuration\ClientExitPath\
```

ve kayıt anahtarı: `ExitsDefaultPath`. Bu yol her kuyruk yöneticisi için ayarlanabilir ve çıkış yalnızca o yolda ya da yollarda aranabilir.

Not:

1. CRTMQCVX paketlenmiş yapıları kullanıyorsa, tüm IBM MQ uygulamalarının bu şekilde derlenmesi gerekir.
2. Veri dönüştürme çıkış programları yeniden girişli olmalıdır.
3. MQXCNVC, bir veri dönüştürme çıkışından yayınlanabilen tek MQI çağrısıdır.

Windows *Windows işletim sistemlerinde çıkış ve anahtar yükleme dosyaları*

IBM WebSphere MQ for Windows 7.5 kuyruk yöneticisi işlemleri 32 bit 'tir. Sonuç olarak, 64 bit uygulamalar kullanılırken, bazı çıkış ve XA anahtar yükleme dosyalarının kuyruk yöneticisi tarafından kullanılmak üzere 32 bit sürümünün de olması gerekir. Çıkış ya da XA anahtar yükleme dosyasının 32 bit sürümü gerekiyorsa ve kullanılamıyorsa, ilgili API çağrısı ya da komutu başarısız olur.

*ExitPath*için `qm.ini file` içinde iki öznitelik desteklenir. Bunlar `ExitsDefaultPath=` *MQ_INSTALLATION_PATH*`\exits` ve `ExitsDefaultPath64=` *MQ_INSTALLATION_PATH*`\exits64'` dir. *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir. Bu bilgilerin kullanılması, uygun kitaplığın bulunmasını sağlar. IBM MQ kümesinde bir çıkış kullanılırsa, bu, uzak sistemdeki uygun kitaplığın bulunmasını da sağlar.

Aşağıdaki çizelge, 32 bit ya da 64 bit uygulamaların kullanılıp kullanılmadığına bağlı olarak, farklı Çıkış ve Anahtar yükleme dosyası tiplerini ve notlarını listeler:

Dosya tipleri	**32 bitlik uygulamalar**	**64 bit uygulamalar**
API çıkışı	32 bit ve 64 bit	64 bit
Veri dönüştürme çıkışı	32 bit	64 bit
Sunucu Kanalı çıkışları (tüm tipler)	64 bit	64 bit
İstemci Kanalı çıkışları (tüm tipler)	32 bit	64 bit
Kurulabilir hizmet çıkışı	64 bit	64 bit
Küme WLM çıkışı	64 bit	64 bit
Pub/Alt yönlendirme çıkışı	64 bit	64 bit
Veritabanı anahtarı yükleme dosyaları	32 bit ve 64 bit	64 bit
Dış Transaction Manager AX kitaplıkları	32 bit	64 bit
Ön bağlantı çıkışı	32 bit	64 bit

Havuzdan ön bağlantı çıkışı kullanarak bağlantı tanımlamalarına gönderme yapma

IBM MQ MQI clients , bağlantı öncesi çıkış kitaplığını kullanarak bağlantı tanımlamalarını elde etmek için bir havuzu aramak üzere yapılandırılabilir.

Giriş

Bir istemci uygulaması, istemci kanal tanımlama çizelgelerini (CCDT) kullanarak kuyruk yöneticisine bağlanabilir. Genellikle, CCDT dosyası merkezi bir ağ dosya sunucusunda bulunur ve bu dosyaya gönderme yapan istemcileri vardır. CCDT dosyasına başvuran çeşitli istemci uygulamalarını yönetmek ve yönetmek zor olduğundan, esnek bir yaklaşım, istemci tanımlarını LDAP dizini, WebSphere Registry and Repository ya da başka bir havuz gibi genel bir havuzda saklamaktır. İstemci bağlantısı tanımlamalarının bir havuzda saklanması, istemci bağlantısı tanımlamalarının yönetilmesini kolaylaştırır ve uygulamalar doğru ve en güncel istemci bağlantısı tanımlamalarına erişebilir.

MQCONN/X çağrısı yürütülürken, IBM MQ MQI client belirtilen bir ön bağlantı çıkış kitaplığını yükler ve bağlantı tanımlamalarını almak için bir çıkış işlevini çağırır. Alınan bağlantı tanımlamaları daha sonra bir kuyruk yöneticisiyle bağlantı kurmak için kullanılır. Çağrılacak çıkış kitaplığının ve işlevinin ayrıntıları mqclient.ini yapılanış dosyasında belirtilir.

Sözdizimi

`void MQ_PRECONNECT_EXIT` (*pExitParms*, *pQMgrName*, *ppConnectOpts*, *pCompCode*, *pReason*);

Parametreler

pExitParms
Tip: PMQNXP giriş/çıkış

PreConnection çıkış parametresi yapısı.

Yapı, çıkışı çağıran tarafından ayrılır ve korunur.

pQMgrAdı
Tip: PMQCHAR giriş/çıkış

Kuyruk yöneticisinin adı.

Girişte bu değiştirge, **QMgrName** değiştirgesiyle MQCONN API çağrısına sağlanan süzgeç dizgisidir. Bu alan boş, açık ya da belirli genel arama karakterleri içeriyor olabilir. Alan çıkış tarafından değiştirilir. Çıkış MQXR_TERM ile çağrıldığında değiştirge boş değerli (NULL) olur.

ppConnectSeçmeler
Tip: ppConnectOpts giriş/çıkış

MQCONNX işlemini denetleyen seçenekler.

Bu, MQCONN API çağrısının işlemini denetleyen bir MQCNO bağlantı seçenekleri yapısına ilişkin bir işaretçidir. Çıkış MQXR_TERM ile çağrıldığında değiştirge boş değerli (NULL) olur. MQI istemcisi, başlangıçta uygulama tarafından sağlanmamış olsa da, çıkışa her zaman bir MQCNO yapısı sağlar. Bir uygulama MQCNO yapısı sağlarsa, istemci bunu değiştirildiği çıkışa geçirmek için bir yineleme yapar. İstemci MQCNO ' nun sahipliğini korur.

MQCNO ile gönderme yapılan bir MQCD, dizi aracılığıyla sağlanan bağlantı tanımlamasından önceliklidir. İstemci kuyruk yöneticisine bağlanmak için MQCNO yapısını kullanır ve diğerleri yoksayılır.

pCompKodu
Tip: PMQLONG giriş/çıkış

Tamamlanma kodu.

Çıkış tamamlama kodunu alan bir MQLONG göstergesi. Aşağıdaki değerlerden biri olmalıdır:

- `MQCC_OK` -Başarılı tamamlama
- `MQCC_WARNING` -Uyarı (kısmi tamamlama)
- `MQCC_FAILED` -Çağrı başarısız oldu

pReason
Tip: PMQLONG giriş/çıkış

Neden niteleyen pCompKodu.

Çıkış neden kodunu alan bir MQLONG göstergesi. Tamamlanma kodu MQCC_OK ise, tek geçerli değer:

- MQRC_NONE-(0, x '000') Raporlamak için neden yok.

Tamamlanma kodu MQCC_FAILED ya da MQCC_WARNING ise, çıkış işlevi neden kodu alanını geçerli bir MQRC_ * değerine ayarlayabilir.

C Çağırma

```
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*/
```

Yayınlama çıkışları yazılıyor ve derleniyor

Yayınlanan bir iletinin içeriğini aboneler tarafından alınmadan önce değiştirmek için kuyruk yöneticisinde bir yayınlama çıkışı yapılandırabilirsiniz. İleti üstbilgisini değiştirebilir ya da iletiyi bir aboneliğe teslim etmeyebilirsiniz.

Not: z/OSüzerinde yayınlama çıkışları desteklenmez.

Abonelere gönderilen iletileri incelemek ve değiştirmek için yayınlama çıkışını kullanabilirsiniz:

- Her bir aboneye yayınlanan bir iletinin içeriğini inceleyin
- Her aboneye yayınlanan bir iletinin içeriğini değiştirme
- İletinin konduğu kuyruğu değiştir
- Bir iletinin aboneye tesliminin durdurulması

Yayınlama çıkışı yazılıyor

Çıkışınızı yazmanıza ve derlemenize yardımcı olması için “AIX, Linux, and Windows üzerinde çıkışlar ve kurulabilir hizmetler yazılıyor” sayfa 897içindeki adımları kullanın.

Yayınlama çıkışının sağlayıcısı, çıkışın ne yaptığını tanımlar. Ancak, çıkışın MQPSXP' de tanımlanan kurallara uyması gerekir.

IBM MQ , MQ_PUBLISH_EXIT giriş noktasının somutlamasını sağlamaz. Bu bir C dili tip tanımı beyannaması sağlar. Parametreleri kullanıcı tarafından yazılan bir çıkışa doğru olarak bildirmek için typedef işlevini kullanın. Aşağıdaki örnek, typedef bildiriminin nasıl kullanılacağını göstermektedir:

```
#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 */
}
```

Yayınlama çıkışı, aşağıdaki işlemlerin sonucu olarak kuyruk yöneticisi işlemi içinde çalışır:

- Bir iletinin bir ya da daha çok aboneye teslim edildiği yayınlama işlemi
- Bir ya da daha çok alıkonan iletinin teslim edildiği bir Abone Olma işlemi
- Bir ya da daha fazla alıkonan iletinin teslim edildiği bir Abonelik İsteği işlemi

Bir bağlantı için yayınlama çıkışı çağrılırsa, bağlantıya ilk kez *ExitReason* kodu MQXR_INIT adı verilir. Yayınlama çıkışı kullanıldıktan sonra bağlantı kesilmeden önce, çıkış *ExitReason* kodu MQXR_TERMile çağrılır.

Yayınlama çıkışı yapılandırıldıysa, ancak kuyruk yöneticisi başlatıldığında yüklenemezse, kuyruk yöneticisi için yayınlama/abone olma ileti işlemleri engellenir. Yayınlama/abone olma ileti sistemi yeniden etkinleştirilmeden önce sorunu düzeltmeniz ya da kuyruk yöneticisini yeniden başlatmanız gerekir.

Yayınlama çıkışını gerektiren her IBM MQ bağlantısı, çıkışı yükleyemeyebilir ya da kullanıma hazırlayamayabilir. Çıkış yüklenemezse ya da kullanıma hazırlanamazsa, o bağlantı için yayınlama çıkışı gerektiren yayınlama/abone olma işlemleri geçersiz kılınır. İşlemler IBM MQ neden kodu MQRC_PUBLISH_EXIT_ERROrile başarısız olur.

Yayınlama çıkışının çağrıldığı bağlam, bir uygulama tarafından kuyruk yöneticisine yapılan bağlantıdır. Yayınlama işlemlerini gerçekleştiren her bağlantı için kuyruk yöneticisi tarafından bir kullanıcı veri alanı sağlanır. Çıkış, her bağlantı için kullanıcı verileri alanındaki bilgileri koruyabilir.

Yayınlama çıkışı bazı MQI çağrılarını kullanabilir. Yalnızca ileti özelliklerini işleyen MQI çağrılarını kullanabilir. Aramalar şunlardır:

- MQBUFMH
- MQCRTMH
- MQDLTMH
- MQDLTMP
- MQMHBUF
- MQINQMP
- MQSETMP

Yayınlama çıkışı hedef kuyruk yöneticisini ya da kuyruk adını değiştirirse, yeni yetki denetimi gerçekleştirilmez.

Yayınlama çıkışını derleme

Yayınlama çıkışı dinamik olarak yüklenen bir kitaplıktır; kanal çıkışı olarak düşünülebilir. Çıkışları derlemeyle ilgili bilgi için bkz. “AIX, Linux, and Windows üzerinde çıkışlar ve kurulabilir hizmetler yazılıyor” sayfa 897.

Örnek yayınlama çıkışı

Örnek çıkış programı amqspse0.colarak adlandırılır. Çıkışın kullanıma hazırlama, yayınlama ya da sonlandırma işlemleri için çağrılıp çağrılmadığına bağlı olarak, günlük dosyasına farklı bir ileti yazar. Ayrıca, depolama alanını uygun şekilde ayırmak ve serbest bırakmak için çıkış kullanıcı alanı alanının kullanılmasını da gösterir.

Yayınlama çıkışlarının yapılandırılması

Yayınlama çıkışını yapılandırmak için belirli öznitelikleri tanımlamanız gerekir.

Windows ve Linux sistemlerinde öznitelikleri tanımlamak için IBM MQ gezginini kullanabilirsiniz. Öznitelikler, Yayınla/Abone Ol altında, kuyruk yöneticisi özellikleri sayfasında tanımlanır.

AIX and Linux sistemlerinde qm.ini dosyasında yayınlama çıkışını yapılandırmak için PublishSubscribeadlı bir kıta oluşturun. PublishSubscribe kısmı aşağıdaki özniteliklere sahiptir:

PublishExitYol = [yol] |*modüla_adı*
Yayınlama çıkış kodunu içeren modül adı ve yolu. Bu alanın uzunluk üst sınırı: MQ_EXIT_NAME_LENGTH. Varsayılan değer, yayınlama çıkışı olmamesidir.

PublishExitİşlev = *işlev_adı*
Yayınlama çıkış kodunu içeren modüldeki işlev giriş noktasının adı. Bu alanın uzunluk üst sınırı: MQ_EXIT_NAME_LENGTH.

IBM i IBM iişletim tarihinde, bir program kullanılıyorsa, PublishExitFunctionöğesini atlayın.

PublishExitVeri = ***dizgi***
Kuyruk yöneticisi bir yayınlama çıkışı çağırıyorsa, giriş olarak bir MQPSXP yapısını geçirir. **PublishExitData** özniteliği kullanılarak belirtilen veriler, yapının *ExitData* alanında sağlanır. Dizgi en çok MQ_EXIT_DATA_LENGTH karakter uzunluğunda olabilir. Varsayılan değer 32 boş karakterdir.

Küme iş yükü çıkışlarının yazılması ve derlenmesi

Kümelerin iş yükü yönetimini özelleştirmek için bir küme iş yükü çıkış programı yazın. İletileri yönlendirirken günün farklı zamanlarında ya da ileti içeriğinde bir kanal kullanma maliyetini göz önünde bulundurabilirsiniz. Bunlar, standart iş yükü yönetimi algoritması tarafından dikkate alınmayan faktörlerdir.

Çoğu durumda iş yükü yönetimi algoritması gereksinimleriniz için yeterlidir. Ancak, iş yükü yönetimini uyarlamak için kendi kullanıcı çıkış programınızı sağlayabilmek için IBM MQ , bir kullanıcı çıkışını, küme iş yükü çıkışını içerir.

Ağınızla ya da iş yükü dengelemeyi etkilemek için kullanabileceğiniz iletilerle ilgili belirli bilgilere sahip olabilirsiniz. Hangilerinin yüksek kapasiteli kanallar ya da ucuz ağ rotaları olduğunu bilebilir ya da iletileri içeriklerine bağlı olarak yönlendirmek isteyebilirsiniz. Bir küme iş yükü çıkış programı yazmaya ya da üçüncü bir kişi tarafından sağlanan bir programı kullanmaya karar verebilirsiniz.

Küme iş yükü çıkışı, bir küme kuyruğuna erişilirken çağrılır. MQOPEN, MQPUT1 ve MQPUTtarafından çağrılır.

MQOO_BIND_ON_OPEN belirtilirse, MQOPEN zamanında seçilen hedef kuyruk yöneticisi düzeltilmiştir. Bu durumda çıkış yalnızca bir kez çalıştırılır.

Hedef kuyruk yöneticisi MQOPEN zamanında düzeltilmezse, MQPUT çağrısı sırasında hedef kuyruk yöneticisi seçilir. Hedef kuyruk yöneticisi kullanılamıyorsa ya da ileti iletim kuyruğundayken başarısız olursa, çıkış yeniden çağrılır. Yeni bir hedef kuyruk yöneticisi seçildi. İleti aktarılırken ileti kanalı başarısız olursa ve ileti geriletilmezse, yeni bir hedef kuyruk yöneticisi seçilir.

Multi Çoklu platformlar' da kuyruk yöneticisi, kuyruk yöneticisinin bir sonraki başlatılışında yeni küme iş yükü çıkışını yükler.

Kuyruk yöneticisi tanımlaması bir küme iş yükü çıkış programı adı içermiyorsa, küme iş yükü çıkışı çağrılmaz.

MQWXPçıkış değiştirgesi yapısındaki bir küme iş yükü çıkışına çeşitli veriler aktarılır:

- İleti tanımlaması yapısı, MQMD.
- İleti uzunluğu parametresi.
- İletinin bir kopyası ya da iletinin bir parçası.

z/OS dışı platformlarda, CLWLMode=FASTkullanıyorsanız, her işletim sistemi işlemi çıkışın kendi kopyasını yükler. Kuyruk yöneticisiyle farklı bağlantılar, çıkışın farklı kopyalarının çağrılmasına neden olabilir. Çıkış varsayılan güvenli kipte (CLWLMode=SAFE) çalıştırılırsa, çıkışın tek bir kopyası kendi ayrı işleminde çalışır.

Küme iş yükü çıkışları yazılıyor

z/OS z/OSiçin küme iş yükü çıkışları yazma hakkında bilgi için bkz. “IBM MQ for z/OS için küme iş yükü çıkış programlaması” sayfa 956.

IBM MQ 9.1.0' den, kuyruk yöneticisi adres alanı yerine kanal başlatıcı adres alanında çalıştırılan küme iş yükü çıkışları. Küme iş yükü çıkışınız varsa, CSQXLIB DD deyimini kuyruk yöneticisi tarafından başlatılan görev yordamından kaldırmanız ve küme iş yükü çıkışını içeren veri kümesini, kanal başlatıcının başlattığı görev yordamındaki CSQXLIB birleşimine eklemeniz gerekir.

Multi Multiplatforms için, küme iş yükü çıkışları MQI çağrılarını kullanmamalıdır. Diğer yönlerden, küme iş yükü çıkış programlarının yazılması ve derlenmesiyle ilgili kurallar, kanal çıkış programları için

geçerli olan kurallar gibidir. “AIX, Linux, and Windows üzerinde çıkışlar ve kurulabilir hizmetler yazılıyor” sayfa 897içindeki adımları izleyin ve çıkışınızın yazılmasına ve derlenmesine yardımcı olmak için örnek programı (“Örnek küme iş yükü çıkışı” sayfa 955) kullanın.

Kanal çıkışları hakkında daha fazla bilgi için bkz. “Kanal çıkış programları yazılıyor” sayfa 923.

Küme iş yükü çıkışlarının yapılandırılması

ALTER QMGR komutunda küme iş yükü çıkış özniteliğini belirterek, kuyruk yöneticisi tanımlamasında küme iş yükü çıkışlarını adlandırın. Örneğin:

```
ALTER QMGR CLWLEXIT(myexit)
```

İlgili başvurular

Küme iş yükü çıkış çağrısı ve veri yapıları

Örnek küme iş yükü çıkışı

IBM MQ , örnek bir küme iş yükü çıkış programı içerir. Örneği kopyalayabilir ve kendi programlarınız için temel olarak kullanabilirsiniz.

z/OS **IBM MQ for z/OS**

Örnek küme iş yükü çıkış programı Assembler 'de ve Ciçinde sağlanır. Çevirici sürümü CSQ4BAF1 olarak adlandırılır ve `thlqual.SCSQASMS`kitaplığında bulunabilir. C sürümü CSQ4BCF1 olarak adlandırılır ve `thlqual.SCSQC37S`kitaplığında bulunabilir. `thlqual` , kuruluşunuzda IBM MQ veri kümeleri için hedef kitaplık üst düzey niteleyicidir.

Multi **IBM MQ for Multiplatforms**

Örnek küme iş yükü çıkış programı C dilinde sağlanır ve `amqswlm0.`olarak adlandırılır. Şu yerde bulunabilir:

Çizelge 144. Multiplatforms için örnek küme iş yükü çıkış programı konumu

Hizmet olarak sunulan	**filePath**
AIX AIX	*MQ_INSTALLATION_PATH*`/samp`
Windows Windows	*MQ_INSTALLATION_PATH* `\Tools\c\Samples`
IBM i IBM i	`qmqm` kitaplığı

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Bu örnek çıkış, kuyruk yöneticisi kullanılamaz duruma gelmezse, tüm iletileri belirli bir kuyruk yöneticisine yöneltir. İletileri başka bir kuyruk yöneticisine yönlendirerek kuyruk yöneticisinin başarısızlığına tepki verir.

İletilerin gönderilmesini istediğiniz kuyruk yöneticisini belirtin. Kuyruk yöneticisi tanımlamasındaki `CLWLDATA` özniteliğinde küme-alıcı kanalının adını belirtin. Örneğin:

```
ALTER QMGR CLWLDATA(' my-cluster-name. my-queue-manager ')
```

Çıkışı etkinleştirmek için, `CLWLEXIT` özniteliğinde tam yolunu ve adını belirtin:

Linux AIX AIX and Linux'ta:

```
ALTER QMGR CLWLEXIT(' path /amqswlm(clwlFunction)')
```

Windows Windows'ta:

```
ALTER QMGR CLWLEXIT(' path \amqswlm(clwlFunction)')
```

z/OS z/OS'ta:

```
ALTER QMGR CLWLEXIT(CSQ4BxF1)
```

Burada x , kullandığınız sürümün programlama diline bağlı olarak `'A'` ya da `'C''` dir.

IBM i IBM i' de aşağıdaki komutlardan birini kullanın:

- MQSC komutunu kullanın:

  ```
  ALTER QMGR CLWLEXIT('AMQSWLM       library       ')
  ```

  Program adı ve kitaplık adı 10 karakter uzunluğunda olur ve gerekirse sağda boşluk bırakılır.
- CL komutunu kullanın:

  ```
  CHGMQM MQMNAME( qmgrname ) CLWLEXIT(' library /AMQSWLM')
  ```

Şimdi, sağlanan iş yükü yönetimi algoritmasını kullanmak yerine, IBM MQ tüm iletileri seçtiğiniz kuyruk yöneticisine yönlendirmek için bu çıkışı çağırır.

z/OS *IBM MQ for z/OS için küme iş yükü çıkış programlaması*

Küme iş yükü çıkışları, bir z/OS **LINK** komutu tarafından çağrılır. Çıkışlar, bir dizi sıkı programlama kuralına tabidir. Bekleme işlemlerini içeren SVC komutlarının çoğunu kullanmaktan ya da iş yükü çıkışında STAE ya da ESTAE kullanmaktan kaçının.

Küme iş yükü çıkışları, aşağıdaki içerdeki bir z/OS **LINK** tarafından çağrılır:

- Yetkisiz sorun programı durumu
- Birincil adres alanı denetim kipi
- Bellek dışı kip
- Erişim dışı kayıt kipi
- 31 bit adresleme kipi
- Depolama anahtarı 8
- Program Anahtar Maskesi 8
- TCB anahtarı 8

Bağlantı düzenlenen modülleri, kanal başlatıcının başlatılan görev yordamının `CSQXLIB DD` deyimiyle belirtilen veri kümesine koyun. Yükleme birimlerinin adları, kuyruk yöneticisi tanımlamasında iş yükü çıkış adları olarak belirtilir.

IBM MQ for z/OSiçin iş yükü çıkışları yazılırken aşağıdaki kurallar geçerlidir:

- Çıkışları çevirici ya da C içine yazmalısınız. C kullanıyorsanız, sistem çıkışları için C sistemleri programlama ortamına uygun olmalıdır; açıklamalar için bkz. *z/OS C/C++ Programming Guide*, SC09-4765.
- `MQXCLWLN` çağrısı kullanılıyorsa, *`thlqual`*`.SCSQLOAD`içinde sağlanan `CSQMFCLW`ile bağlantı düzenleyin.

- Çıkışlar, bir CSQXLIB DD deyimi tarafından tanımlanan yetkisiz kitaplıklardan yüklenir. CSQXLIB ' da DISP=SHRvarsa, kuyruk yöneticisi çalışırken çıkışlar güncellenebilir ve kuyruk yöneticisinin başlatabileceği sonraki MQCONN iş parçacığında kullanılan yeni sürümle güncellenebilir.
- Çıkışların yeniden girilebilmesi ve sanal saklama alanında herhangi bir yerde çalışabilmesi gerekir.
- Çıkışlar, girişte o ortama geri döndüğünüzde ortamı sıfırlamalıdır.
- Çıkışlar, elde edilen saklama alanını serbest bırakmalı ya da saklama alanının sonraki bir çıkış çağrısıyla serbest bırakıldığından emin olmalıdır.
- MQI çağrılarına izin verilmez.
- Bekleme, kuyruk yöneticisinin başarımını önemli ölçüde düşürdüğü için, çıkışlar beklemenin neden olabileceği hiçbir sistem hizmetini kullanmamalıdır. Bu nedenle genel olarak bir SVC, PC ya da G/Ç ' den kaçının.
- Çıkışlar, ekledikleri alt görevler dışında bir ESTAE ya da SPIE yayınlamamalıdır.

Not: Bir çıkışta neler yapabileceğinize ilişkin mutlak bir kısıtlama yoktur. Ancak, çoğu SVC bekleme içerir, bu nedenle aşağıdaki komutlar dışında bunlardan kaçının:

- **GETMAIN** / **FREEMAIN**
- **LOAD** / **DELETE**

ETAE 'lerin ve ESPIE'lerin hata işlemesi IBM MQtarafından gerçekleştirilen hata işlemeyi engelleyebileceğinden, ESTAE' leri ve ESPIE' leri kullanmayın. IBM MQ bir hatadan kurtarılamayabilir ya da çıkış programınız tüm hata bilgilerini almayabilir.

EXITLIM sistem parametresi, bir çıkışın çalışabileceği süreyi sınırlar. EXITLIM için varsayılan değer 30 saniyedir. MQRC_CLUSTER_EXIT_ERRORdönüş kodunu görürseniz, 2266 X'8DA' çıkışınız döngüye dönüyor olabilir. Çıkışın tamamlanması için 30 saniyeden fazla süre gerektiğini düşünüyorsanız, EXITLIMdeğerini artırın.

Yordamsal uygulama oluşturma

Birkaç yordam dilinden birinde bir IBM MQ uygulaması yazabilir ve uygulamayı birkaç farklı platformda çalıştırabilirsiniz.

AIX AIX üzerinde yordamsal uygulamanızı oluşturma

AIX yayınları, yazdığınız programlardan yürütülebilir uygulamaların nasıl oluşturulacağını açıklar.

Bu konuda, AIXaltında çalıştırılacak IBM MQ for AIX uygulamaları oluştururken gerçekleştirmeniz gereken ek görevler ve standart görevlerdeki değişiklikler açıklanmaktadır. C, C + + ve COBOL desteklenir. C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. C++ kullanarak.

IBM MQ for AIX kullanarak yürütülebilir bir uygulama oluşturmak için gerçekleştirmeniz gereken görevler, kaynak kodunuzun yazıldığı programlama diline göre değişir. Kaynak kodunuzda MQI çağrılarını kodlamanın yanı sıra, kullanmakta olduğunuz dile ilişkin IBM MQ for AIX dosyalarını içerecek uygun dil deyimlerini de eklemeniz gerekir. Bu dosyaların içeriğini tanıtın. Tam açıklama için bkz. “IBM MQ veri tanımlama dosyaları” sayfa 688 .

İş parçacıklı sunucu ya da iş parçacıklı istemci uygulamalarını çalıştırdığınızda, AIXTHREAD_SCOPE = S. ortam değişkenini ayarlayın.

AIX *AIX ' da C programlarını hazırlama*

Bu konu, AIXüzerinde C programları hazırlamak için gereken kitaplıkları bağlamaya ilişkin bilgileri içerir.

Ön derlenmiş C programları *MQ_INSTALLATION_PATH*/samp/bin dizininde sağlanır. ANSI derleyicisini kullanın ve aşağıdaki komutları çalıştırın. 64 bitlik programlama uygulamaları hakkında daha fazla bilgi için bkz. 64 bitlik platformlarda kodlama standartları.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

32 bit uygulamalar için:

```
$ xlc_r -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm
```

Burada `amqsput0` örnek bir programdır.

64 bit uygulamalar için:

```
$ xlc_r -q64 -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64
-lmqm
```

Burada `amqsput0` örnek bir programdır.

V 9.3.5 XLC 17 derleyicisini kullanan 32 bit uygulamalar için:

```
$ ibm-clang -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-lmqm
```

Burada `amqsput0` örnek bir programdır.

V 9.3.5 XLC 17 derleyicisini kullanan 64 bit uygulamalar için:

```
$ ibm-clang -m64 -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/
lib64 -lmqm
```

Burada `amqsput0` örnek bir programdır.

C++ programları için VisualAge C/C++ derleyicisini kullanıyorsanız, kitaplıkları bağlarken çözümlenen tüm IBM MQ simgelerini almak için `-q namemangling=v5` seçeneğini eklemeniz gerekir.

Yalnızca IBM MQ MQI client for AIX kurulu olan bir makinede programları kullanmak istiyorsanız, bunları istemci kitaplığına bağlamak için programları yeniden derleyin (`-lmqic`).

Kitaplıkları bağlama

Aşağıdaki kitaplıklara gereksinim duyarsınız:

- Programlarınızı IBM MQtarafından sağlanan uygun kitaplığa bağlayın.

  İş parçacıklı olmayan bir ortamda, aşağıdaki kitaplıklardan birine bağlanın:

Kitaplık dosyası	Program/çıkış tipi
libmqm.a	C İçin Sunucu
libmqic.a & libmqm.a	C İçin İstemci

  İş parçacıklı bir ortamda, aşağıdaki kitaplıklardan birine bağlanın:

Kitaplık dosyası	Program/çıkış tipi
libmqm_r.a	C İçin Sunucu
libmqic_r.a & libmqm_r.a	C İçin İstemci

  Örneğin, tek bir derleme biriminden basit bir iş parçacıklı IBM MQ uygulaması oluşturmak için aşağıdaki komutları çalıştırın.

  32 bit uygulamalar için:

```
$ xlc_r -o amqsputc_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm_r
```

  Burada `amqsput0` örnek bir programdır.

  64 bit uygulamalar için:

```
$ xlc_r -q64 -o amqsputc_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib64 -lmqm_r
```

Burada `amqsput0` örnek bir programdır.

V 9.3.5 XLC 17 derleyicisini kullanan 32 bit uygulamalar için:

```
$ ibm-clang_r -o amqsput_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib -lmqm_r
```

Burada `amqsput0` örnek bir programdır.

V 9.3.5 XLC 17 derleyicisini kullanan 64 bit uygulamalar için:

```
$ ibm-clang_r -m64 -o amqsput_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib64 -lmqm_r
```

Burada `amqsput0` örnek bir programdır.

Yalnızca IBM MQ MQI client for AIX kurulu olan bir makinede programları kullanmak istiyorsanız, bunları istemci kitaplığına bağlamak için programları yeniden derleyin (`-lmqic`).

Not:

1. Birden çok kitaplığa bağlantı veremezsiniz. Yani, aynı anda hem iş parçacıklı hem de iş parçacıklı olmayan bir kitaplığa bağlantı veremezsiniz.
2. Kurulabilir bir hizmet yazıyorsanız (ek bilgi için bkz. IBM MQ Yönetimi), iş parçacıklı olmayan bir uygulamadaki `libmqmzf.a` kitaplığına ve iş parçacıklı bir uygulamadaki `libmqmzf_r.a` kitaplığına bağlanmanız gerekir.
3. IBM TXSeries, Encina ya da BEA Tuxedo gibi XA uyumlu bir hareket yöneticisi tarafından dış eşgüdüm için bir uygulama üretiyorsanız, `libmqmxa.a` (ya da hareket yöneticiniz 'long' tipini 64 bit olarak işliyorsa `libmqmxa64.a`) ve `libmqz.a` kitaplıklarını `libmqmxa_r.a` (ya da `libmqmxa64_r.a`) ile ilişkilendirmeniz gerekir. ve `libmqz_r.a` kitaplıklarını iş parçacığı olarak kullanın.
4. Güvenilir uygulamaları iş parçacıklı IBM MQ kitaplıklarına bağlamanız gerekir. Ancak, IBM MQ for AIX or Linux sistemlerinde güvenilen bir uygulamada aynı anda yalnızca bir iş parçacığı bağlanabilir.
5. IBM MQ kitaplıklarını diğer ürün kitaplıklarına bağlamanız gerekir.

AIX *AIX ' da COBOL programlarını hazırlama*

AIX içinde IBM COBOL Set ve Micro Focus COBOL kullanarak COBOL programlarını hazırlarken bu bilgileri kullanın.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

- 32 bit COBOL kopya kitapları aşağıdaki dizine kurulur:

  ```
  MQ_INSTALLATION_PATH/inc/cobcpy32
  ```

  ve simgesel bağlantılar şu yerde oluşturulur:

  ```
  MQ_INSTALLATION_PATH/inc
  ```

- 64 bit COBOL kopya kitapları aşağıdaki dizine kurulur:

  ```
  MQ_INSTALLATION_PATH/inc/cobcpy64
  ```

Aşağıdaki örneklerde **COBCPY** ortam değişkenini şuna ayarlayın:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

32 bit uygulamalar için ve:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

64 bit uygulamalar için.

Programınızı aşağıdaki kitaplık dosyalarından biriyle ilişkilendirmeniz gerekir:

Kitaplık dosyası	Program/çıkış tipi
libmqmcb.a	COBOL için sunucu (iş parçacıklı olmayan uygulama)
libmqmcb_r.a	COBOL için Sunucu (iş parçacıklı uygulama)
libmqicb.a	Client for COBOL (iş parçacığı olmayan uygulama)
libmqicb_r.a	Client for COBOL (iş parçacıklı uygulama)

Programa bağlı olarak IBM COBOL Set derleyicisini ya da Micro Focus COBOL derleyicisini kullanabilirsiniz:

- `amqm` ile başlayan programlar, Micro Focus COBOL derleyicisi için uygundur ve
- `amq0` ile başlayan programlar, her iki derleyici için de uygundur.

IBM COBOL Set for AIX ürününü kullanarak COBOL programlarını hazırlama

Örnek COBOL programları IBM MQile birlikte sağlanır. Böyle bir programı derlemek için, aşağıdaki listede uygun komutu girin:

32 bit iş parçacığı kullanmayan sunucu uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb -qLIB \
-ICOBCPY_VALUE
```

32 bit iş parçacıklı olmayan istemci uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb -qLIB \
-ICOBCPY_VALUE
```

32 bit iş parçacıklı sunucu uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -ICOBCPY_VALUE
```

32 bit iş parçacıklı istemci uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -ICOBCPY_VALUE
```

64 bit iş parçacığı kullanmayan sunucu uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqmcb \
-qLIB -ICOBCPY_VALUE
```

64 bit iş parçacıklı olmayan istemci uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib - lmqicb \
-qLIB -ICOBCPY_VALUE
```

64 bit iş parçacıklı sunucu uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqmcb_r -qLIB -ICOBCPY_VALUE
```

64 bit iş parçacıklı istemci uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \
-lmqicb_r -qLIB -ICOBCPY_VALUE
```

Micro Focus COBOL kullanarak COBOL programlarını hazırlama

Programınızı derlemeden önce ortam değişkenlerini aşağıdaki gibi ayarlayın:

```
export COBCPY=COBCPY_VALUE
export LIBPATH=MQ_INSTALLATION_PATH/lib:$LIBPATH
```

Micro Focus COBOL kullanarak 32 bit COBOL programını derlemek için şunu girin:

- COBOL Sunucusu

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb
```

- COBOL İçin İstemci

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb
```

- COBOL İçin İş parçacıklı Sunucu

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcb_r
```

- COBOL İçin İş parçacığı İstemcisi

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r
```

Micro Focus COBOL kullanarak 64 bit COBOL programını derlemek için şunu girin:

- COBOL Sunucusu

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb
```

- COBOL İçin İstemci

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb
```

- COBOL İçin İş parçacıklı Sunucu

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcb_r
```

- COBOL İçin İş parçacığı İstemcisi

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicb_r
```

Burada `amqminqx` örnek bir programdır

Ayarlamanız gereken ortam değişkenlerinin açıklaması için Micro Focus COBOL belgelerine bakın.

AIX *AIX içinde CICS uygulama programlarını hazırlama*

AIXiçinde CICS programlarını hazırlarken bu bilgileri kullanın.

IBM MQile CICS bağlantısını kurmak için *XA anahtarı* modüllerini kullanın. XA anahtar yapısı hakkında daha fazla bilgi için bkz. XA anahtar yapıları.

Örnek kaynak kod dosyası, diğer işlem iletileri için XA anahtarları geliştirmenizi sağlamak üzere sağlanır. Sağlanan anahtar yükleme modülünün adıÇizelge 145 sayfa 962içinde listelenir.

Çizelge 145. AIXüzerinde CICS uygulama programları için temel kod: XA başlatma yordamı

Açıklama	**C (kaynak)**	**C (exec)- XAD.Stanza**
XA kullanıma hazırlama yordamı	amqzscix.c	*amqzsc* - CICS for AIX

Ürünle birlikte sağlanan IBM MQ anahtar yükleme kütüğünün *amqzsc*önceden oluşturulmuş sürümünü kullanın.

C hareketlerinizi her zaman iş parçacığı korumalı IBM MQ kitaplığı *libmqm_r.aile bağlantılandırın.*, ve COBOL kitaplığı *libmqmcb_r.aile COBOL işlemleriniz.*

CICS hareketlerini desteklemeye ilişkin daha fazla bilgiyi IBM MQ Yönetimi IBM MQ Sistem Yönetimi Kılavuzu 'nda bulabilirsiniz.

AIX *TXSeries CICS desteği*

AIX üzerinde IBM MQ , XA arabirimini kullanarak TXSeries CICS ürününü destekler. CICS uygulamalarının IBM MQ kitaplıklarının iş parçacıklı sürümüyle bağlantılı olduğundan emin olun.

CICS programlarını IBM COBOL Set for AIX ya da Micro Focus COBOL kullanarak çalıştırabilirsiniz. Aşağıdaki bölümlerde, IBM COBOL Set for AIX ve Micro Focus COBOL dillerinde CICS programlarının çalıştırılması arasındaki fark açıklanmaktadır.

C ya da COBOL içinde aynı CICS bölgesine yüklenen IBM MQ programlarını yazın. Aynı CICS bölgesinde C ve COBOL MQI çağrılarının birleşimini yapamazsınız. Kullanılan ikinci dildeki MQI çağrılarının çoğu `MQRC_HOBJ_ERROR`neden koduyla başarısız olur.

IBM COBOL Set for AIX ürününü kullanarak CICS COBOL programlarını hazırlama

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

IBM COBOL kullanmak için aşağıdaki adımları izleyin:

1. Şu ortam değişkenini dışa aktar:

   ```
   export LDFLAGS="-qLIB -bI:/usr/lpp/cics/lib/cicsprIBMCOB.exp \
                   -I MQ_INSTALLATION_PATH/inc -I/usr/lpp/cics/include \
                   -e _iwz_cobol_main  \
   ```

   Burada `LIB` bir derleyici yönergesi.
2. Aşağıdakileri yazarak programı çevirin, derleyin ve birbirine bağlayın:

   ```
   cicstcl -l IBMCOB yourprog.ccp
   ```

Micro Focus COBOL kullanarak CICS COBOL programlarını hazırlama

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Micro Focus COBOL kullanmak için aşağıdaki adımları izleyin:

1. IBM MQ COBOL çalıştırma zamanı kitaplık modülünü aşağıdaki komutu kullanarak çalıştırma zamanı kitaplığına ekleyin:

```
cicsmkcobol -L/usr/lib/dce -L MQ_INSTALLATION_PATH/lib \
            MQ_INSTALLATION_PATH/lib/libmqmcbrt.o -lmqe_r
```

   Not: `cicsmkcobol`ile IBM MQ , COBOL uygulamanızdan C programlama dilinde MQI çağrıları yapmanıza izin vermez.

   Var olan uygulamalarınızda bu tür çağrılar varsa, bu işlevleri COBOL uygulamalarından kendi kitaplığınıza taşımanız önerilir; örneğin, `myMQ.so`. İşlevleri taşıdıktan sonra, CICSiçin COBOL uygulamasını oluştururken IBM MQ kitaplığı `libmqmcbrt.o` eklemeyin.

   Buna ek olarak, COBOL uygulamanız herhangi bir COBOL MQI çağrısı yapmazsa, `libmqmz_r` ile `cicsmkcobol`arasında bağlantı kurmayın.

   Bu, Micro Focus COBOL dil yöntemi dosyasını oluşturur ve CICS çalıştırma zamanı COBOL kitaplığının IBM MQ for AIX or Linux sistemlerini çağırmasını sağlar.

   Not: `cicsmkcobol` komutunu yalnızca aşağıdaki ürünlerden birini kurarken çalıştırın:

   - Micro Focus COBOL ' un yeni sürümü ya da yayını
   - AIX için yeni CICS sürümü ya da yayını
   - Desteklenen herhangi bir veritabanı ürününün yeni sürümü ya da yayını (yalnızca COBOL işlemleri için)
   - IBM MQ ' in yeni sürümü ya da yayını
2. Şu ortam değişkenini dışa aktar:

```
COBCPY= MQ_INSTALLATION_PATH/inc export COBCPY
```

3. Aşağıdakileri yazarak programı çevirin, derleyin ve birbirine bağlayın:

```
cicstcl -l COBOL -e yourprog.ccp
```

CICS C programlarının hazırlanması

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Standart CICS olanaklarını kullanarak CICS C programları oluşturun:

1. Aşağıdaki ortam değişkenlerinden ***birini*** dışa aktarın:
   - LDFLAGS = "-L/ *MQ_INSTALLATION_PATH* lib -lmqm_r" dışa aktarma LDFLAGS
   - USERLIB = "-L *MQ_INSTALLATION_PATH* lib -lmqm_r" export USERLIB
2. Aşağıdakileri yazarak programı çevirin, derleyin ve birbirine bağlayın:

```
cicstcl -l C amqscic0.ccs
```

CICS C örnek hareketi

Bir AIX IBM MQ işlemine ilişkin örnek C kaynağı AMQSCIC0.CCS. Hareket, iletileri SYSTEM.SAMPLE.CICSiletim kuyruğundan okur.WORKQUEUE varsayılan kuyruk yöneticisinde yer alır ve bunları, iletinin iletim üstbilgisinde bulunan bir kuyruk adıyla yerel kuyruğa yerleştirir. Hatalar SYSTEM.SAMPLECICSkuyruğuna gönderilir.-DLQ. AMQSCIC0.TST .

IBM i IBM i üzerinde yordamsal uygulamanızı oluşturma

IBM i yayınlarında, iSeries ya da System i sistemlerinde IBM i ile çalıştırılacak, yazdığınız programlardan yürütülebilir uygulamaların nasıl oluşturulaeceği açıklanır.

Bu konuda, IBM i sistemlerinde çalıştırılacak IBM MQ for IBM i yordamsal uygulamaları oluştururken gerçekleştirmeniz gereken ek görevler ve standart görevlerdeki değişiklikler açıklanmaktadır. COBOL, C, C + +, Java ve RPG programlama dilleri desteklenir. C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. C+ + kullanarak. Java programlarınızı hazırlama hakkında bilgi için bkz. IBM MQ classes for Java ' yi kullanma.

Yürütülebilir bir IBM MQ for IBM i uygulaması oluşturmak için gerçekleştirmeniz gereken görevler, kaynak kodun yazıldığı programlama diline bağlıdır. Kaynak kodunuzda MQI çağrılarını kodlamaya ek olarak, kullanmakta olduğunuz dile ilişkin IBM MQ for IBM i veri tanımlaması dosyalarını içerecek uygun dil deyimlerini de eklemeniz gerekir. Bu dosyaların içeriğini tanıtın. Tam açıklama için bkz. “IBM MQ veri tanımlama dosyaları” sayfa 688 .

IBM i *IBM i ' da C programlarını hazırlama*

IBM MQ for IBM i boyutu 100 MB ' ye kadar olan iletileri destekler. ILE C dilinde yazılan ve 16 MB ' den büyük IBM MQ iletilerini destekleyen uygulama programlarının, bu iletiler için yeterli bellek ayırmak üzere `Teraspace` derleyici seçeneğini kullanması gerekir.

C derleyici seçenekleri hakkında daha fazla bilgi için bkz. *WebSphere Development Studio ILE C/C++ Programmer's Guide*.

Bir C modülünü derlemek için IBM i komutunu **CRTCMOD**kullanabilirsiniz. Derlerken, içerme dosyalarını (QMQM) içeren kitaplığın kitaplık listesinde bulunduğundan emin olun.

Daha sonra, **CRTPGM** komutunu kullanarak derleyicinin çıkışını hizmet programıyla bağlamanız gerekir.

Çizelge 146. İş parçacıklı olmayan ve iş parçacıklı ortamlardaki CRTPGM örneği

Ortam tipi	Komut	Program/çıkış tipi
İş parçacıklı olmayan ortam	`CRTPGM PGM( pgmname ) MODULE( pgmname ) BNDSRVPGM(QMQM/LIBMQM)`	C için sunucu ya da istemci
İş parçacıklı ortam	`CRTPGM PGM( pgmname ) MODULE( pgmname ) BNDSRVPGM(QMQM/LIBMQM_R)`	C için sunucu ya da istemci

Burada *pgmname* , programınızın adıdır.

Çizelge 147 sayfa 964 içinde, iş parçacıklı olmayan bir ortamda ve iş parçacıklı bir ortamda IBM i üzerinde C programları hazırlanırken gerekli olan kitaplıklar listelenir.

Çizelge 147. İş parçacıklı olmayan ve iş parçacıklı ortamlar için gereken kitaplıklar

Ortam tipi	Kitaplık dosyası	Program/çıkış tipi
İş parçacıklı olmayan ortam	LIBMQM	C İçin Sunucu
	LIBMQIC ve LIBMQM	C İçin İstemci
İş parçacıklı ortam	LIBMQM_R	C İçin Sunucu
	LIBMQIC_R & LIBMQM_R	C İçin İstemci

IBM i *IBM i ' da COBOL programlarını hazırlama*

COBOL programlarını IBM i içinde hazırlama ve COBOL programından MQI ' a erişme yöntemi hakkında bilgi edinin.

Bu görev hakkında

MQI 'a COBOL programlarından erişmek için IBM MQ for IBM i , hizmet programları tarafından sağlanan bağlı bir yordamsal çağrı arabirimi sağlar. Bu, IBM MQ for IBM i içindeki tüm MQI işlevlerine erişim ve iş parçacıklı uygulamalar için destek sağlar. Bu arabirim yalnızca ILE COBOL derleyicisiyle kullanılabilir.

MQI işlevlerine erişmek için standart COBOL CALL sözdizimi kullanılır.

MQI ile kullanılacak adlandırılmış değişmezleri ve yapı tanımlamalarını içeren COBOL kopyalama dosyaları, QMQM/QCBLLESRC kaynak fiziksel dosyasında bulunur.

COBOL kopyalama dosyaları, dizilim sınırlayıcı olarak tek tırnak işareti karakterini (') kullanır. IBM i COBOL derleyicileri, sınırlayıcının tırnak işareti (") olduğunu varsayar. Derleyicilerin uyarı iletileri oluşturmasını önlemek için, **CRTCBLPGM**, **CRTBNDCBL**ya da **CRTCBLMOD**komutlarında OPTION (*APOST) komutunu belirleyin.

Derleyicinin COBOL kopya dosyalarında dizilim sınırlayıcı olarak tek tırnak işareti karakterini (') kabul etmesini sağlamak için \APOST derleyici seçeneğini kullanın.

Not: Dinamik çağrı arabirimi, IBM MQ 9.0 ya da sonraki bir sürümde sağlanmaz.

Bağlı yordam çağrısı arabirimini kullanmak için aşağıdaki adımları tamamlayın.

Yordam

1. Aşağıdaki parametreyi belirterek **CRTCBLMOD** derleyicisini kullanarak bir modül yaratın:

   ```
   LINKLIT(*PRC)
   ```

2. Uygun parametreyi belirterek program nesnesini yaratmak için **CRTPGM** komutunu kullanın:

   İş parçacıklı olmayan uygulamalar için:

   ```
   BNDSRVPGM(QMQM/AMQ0STUB)        Server for COBOL for non-threaded applications
   BNDSRVPGM(QMQM/AMQCSTUB)        Client for COBOL for non-threaded applications
   ```

   İş parçacıklı uygulamalar için:

   ```
   BNDSRVPGM(QMQM/AMQ0STUB_R)        Server for COBOL for threaded applications
   BNDSRVPGM(QMQM/AMQCSTUB_R)        Client for COBOL for threaded applications
   ```

   Not: V4R4 ILE COBOL derleyicisi kullanılarak yaratılan ve PROCESS deyiminde THREAD (SERIALIZE) seçeneğini içeren programlar dışında, COBOL programları iş parçacığı IBM MQ kitaplıklarını kullanmamalıdır. Bir COBOL programı bu şekilde iş parçacığı güvenli hale getirilse bile, THREAD (SERIALIZE), COBOL yordamlarının modül düzeyinde diziselleştirilmesini zorladığından ve genel performansı etkileyebileceğinden, uygulamayı tasarlarken dikkatli olun.

   Daha fazla bilgi için *WebSphere Development Studio: ILE COBOL Programmer's Guide* ve *WebSphere Development Studio: ILE COBOL Reference* adlı yayına bakın.

   CICS uygulamasını derleme hakkında daha fazla bilgi için bkz. *CICS for IBM i Application Programming Guide*, SC41-5454.

IBM i *IBM i ' da CICS programlarının hazırlanması*

IBM iiçinde CICS programlarını hazırlarken gerekli adımlar hakkında bilgi edinin.

EXEC CICS deyimlerini ve MQI çağrılarını içeren bir program yaratmak için aşağıdaki adımları izleyin:

1. Gerekirse, CRTCICSMAP komutunu kullanarak eşlemeleri hazırlayın.
2. EXEC CICS komutlarını yerel dil deyimlerine çevirin. C programı için CRTCICSC komutunu kullanın. COBOL programı için CRTCICSCBL komutunu kullanın.

CSTCICSC ya da CRTCICSCBL komutuna `CICSOPT(*NOGEN)` ekleyin. Bu, uygun CICS ve IBM MQ hizmet programlarını eklemeniz için işlemeyi durdurur. Bu komut, kodu varsayılan olarak QTEMP/QACYCICSiçine koyar.

3. Kaynak kodu CRTCMOD komutunu (C programı için) ya da CRTCBLMOD komutunu (COBOL programı için) kullanarak derleyin.
4. Derlenen kodu uygun CICS ve IBM MQ hizmet programlarına bağlamak için CRTPGM ' yi kullanın. Bu, yürütülebilir programı oluşturur.

Bu tür bir kod örneği aşağıdaki gibidir (gönderilen CICS örnek programını derler):

```
CRTCICSC  OBJ(QTEMP/AMQSCIC0) SRCFILE(/MQSAMP/QCSRC) +
     SRCMBR(AMQSCIC0) OUTPUT(*PRINT) +
     CICSOPT(*SOURCE *NOGEN)
CRTCMOD   MODULE(MQTEST/AMQSCIC0) +
     SRCFILE(QTEMP/QACYCICS) OUTPUT(*PRINT)
CRTPGM    PGM(MQTEST/AMQSCIC0) MODULE(MQTEST/AMQSCIC0) +
     BNDSRVPGM(QMQM/LIBMQIC QCICS/AEGEIPGM)
```

IBM i *IBM i içinde RPG programlarını hazırlama*

IBM MQ for IBM ikullanıyorsanız, uygulamalarınızı RPG dilinde yazabilirsiniz.

Daha fazla bilgi için bkz. “RPG ' de IBM MQ programlarının kodlanması (yalnızcaIBM i)” sayfa 1012ve bkz. IBM i Application Programming Reference (ILE/RPG).

IBM i *IBM i için SQL programlamasıyla ilgili önemli noktalar*

SQL kullanarak IBM i üzerinde uygulama oluştururken gerekli adımlar hakkında bilgi edinin.

Programınızda EXEC SQL deyimleri ve MQI çağrıları varsa, aşağıdaki adımları izleyin:

1. EXEC SQL komutlarını yerel dil deyimlerine çevirin. C programı için CRTSQLCI komutunu kullanın. COBOL programı için CRTSQLCBLI komutunu kullanın.

   CRTSQLCI ya da CRTSQLCBLI komutuna `OPTION(*NOGEN)` komutunu ekleyin. Bu işlem, uygun IBM MQ hizmet programlarını eklemeniz için işlemleri durdurur. Bu komut, kodu varsayılan olarak QTEMP/QSQLTEMP ' ye yerleştirir.
2. Kaynak kodu CRTCMOD komutunu (C programı için) ya da CRTCBLMOD komutunu (COBOL programı için) kullanarak derleyin.
3. Derlenen kodu uygun IBM MQ hizmet programlarına bağlamak için CRTPGM ' yi kullanın. Bu, yürütülebilir programı oluşturur.

Bu tür bir kod örneği (SQLUSER kitaplığında SQLTEST adlı bir programı derler):

```
CRTSQLCI  OBJ(MQTEST/SQLTEST) SRCFILE(SQLUSER/QCSRC) +
     SRCMBR(SQLTEST) OUTPUT(*PRINT) OPTION(*NOGEN)
CRTCMOD   MODULE(MQTEST/SQLTEST) +
     SRCFILE(QTEMP/QSQLTEMP) OUTPUT(*PRINT)
CRTPGM    PGM(MQTEST/SQLTEST) +
     BNDSRVPGM(QMQM/LIBMQIC)
```

Linux Linux üzerinde yordamsal uygulamanızı oluşturma

Bu bilgiler, Linux uygulamalarının çalışması için IBM MQ oluştururken gerçekleştirmeniz gereken ek görevleri ve standart görevlerdeki değişiklikleri açıklar.

C ve C++ desteklenir. C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. C++ kullanarak.

Linux *Linux ' da C programlarını hazırlama*

Ön derlenmiş C programları *MQ_INSTALLATION_PATH*/samp/bin dizininde sağlanır. Kaynak koddan örnek oluşturmak için gcc derleyicisini kullanın.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Normal ortamınızda çalışın. 64 bit programlama uygulamaları hakkında daha fazla bilgi için bkz. 64 bit platformlarda kodlama standartları.

Kitaplıkları bağlama

Aşağıdaki çizelgelerde, Linuxüzerinde C programları hazırlanırken gerekli olan kitaplıklar listelenmektedir.

- Programlarınızı IBM MQtarafından sağlanan uygun kitaplığa bağlamanız gerekir.

  İş parçacıklı olmayan bir ortamda, aşağıdaki kitaplıklardan yalnızca birine bağlanın:

Kitaplık dosyası	Program/çıkış tipi
libmqm.so	C İçin Sunucu
libmqic.so & libmqm.so	C İçin İstemci

  İş parçacıklı bir ortamda, aşağıdaki kitaplıklardan yalnızca birine bağlanın:

Kitaplık dosyası	Program/çıkış tipi
libmqm_r.so	C İçin Sunucu
libmqic_r.so & libmqm_r.so	C İçin İstemci

Not:

1. Birden çok kitaplığa bağlantı veremezsiniz. Yani, aynı anda hem iş parçacıklı hem de iş parçacıklı olmayan bir kitaplığa bağlantı veremezsiniz.
2. Kurulabilir bir hizmet yazıyorsanız (ek bilgi için bkz. IBM MQ Yönetimi), `libmqmzf.so` kitaplığına bağlanmanız gerekir.
3. IBM TXSeries Encina ya da BEA Tuxedo gibi XA uyumlu bir hareket yöneticisi tarafından dış eşgüdüm için bir uygulama üretiyorsanız, `libmqmxa.so` (ya da işlem yöneticiniz 'long' tipini 64 bit olarak işliyorsa `libmqmxa64.so`) ve `libmqz.so` kitaplıklarını `libmqmxa_r.so` (ya da `libmqmxa64_r.so`) ile bağlantılandırmanız gerekir. ve `libmqz_r.so` kitaplıklarını iş parçacığı olarak kullanın.
4. IBM MQ kitaplıklarını diğer ürün kitaplıklarına bağlamanız gerekir.

Linux *31 bitlik uygulamalar oluşturma*

Bu konu, çeşitli ortamlarda 31 bit programlar oluşturmak için kullanılan komutlara ilişkin örnekleri içerir.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

C istemci uygulaması, 31 bit, iş parçacığı kullanmayan

```
gcc -m31 -o famqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

C istemci uygulaması, 31 bit, iş parçacığı

```
gcc -m31 -o amqsputc_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

C sunucu uygulaması, 31 bit, iş parçacıklı değil

```
gcc -m31 -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

C sunucu uygulaması, 31 bit, iş parçacığı

```
gcc -m31 -o amqsput_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

C++ istemci uygulaması, 31 bit, iş parçacığı kullanmayan

```
g++ -m31 -fsigned-char -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl
-limqb23gl -lmqic
```

C++ istemci uygulaması, 31 bit, iş parçacığı

```
g++ -m31 -fsigned-char -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r
-limqb23gl_r -lmqic_r -lpthread
```

C++ sunucu uygulaması, 31 bit, iş parçacığı kullanmayan

```
g++ -m31 -fsigned-char -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl
-limqb23gl -lmqm
```

C++ sunucu uygulaması, 31 bit, iş parçacığı

```
g++ -m31 -fsigned-char -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r
-limqb23gl_r -lmqm_r -lpthread
```

C istemci çıkışı, 31 bit, iş parçacıklı değil

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqic
```

C istemci çıkışı, 31 bit, iş parçacığı

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/cliexit_32_r cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

C sunucusu çıkışı, 31 bit, iş parçacıklı değil

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqm
```

C sunucu çıkışı, 31 bit, yivli

```
gcc -m31 -shared -fPIC -o /var/mqm/exits/srvexit_32_r srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Linux *32 bit uygulamalar oluşturma*

Bu konu, çeşitli ortamlarda 32 bit programlar oluşturmak için kullanılan komutlara ilişkin örnekleri içerir.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

C istemci uygulaması, 32 bit, iş parçacıklı değil

```
gcc -m32 -o amqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

C istemci uygulaması, 32 bit, iş parçacığı

```
gcc -m32 -o amqsputc_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

C sunucu uygulaması, 32 bit, iş parçacıklı değil

```
gcc -m32 -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

C sunucu uygulaması, 32 bit, iş parçacığı

```
gcc -m32 -o amqsput_32_r amqsput0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib
-Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

C++ istemci uygulaması, 32 bit, iş parçacıklı değil

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl -limqb23gl -lmqic
```

C++ istemci uygulaması, 32 bit, iş parçacığı

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

C++ sunucu uygulaması, 32 bit, iş parçacıklı değil

```
g++ -m32 -fsigned-char -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl -limqb23gl -lmqm
```

C++ sunucu uygulaması, 32 bit, iş parçacığı

```
g++ -m32 -fsigned-char -o imqsput_32_r imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib
-limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

C istemci çıkışı, 32 bit, iş parçacıklı değil

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqic
```

C istemci çıkışı, 32 bit, iş parçacığı

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/cliexit_32_r cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqic_r -lpthread
```

C sunucu çıkışı, 32 bit, iş parçacıklı değil

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32 srvexit.c -I MQ_INSTALLATION_PATH/inc
```

```
-L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqm
```

C sunucu çıkışı, 32 bit, yivli

```
gcc -m32 -shared -fPIC -o /var/mqm/exits/srvexit_32_r srvexit.c
I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath= MQ_INSTALLATION_PATH/lib
-Wl,-rpath=/usr/lib -lmqm_r -lpthread
```

Linux *64 bit uygulamalar oluşturuluyor*

Bu konu, çeşitli ortamlarda 64 bit programlar oluşturmak için kullanılan komutlara ilişkin örnekler içerir.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

C istemci uygulaması, 64 bit, iş parçacığı kullanmayan

```
gcc -m64 -o amqsputc_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic
```

C istemci uygulaması, 64 bit, iş parçacığı

```
gcc -m64 -o amqsputc_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqic_r
-lpthread
```

C sunucu uygulaması, 64 bit, iş parçacıklı değil

```
gcc -m64 -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm
```

C sunucu uygulaması, 64 bit, iş parçacığı

```
gcc -m64 -o amqsput_64_r amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64 -lmqm_r
-lpthread
```

C++ istemci uygulaması, 64 bit, iş parçacığı kullanmayan

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsput.cpp
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl -limqb23gl -lmqic
```

C++ istemci uygulaması, 64 bit, iş parçacığı

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsput.cpp
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-limqc23gl_r -limqb23gl_r -lmqic_r -lpthread
```

C++ sunucu uygulaması, 64 bit, iş parçacıklı değil

```
g++ -m64 -fsigned-char -o imqsput_64 imqsput.cpp
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl -limqb23gl -lmqm
```

C++ sunucu uygulaması, 64 bit, iş parçacığı

```
g++ -m64 -fsigned-char -o imqsput_64_r imqsput.cpp
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -limqs23gl_r -limqb23gl_r -lmqm_r -lpthread
```

C istemci çıkışı, 64 bit, iş parçacıklı değil

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic
```

C istemci çıkışı, 64 bit, iş parçacığı

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/cliexit_64_r cliexit.c
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqic_r -lpthread
```

C sunucu çıkışı, 64 bit, iş parçacıklı değil

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/srvexit_64 srvexit.c
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm
```

C sunucu çıkışı, 64 bit, iş parçacığı

```
gcc -m64 -shared -fPIC -o /var/mqm/exits64/srvexit_64_r srvexit.c
-I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=MQ_INSTALLATION_PATH/lib64
-Wl,-rpath=/usr/lib64 -lmqm_r -lpthread
```

Linux *Linux ' da COBOL programlarını hazırlama*

Linux ' da COBOL programlarını hazırlama ve x86 ve Micro Focus COBOL üzerinde IBM COBOL for Linux kullanarak COBOL programları hazırlama hakkında bilgi edinin.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

1. 32 bit COBOL kopya kitapları aşağıdaki dizine kurulur:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

   ve simgesel bağlantılar şu yerde oluşturulur:

   ```
   MQ_INSTALLATION_PATH/inc
   ```

2. 64 bitlik platformlarda, 64 bitlik COBOL kopya kitapları aşağıdaki dizine kurulur:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy64
   ```

3. Aşağıdaki örneklerde COBCPY değerini şuna ayarlayın:

   ```
   MQ_INSTALLATION_PATH/inc/cobcpy32
   ```

32 bit uygulamalar için ve:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

64 bit uygulamalar için.

Programınızı aşağıdakilerden biriyle ilişkilendirmeniz gerekir:

Kitaplık dosyası	Program/çıkış tipi
libmqmcb.so	COBOL Sunucusu
libmqicb.so	COBOL İçin İstemci
libmqmcb_r.so	COBOL için Sunucu (iş parçacıklı uygulama)
libmqicb_r.so	Client for COBOL (iş parçacıklı uygulama)

IBM COBOL for Linux on x86 kullanılarak COBOL programlarının hazırlanması

Örnek COBOL programları, IBM MQile birlikte sağlanır. Böyle bir programı derlemek için, aşağıdaki listede uygun komutu girin:

32 bit iş parçacığı kullanmayan sunucu uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-L MQ_INSTALLATION_PATH/lib -lmqmcb -ICOBCPY_VALUE
```

32 bit iş parçacıklı olmayan istemci uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-L MQ_INSTALLATION_PATH/lib -lmqicb -ICOBCPY_VALUE
```

32 bit iş parçacıklı sunucu uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-qTHREAD -L MQ_INSTALLATION_PATH/lib -lmqmcb_r -ICOBCPY_VALUE
```

32 bit iş parçacıklı istemci uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-qTHREAD -L MQ_INSTALLATION_PATH/lib -lmqicb_r -ICOBCPY_VALUE
```

Micro Focus COBOL kullanarak COBOL programlarını hazırlama

Programınızı derlemeden önce ortam değişkenlerini aşağıdaki gibi ayarlayın:

```
export COBCPY=COBCPY_VALUE
export LIB= MQ_INSTALLATION_PATH lib:$LIB
```

Desteklenen yerlerde, Micro Focus COBOL kullanarak 32 bitlik bir COBOL programını derlemek için şunu girin:

```
$ 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
```

Micro Focus COBOL kullanarak 64 bitlik bir COBOL programını derlemek için şunu girin:

```
$ 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
```

Burada `amqsput` örnek bir programdır

Gereksinim duyduğunuz ortam değişkenlerinin açıklaması için Micro Focus COBOL belgelerine bakın.

Windows Windows üzerinde yordamsal uygulamanızı oluşturma

Windows sistemleri yayınları, yazdığınız programlardan yürütülebilir uygulamaların nasıl oluşturulacağını açıklar.

Bu konuda, Windows sistemleri altında çalışacak IBM MQ for Windows uygulamaları oluştururken gerçekleştirmeniz gereken ek görevler ve standart görevlerde yapılması gereken değişiklikler açıklanmaktadır. C, C + +, COBOL ve Visual Basic programlama dilleri desteklenir. C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. C++ kullanarak.

IBM MQ for Windows kullanarak yürütülebilir bir uygulama oluşturmak için gerçekleştirmeniz gereken görevler, kaynak kodunuzun yazıldığı programlama diline göre değişir. Kaynak kodunuzda MQI çağrılarını kodlamanın yanı sıra, kullanmakta olduğunuz dile ilişkin IBM MQ for Windows dosyalarını içerecek uygun dil deyimlerini de eklemeniz gerekir. Bu dosyaların içeriğini tanıtın. Tam açıklama için bkz. “IBM MQ veri tanımlama dosyaları” sayfa 688 .

Windows *Windows üzerinde 64 bit uygulamalar oluşturma*

Hem 32 bit hem de 64 bit uygulamalar IBM MQ for Windowsüzerinde desteklenir. IBM MQ yürütülür dosyaları ve kitaplık dosyaları hem 32 bit hem de 64 bit biçiminde sağlanır, çalıştığınız uygulamaya bağlı olarak uygun sürümü kullanın.

Yürütülebilir dosyalar ve kitaplıklar

IBM MQ kitaplıklarının 32 bit ve 64 bit sürümleri aşağıdaki yerlerde sağlanır:

Çizelge 148. IBM MQ kitaplıklarının yeri

Kitaplık sürümü	**Kitaplık dosyalarını içeren dizin**
32 bit	*MQ_INSTALLATION_PATH* `\Tools\Lib`
64 bit	*MQ_INSTALLATION_PATH* `\Tools\Lib64`

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

32 bit uygulamalar geçişten sonra olağan şekilde çalışmaya devam eder. 32 bit dosyalar, ürünün önceki sürümleriyle aynı dizinde bulunur.

64 bit sürüm oluşturmak istiyorsanız, ortamınızın *MQ_INSTALLATION_PATH* `\Tools\Lib64`içindeki kitaplık dosyalarını kullanacak şekilde yapılandırıldığından emin olmanız gerekir. LIB ortam değişkeninin, 32 bit kitaplıkları içeren klasöre bakacak şekilde ayarlanmadığını doğrulayın.

Windows *Windows ' da C programlarını hazırlama*

Tipik Windows ortamında çalışın; IBM MQ for Windows özel bir şey gerektirmez.

64 bitlik programlama uygulamaları hakkında daha fazla bilgi için bkz. 64 bitlik platformlarda kodlama standartları.

- Programlarınızı IBM MQtarafından sağlanan uygun kitaplıklara bağlayın:

Kitaplık dosyası	**Program/çıkış tipi**
MQ_INSTALLATION_PATH `\Tools\Lib\mqm.lib`	32 bit C için sunucu
MQ_INSTALLATION_PATH `\Tools\Lib\mqic.lib`	32 bit C için istemci

Kitaplık dosyası	Program/çıkış tipi
MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib	işlem koordinasyonuna sahip 32 bit C için istemci
MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib	64 bit C için sunucu
MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib	64 bit C için istemci
MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib	işlem koordinasyonuna sahip 64 bit C için istemci

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Aşağıdaki komut, amqsget0 örnek programının (Microsoft Visual C++ derleyicisini kullanarak) derlenmesinin bir örneğini verir.

32 bit uygulamalar için:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib
```

64 bit uygulamalar için:

```
cl -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib
```

Not:

- Kurulabilir bir hizmet yazıyorsanız (ek bilgi için bkz. IBM MQ Yönetimi), mqmzf.lib kitaplığına bağlanmanız gerekir.
- IBM TXSeries Encina ya da BEA Tuxedo gibi XA uyumlu bir hareket yöneticisi tarafından dış eşgüdüm için bir uygulama üretiyorsanız, mqmxa.lib ya da mqmxa.lib kitaplığına bağlanmanız gerekir.
- Bir CICS çıkışı yazıyorsanız, mqmcics4.lib kitaplığına bağlantı oluşturun.
- IBM MQ kitaplıklarını diğer ürün kitaplıklarına bağlamanız gerekir.

- DLL ' ler belirttiğiniz yolda (PATH) olmalıdır.
- Mümkün olduğunda küçük harfli karakterler kullanırsanız, IBM MQ for Windows sisteminden IBM MQ for AIX or Linux sistemine geçebilirsiniz; burada küçük harf kullanımı gereklidir.

CICS ve Transaction Server programlarının hazırlanması

Bir CICS IBM MQ işlemine ilişkin örnek C kaynağı AMQSCIC0.CCS. Standart CICS olanaklarını kullanarak oluşturun. Örneğin, Windows 2000için TXSeries için:

1. Ortam değişkenini ayarlayın (bir satırda aşağıdaki kodu girin):

   ```
   set CICS_IBMC_FLAGS=-I MQ_INSTALLATION_PATH\Tools\C\Include;
   %CICS_IBMC_FLAGS%
   ```

2. USERLIB ortam değişkenini ayarlayın:

```
set USERLIB=MQM.LIB;%USERLIB%
```

3. Örnek programı çevir, derle ve bağla:

```
cicstcl -l IBMC amqscic0.ccs
```

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Bu, *Transaction Server for Windows NT Application Programming Guide (CICS) belgesinde açıklanmıştır. V4*.

IBM MQ Yönetimiiçinde CICS hareketlerini desteklemeye ilişkin daha fazla bilgi bulabilirsiniz.

Windows *Windows ' da COBOL programlarını hazırlama*

Windowsiçinde COBOL programlarını hazırlamak ve CICS ve Transaction Server programlarını hazırlamak için bu bilgileri kullanın.

1. 32 bit COBOL kopya kitapları şu dizine kurulur: *MQ_INSTALLATION_PATH* \Tools\cobol\CopyBook.
2. 64 bitlik COBOL kopya kitapları şu dizine kurulur: *MQ_INSTALLATION_PATH* \Tools\cobol\CopyBook64
3. Aşağıdaki örneklerde CopyBook değerini şuna ayarlayın:

```
CopyBook
```

   32 bit uygulamalar için ve:

```
CopyBook64
```

   64 bit uygulamalar için.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Windows sistemlerinde COBOL programlarını hazırlamak için, programınızı IBM MQtarafından sağlanan aşağıdaki kitaplıklardan birine bağlayın:

Kitaplık dosyası	Program ya da çıkış tipi
MQ_INSTALLATION_PATH \Tools\Lib\mqmcb	Micro Focus COBOL için 32 bit sunucu
MQ_INSTALLATION_PATH \Tools\Lib\mqiccb	Micro Focus COBOL için 32 bit istemci
MQ_INSTALLATION_PATH \Tools\Lib64\mqmcb	Micro Focus COBOL için 64 bit sunucu
MQ_INSTALLATION_PATH \Tools\Lib64\mqiccb	Micro Focus COBOL için 64 bit istemci

MQI istemcisi ortamında bir program çalıştırırken, DOSCALLS kitaplığının herhangi bir COBOL ya da IBM MQ kitaplığından önce göründüğünden emin olun.

Micro Focus COBOL kullanarak COBOL programlarını hazırlama

Var olan 32 bit IBM MQ Micro Focus COBOL programlarını mqmcbb ve mqiccbb kitaplıkları yerine mqmcb.lib ya da mqiccb.libkullanarak yeniden bağlayın.

Örneğin, Micro Focus COBOL kullanarak amq0put0örnek programını derlemek için:

1. COBCPY ortam değişkenini IBM MQ COBOL kopya kitaplarını gösterecek şekilde ayarlayın (bir satırda aşağıdaki kodu girin):

```
set COBCPY= MQ_INSTALLATION_PATH\
Tools\Cobol\Copybook
```

2. Bir nesne dosyası vermek için programı derleyin:

```
cobol amq0put0 LITLINK
```

3. Nesne dosyasını çalıştırma zamanı sistemine bağlayın.
   - LIB ortam değişkenini derleyici COBOL kitaplıklarını gösterecek şekilde ayarlayın.
   - Nesne dosyasını IBM MQ sunucusunda kullanılmak üzere bağlayın:

     ```
     cbllink amq0put0.obj mqmcb.lib
     ```

   - Ya da nesne dosyasını IBM MQ istemcisinde kullanmak üzere bağlayın:

     ```
     cbllink amq0put0.obj mqiccb.lib
     ```

CICS ve Transaction Server programlarının hazırlanması

Bir TXSeries for Windows NT, V5.1 programını IBM VisualAge COBOL kullanarak derlemek ve bağlamak için:

1. Ortam değişkenini ayarlayın (bir satırda aşağıdaki kodu girin):

```
set CICS_IBMCOB_FLAGS= MQ_INSTALLATION_PATH\
Cobol\Copybook\VAcobol;%CICS_IBMCOB_FLAGS%
```

2. USERLIB ortam değişkenini ayarlayın:

```
set USERLIB=MQMCBB.LIB
```

3. Programınızı çevirin, derleyin ve birbirine bağlayın:

```
cicstcl -l IBMCOB myprog.ccp
```

Bu, *Transaction Server for Windows NT, V4 Application Programming* Guide belgesinde açıklanmaktadır.

Micro Focus COBOL kullanarak bir CICS for Windows V5 programını derlemek ve bağlamak için:

- INCLUDE değişkenini ayarlayın:

```
set
INCLUDE=drive:\programname\ibm\websphere\tools\c\include;
        drive:\opt\cics\include;%INCLUDE%
```

- COBCPY ortam değişkenini ayarlayın:

```
setCOBCPY=drive:\programname\ibm\websphere\tools\cobol\copybook;
          drive:\opt\cics\include
```

- COBOL seçeneklerini ayarlayın:
  - `set`
  - `COBOPTS=/LITLINK /NOTRUNC`

ve aşağıdaki kodu çalıştırın:

```
cicstran cicsmq00.ccp
cobol cicsmq00.cbl /LITLINK /NOTRUNC
cbllink -D -Mcicsmq00 -Ocicsmq00.cbmfnt cicsmq00.obj
%CICSLIB%\cicsprCBMFNT.lib user32.lib msvcrt.lib kernel32.lib mqmcb.lib
```

Windows *Windows ' da Visual Basic programlarının hazırlanması*

Windowsüzerinde Microsoft Visual Basic programlarını kullanırken göz önünde bulundurulması gereken bilgiler.

Deprecated IBM MQ 9.0' den Microsoft Visual Basic 6.0 desteği kullanımdan kaldırılmıştır. .NET için IBM MQ sınıfları önerilen değiştirme teknolojisidir. Daha fazla bilgi için bkz .NET uygulamaları geliştirilmesi.

Not: Visual Basic modül dosyalarının 64 bit sürümleri sağlanmaz.

Windowsüzerinde Visual Basic programlarını hazırlamak için:

1. Yeni bir proje yaratır.
2. Sağlanan modül dosyasını (CMQB.BAS, projeye.
3. Gerekiyorsa, sağlanan diğer modül dosyalarını ekleyin:
   - CMQBB.BAS: MQAI desteği
   - CMQCFB.BAS: PCF desteği
   - CMQXB.BAS: Kanal çıkışları desteği
   - CMQPSB.BAS: Yayınla/abone ol

Visual Basiciçinden MQCONNXAny çağrıyı kullanma hakkında bilgi için bkz. “Visual Basic içinde kodlama” sayfa 1008 .

Proje kodunda MQI çağrıları yapmadan önce MQ_SETDEFAULTS yordamını çağırın. Bu yordam, MQI çağrılarının gerektirdiği varsayılan yapıları ayarlar.

Koşullu derleme değişkenini *MqType*ayarlayarak, projeyi derlemeden ya da çalıştırmadan önce bir IBM MQ sunucusu ya da istemcisi yaratıp yaratmayacağınızı belirtin. Bir Visual Basic projesinde sunucu için *MqType* değerini 1 olarak ya da istemci için 2 olarak ayarlayın:

1. Proje menüsünü seçin.
2. *Name* Özellikler (burada *Name* , yürürlükteki projenin adıdır) seçeneğini belirleyin.
3. İletişim kutusunda Make etiketini seçin.
4. Koşullu Derleme Bağımsız Değişkenleri alanına bir sunucu için şunu girin:

   ```
   MqType=1
   ```

   ya da bir istemci için:

   ```
   MqType=2
   ```

İlgili kavramlar

“Visual Basic içinde kodlama” sayfa 1008
Microsoft Visual Basiciçinde IBM MQ programlarını kodlarken göz önünde bulundurulması gereken bilgiler. Visual Basic yalnızca Windowsüzerinde desteklenir.

İlgili başvurular

“Visual Basic uygulamalarının IBM MQ MQI client koduyla bağlanması” sayfa 881
Microsoft Visual Basic uygulamalarını Windowsüzerindeki IBM MQ MQI client koduyla bağlayabilirsiniz.

Windows *SSPI güvenlik çıkışı*

IBM MQ for Windows , hem IBM MQ MQI client hem de IBM MQ sunucusu için bir güvenlik çıkışı sağlar. Bu, Güvenlik Hizmetleri Programlama Arabirimi 'ni (SSPI) kullanarak IBM MQ kanalları için kimlik doğrulaması sağlayan bir kanal çıkış programıdır. SSPI, Windows sistemlerinin tümleşik güvenlik olanaklarını sağlar.

Güvenlik paketleri security.dll ya da secur32.dll' den yüklenir. Bu DLL ' ler işletim sisteminizle birlikte sağlanır.

Tek yönlü kimlik doğrulama, NTLM kimlik doğrulama hizmetleri kullanılarak sağlanır. İki yönlü kimlik doğrulama, Kerberos kimlik doğrulama hizmetleri kullanılarak sağlanır.

Güvenlik çıkış programı kaynak ve nesne biçiminde sağlanır. Nesne kodunu olduğu gibi kullanabilir ya da kaynak kodu başlangıç noktası olarak kullanarak kendi kullanıcı çıkış programlarınızı yaratabilirsiniz.

Ayrıca bkz. "Windows üzerinde SSPI güvenlik çıkışının kullanılması" sayfa 1086.

Güvenlik çıkışlarına giriş

Bir güvenlik çıkışı, iki güvenlik çıkış programı arasında güvenli bir bağlantı oluşturur; burada bir program, gönderen ileti kanalı aracısı (MCA) için, diğeri ise alan MCA içindir.

Güvenli bağlantıyı başlatan program (MCA oturumu kurulduktan sonra denetimi alacak ilk program), *bağlam başlatıcısı*olarak bilinir. İş ortağı programı, *bağlam kabul edici*olarak bilinir.

Aşağıdaki tabloda, bağlam başlatıcıları ve ilişkili bağlam alıcıları olan kanal tiplerinden bazıları gösterilmektedir.

Çizelge 149. Bağlam başlatıcılar ve ilişkili bağlam alıcıları

Bağlam Başlatıcısı	**Bağlam Kabul Eden**
MQCHT_CLNTCONN	MQCHT_SVRCONN
MQCHT_ALICI	MQCHT_SENDER
MQCHT_CLUSRCVR	MQCHT_CLUSSDR

Güvenlik çıkış programının iki giriş noktası vardır:

- **SCY_NTLM**

  Bu, tek yönlü kimlik doğrulaması sağlayan NTLM kimlik doğrulama hizmetlerini kullanır. NTLM, sunucuların istemcilerinin kimliklerini doğrulamasını sağlar. İstemcilerin bir sunucunun kimliğini ya da bir sunucunun başka bir sunucunun kimliğini doğrulamasını sağlamaz. NTLM kimlik doğrulaması, sunucuların gerçek olduğu kabul edilen bir ağ ortamı için tasarlanmıştır.

- **SCY_KERBEROS**

  Bu, Kerberos karşılıklı kimlik doğrulama hizmetlerini kullanır. Kerberos iletişim kuralı, bir ağ ortamındaki sunucuların gerçek olduğunu varsaymaz. Bir ağ bağlantısının her iki ucundaki taraflar, diğer tarafın kimliğini doğrulayabilir. Yani, sunucular istemcilerin ve diğer sunucuların kimliğini doğrulayabilir ve istemciler bir sunucunun kimliğini doğrulayabilir.

Güvenlik çıkışının ne işe yaradığı

Bu konuda, SSPI kanal çıkış programlarının ne işi olduğu açıklanmaktadır.

Sağlanan kanal çıkış programları, bir oturum kurulurken ortak bir sistemin tek yönlü ya da iki yönlü (karşılıklı) kimlik doğrulamasını sağlar. Belirli bir kanal için, her çıkış programının ilişkili bir *asıl adı* vardır (kullanıcı kimliğine benzer, bkz. "IBM MQ erişim denetimi ve Windows asıl adları" sayfa 979). İki çıkış programı arasındaki bağlantı, iki birincil kullanıcı arasındaki bir ilişkidir.

Temel oturum kurulduktan sonra, iki güvenlik çıkış programı (biri gönderen MCA için, diğeri alan MCA için) arasında güvenli bir bağlantı kurulur. İşlem sırası aşağıdaki gibidir:

1. Her program belirli bir birincil kullanıcıyla ilişkilendirilir; örneğin, belirtik bir oturum açma işleminin sonucu olarak.
2. Bağlam başlatıcı, güvenlik paketinden (Kerberos, adı belirtilen ortak için) iş ortağıyla güvenli bir bağlantı ister ve bir belirteç (token1olarak adlandırılır) alır. Simge, önceden oluşturulan temel oturum kullanılarak iş ortağı programına gönderilir.

3. Ortak program (bağlam kabul edici), token1 öğesini güvenlik paketine geçirir ve bağlam başlatıcının gerçek olduğunu doğrular. NTLM için bağlantı kuruldu.
4. Kerberostarafından sağlanan güvenlik çıkışı (karşılıklı kimlik doğrulaması için) için, güvenlik paketi ayrıca, bağlam alıcının temel oturumu kullanarak bağlam başlatıcısına döndürdüğü ikinci bir simge (token2olarak adlandırılır) oluşturur.
5. Bağlam başlatıcı, bağlam alıcının gerçek olduğunu doğrulamak için token2 öğesini kullanır.
6. Bu aşamada, her iki uygulama da ortağın belirtecinin doğruluğu konusunda memnunsa, güvenli (kimliği doğrulanmış) bağlantı kurulur.

IBM MQ erişim denetimi ve Windows asıl adları

IBM MQ ' in sağladığı erişim denetimi, kullanıcı ve gruba dayalıdır. Windows tarafından sağlanan kimlik doğrulaması, kullanıcı ve servicePrincipalAd (SPN) gibi asıl adlara dayalıdır. servicePrincipalAdı söz konusu olduğunda, bunlardan tek bir kullanıcıyla ilişkilendirilmiş çok sayıda olabilir.

SSPI güvenlik çıkışı, kimlik doğrulaması için ilgili Windows asıl adlarını kullanır. Windows kimlik doğrulaması başarılı olursa, çıkış, Windows birincil kullanıcısıyla ilişkili kullanıcı kimliğini erişim denetimi için IBM MQ ' e aktarır.

Kimlik doğrulamasıyla ilgili Windows birincil kullanıcıları, kullanılan kimlik doğrulamasının tipine bağlı olarak değişiklik gösterir.

- NTLM kimlik doğrulaması için, bağlam başlatıcısı için Windows birincil kullanıcısı, çalışmakta olan süreçle ilişkilendirilmiş kullanıcı kimliğidir. Bu kimlik doğrulaması tek bir yol olduğundan, Bağlam Kabul Eden ile ilişkili birincil kullanıcı ilgisiz.
- Kerberos kimlik doğrulaması için, CLNTCONN kanallarında Windows asıl adı, çalışmakta olan işlemle ilişkilendirilmiş kullanıcı kimliğidir. Tersi durumda, Windows asıl adı, QueueManagerAdına aşağıdaki önek eklenerek oluşturulan servicePrincipaladıdır.

```
ibmMQSeries/
```

z/OS z/OS üzerinde yordamsal uygulamanızı oluşturma

CICS, IMSve z/OS yayınları, bu ortamlarda çalışan uygulamaların nasıl oluşturulacağını açıklar.

Bu konu derlemi, bu ortamlar için IBM MQ for z/OS uygulamaları oluştururken gerçekleştirmeniz gereken ek görevleri ve standart görevlerdeki değişiklikleri açıklar. COBOL, C, C + +, Assembler ve PL/I programlama dilleri desteklenir. (C++ uygulamaları oluşturma hakkında bilgi için bkz. C++ kullanarak.)

Yürütülebilir IBM MQ for z/OS uygulaması yaratmak için gerçekleştirmeniz gereken görevler, programın yazıldığı programlama diline ve uygulamanın çalışacağı ortama bağlıdır.

Programınızdaki MQI çağrılarını kodlamaya ek olarak, kullandığınız dile ilişkin IBM MQ for z/OS veri tanımlama dosyasını eklemek için uygun dil deyimlerini ekleyin. Bu dosyaların içeriğini tanıtın. Tam açıklama için bkz. “IBM MQ veri tanımlama dosyaları” sayfa 688 .

Not

thlqual adı, z/OSüzerindeki kuruluş kitaplığının üst düzey niteleyicidir.

z/OS *Programınızın çalıştırılmak üzere hazırlanması*

IBM MQ uygulamanızın yürütülebilir bir uygulama oluşturması için programı yazdıktan sonra, programı derlemeniz ya da derlemeniz, ardından sonuçtaki nesne kodunu IBM MQ for z/OS ' un desteklediği her ortam için sağladığı sınırlı kod öbeği programıyla ilişkilendirmeniz gerekir.

Programınızı nasıl hazırlayacağınız, uygulamanın çalıştığı ortama (toplu iş, CICS, IMS(BMP ya da MPP), Linux ya da z/OS UNIX System Services) ve z/OS kurulumunuzda veri kümelerinin yapısına bağlıdır.

"IBM MQ sınırlı kod öbeğini dinamik olarak çağırma" sayfa 986 , bir IBM MQ sınırlı kod öbeğine bağlantı düzenlemenize gerek kalmaması için programlarınızda MQI çağrıları yapmanın alternatif bir yöntemini açıklar. Bu yöntem tüm diller ve ortamlar için kullanılamaz.

Programınızın çalıştığı IBM MQ for z/OS sürümünden daha yüksek düzeydeki bir sınırlı kod öbeği programını bağlamayın. Örneğin, OS/390için MQSeries üzerinde çalışan bir program V5.2 , IBM MQ for z/OS V7ile verilen bir sınırlı kod öbeği programıyla bağlantı düzenlenmemelidir.

z/OS *64 bit C uygulamaları oluşturuluyor*

z/OSiçinde, 64 bit C uygulamaları LP64 derleyici ve bağlayıcı seçenekleri kullanılarak oluşturulur. IBM MQ for z/OS *cmqc.h* üstbilgi dosyası, derleyiciye bu seçenek sağlandığında tanır ve 64 bit işleme uygun IBM MQ veri tipleri ve yapıları oluşturur.

Bu seçenekle oluşturulan C kodu, gereken eşgüdüm anlambilimi için uygun dinamik bağlantı kitaplıklarını (DLL ' ler) kullanacak şekilde oluşturulmalıdır. Bunu gerçekleştirmek için, derlenen kodu aşağıdaki çizelgede tanımlanan uygun yan desteyle bağlayın:

Çizelge 150. Her eşgüdüm anlambilimi için gereken yan deste adı

coordination	**Yan deste adı**
Tek aşamalı kesinleştirme MQI	CSQBMQ2X
RRS fiillerini kullanarak RRS koordinasyonuyla iki aşamalı kesinleştirme	CSQBRR2X
MQI fiillerini kullanarak RRS koordinasyonuyla iki aşamalı kesinleştirme	CSQBRI2X

Not: 31 bit C uygulamaları için, çağıran arabirime (Dil Ortamı ya da XPLINK) ilişkin derleyici seçeneklerini "31 bit Dil Ortamı ya da XPLINK kullanarak z/OS toplu iş uygulamaları oluşturma" sayfa 982başlıklı konuda açıklandığı gibi ayarlayın. Desteklenen tek bağlantı XPLINKolduğundan, 64 bitlik C uygulamaları için çağıran arabirimi belirtmezsiniz.

*z/OS XL C/C++*ile verilen EDCQCB JCL yordamını kullanarak, toplu iş olarak tek aşamalı bir IBM MQ programı oluşturun:

```
//PROCS  JCLLIB ORDER=CBC.SCCNPRC
//CLG EXEC EDCQCB,
// INFILE='thlqual.SCSQC37S(CSQ4BCG1)',   < MQ SAMPLES
// CPARM='RENT,SSCOM,DLL,LP64,LIST,NOMAR,NOSEQ',  < COMPILER OPTIONS
// LIBPRFX='CEE',            < PREFIX FOR LIBRARY DSN
// LNGPRFX='CBC',            < PREFIX FOR LANGUAGE DSN
// BPARM='MAP,XREF,RENT,DYNAM=DLL', < LINK EDIT OPTIONS
// OUTFILE='userid.LOAD(CSQ4BCG1),DISP=SHR'
//COMPILE.SYSLIB DD
//        DD
//        DD DISP=SHR,DSN=thlqual.SCSQC370
//BIND.SCSQDEFS DD DISP=SHR,DSN=thlqual.SCSQDEFS
//BIND.SYSIN DD *
INCLUDE SCSQDEFS(CSQBMQ2X)
NAME CSQ4BCG1
```

z/OS UNIX System Servicesiçinde bir RRS eşgüdümlü programı oluşturmak için aşağıdaki şekilde derleyin ve bağlantı oluşturun:

```
cc -o mqsamp -W c,LP64,DLL -W l,DYNAM=DLL,LP64 -I"//'thlqual.SCSQC370'" "//'thlqual.SCSQDEFS(CSQBRR2X)'" mqsamp.c
```

z/OS *z/OS toplu iş uygulamaları oluşturma*

z/OS toplu iş uygulamalarını nasıl oluşturacağınızı ve bunu yaparken dikkate alınacak adımları öğrenin.

IBM MQ for z/OS için z/OS toplu iş altında çalışan bir uygulama oluşturmak üzere şu görevleri gerçekleştiren iş denetim dili (JCL) oluşturun:

1. Nesne kodu üretmek için programı derleyin (ya da birleştirin). Derlemeniz için JCL, ürün verileri tanımlama dosyalarını derleyicinin kullanımına sağlayan SYSLIB deyimlerini içermelidir. Veri tanımlamaları aşağıdaki IBM MQ for z/OS kitaplıklarında sağlanır:

- COBOL için, **thlqual**.SCSQCOBC
- Çevirici dili için, **thlqual**.SCSQMACS
- C için, **thlqual**.SCSQC370
- PL/I için, **thlqual**.SCSQPLIC

2. Bir C uygulaması için, “1” sayfa 980. adımda oluşturulan nesne kodunu önceden bağlayın.
3. PL/I uygulamaları için, EXTRN (SHORT) derleyici seçeneğini kullanın.
4. Bir yükleme modülü oluşturmak için “1” sayfa 980 . adımda (ya da bir C uygulaması için “2” sayfa 981 . adımda) yaratılan nesne kodunu düzenleyin. Kodu düzenlerken, IBM MQ for z/OS toplu iş kod parçası programlarından birini (CSQBSTUB ya da RRS sınırlı kod öbeği programlarından birini (CSQBRRSI ya da CSQBRSTB) eklemeniz gerekir.

   CSQBSTUB
   IBM MQ for z/OS tarafından sağlanan tek aşamalı kesinleştirme

   CSQBRRSI
   MQI kullanılarak RRS tarafından sağlanan iki aşamalı kesinleştirme

   CSQBRSTB
   doğrudan RRS tarafından sağlanan iki aşamalı kesinleştirme

   Notlar:

   a. CSQBRSTB kullanıyorsanız, uygulamanızı SYS1.CSSLIB. Şekil 113 sayfa 981 ve Şekil 114 sayfa 981 , bunu yapmak için JCL parçalarını gösterir. Sınırlı kod öbekleri dilden bağımsızdır ve **thlqual**.SCSQLOAD kitaplığında sağlanır.

   b. Uygulamanız Dil Ortamı altında çalışıyorsa, “31 bit Dil Ortamı ya da XPLINK kullanarak z/OS toplu iş uygulamaları oluşturma” sayfa 982başlıklı konuda açıklandığı gibi Dil Ortamı DLL 'si ile bağlantı düzenlediğinizden emin olmanız gerekir.

5. Yükleme modülünü bir uygulama yükleme kitaplığında saklayın.

```
.
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING BATCH STUB
//*
//CSQSTUB   DD  DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
//*
.
//SYSIN     DD *
  INCLUDE CSQSTUB(CSQBSTUB)
.
/*
```

Şekil 113. Tek aşamalı kesinleştirmeyi kullanarak toplu ortamda nesne modülünü düzenlemek için JCL parçaları

```
.
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING BATCH STUB
//*
//CSQSTUB   DD  DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
//CSSLIB    DD  DSN=SYS1.CSSLIB,DISP=SHR
//*
.
//SYSIN     DD *
INCLUDE CSQSTUB(CSQBRSTB)
INCLUDE CSSLIB(ATRSCSS)
.
/*
```

Şekil 114. İki aşamalı kesinleştirmeyi kullanarak toplu ortamda nesne modülünü düzenlemek için JCL parçaları

Bir toplu iş ya da RRS programını çalıştırmak için, STEPLIB ya da JOBLIB veri kümesi birleşiminde **thlqual**.SCSQAUTH ve **thlqual**.SCSQLOAD kitaplıklarını eklemeniz gerekir.

Bir TSO programını çalıştırmak için, TSO oturumu tarafından kullanılan STEPLIB ' e **thlqual**.SCSQAUTH ve **thlqual**.SCSQLOAD kitaplıklarını eklemeniz gerekir.

z/OS UNIX System Services kabuğundan bir toplu iş programını çalıştırmak için, **thlqual**.SCSQAUTH ve **thlqual**.SCSQLOAD kitaplıklarını $HOME? .profilinizdeki STEPLIB belirtimine ekleyin:

```
STEPLIB= thlqual.SCSQAUTH: thlqual.SCSQLOAD
export STEPLIB
```

z/OS *31 bit Dil Ortamı ya da XPLINK kullanarak z/OS toplu iş uygulamaları oluşturma*

IBM MQ for z/OS , uygulamalarınız için bağlantı düzenlerken kullanılması gereken dinamik bağlantı kitaplıkları (DLL) kümesini sağlar.

Uygulamanın aşağıdaki çağıran arabirimlerden birini kullanmasına izin veren iki kitaplık değişkeni vardır:

- 31 bitlik Dil Ortamı çağıran arabirimi.
- 31 bit XPLINK arama arabirimi. z/OS XPLINK, C uygulamaları için kullanılabilen yüksek performanslı bir çağrı kuralıdır. z/OS 2.2 belgelerinde XPLINK | NOXPLINK bölümüne bakın.

DLL ' leri kullanmak için, uygulama daha önceki sürümlerle sağlanan sınırlı kod öbekleri yerine *sidedecks*(yan desteler) olarak adlandırılan öğelere bağlanır. Yan desteler SCSQDEFS kitaplığında (SCSQLOAD kitaplığı yerine) bulunur.

Çizelge 151. Dinamik bağlantı kitaplıklarının çeşitlemeleri

Kesinleştir	31 bitlik Dil Ortamı DLL 'si	31 bit XPLINK DLL	Eşdeğer sınırlı kod öbeği adı
1 aşamalı kesinleştirme MQI kitaplıkları	CSQBMQ1	CSQBMQ1X	CSQBSTUB
RRS işlem denetimi fiillerini kullanarak RRS koordinasyonu ile 2 aşamalı kesinleştirme	CSQBRR1	CSQBRR1X	CSQBRSTB
MQI hareket denetimi fiillerini kullanarak RRS eşgüdüm ile 2 aşamalı kesinleştirme	CSQBRI1	CSQBRI1X	CSQBRRSI

Not: Tüm yan desteler, daha önce CSQASTUB eklenerek çözülen MQXCNVC veri dönüştürme giriş noktası tanımlamasını içerir.

Genel sorunlar:

- Uygulamanız zamanuyumsuz ileti kullanımı (MQCB, MQCTL ya da MQSUB çağrıları) kullanıyorsa ve önceki DLL arabirimi kullanılmıyorsa, iş günlüğünde aşağıdaki ileti görüntülenir:

  ```
  CSQB001E z/OS toplu işte çalışan dil ortamı programları ya da z/OS UNIX System Services , IBM
  MQ için DLL arabirimini kullanmalıdır.
  ```

  Çözüm: Daha önce ayrıntılı olarak açıklandığı gibi, sınırlı kod öbekleri yerine yan tarafları kullanarak uygulamanızı yeniden oluşturun.

- Program oluşturma sırasında aşağıdaki ileti görüntülenir

  ```
  IEW2469E kodunuz bölümündeki MQAPI-NAME öğesine ilişkin bir başvurunun öznitelikleri
  hedef simgesi
  ```

  Neden: Bu, XPLINK programınızı cmqc.h' nin V701 (ya da daha sonraki bir sürümü) sürümüyle derlediğiniz, ancak kenar desteleriyle bağ tanımlamadığınız anlamına gelir.

Çözüm: Programınızın oluşturma dosyasını SCSQLOAD sınırlı kod öbeği yerine SCSQDEFS ' den uygun yan desteye bağlanacak şekilde değiştirin

Aşağıdaki örnek JCL, 31 bit Dil Ortamı DLL çağıran arabirimi kullanmak üzere bir C programını nasıl derleyebileceğinizi ve bağlayabileceğinizi göstermektedir:

```
//CLG EXEC EDCCB,
//  INFILE=MYPROGS.CPROGS(MYPROGRAM),
//  CPARM='OPTF(DD:OPTF)',
//  BPARM='XREF,MAP,DYNAM=DLL'           < LINKEDIT OPTIONS
//COMPILE.OPTF DD *
RENT,CHECKOUT(ALL),SSCOM,DEFINE(MVS),NOMARGINS,NOSEQ,DLL
SE(DD:SYSLIBV)
//COMPILE.SYSLIB DD
//               DD
//               DD DISP=SHR,DSN=hlq.SCSQC370
//COMPILE.SYSLIBV DD DISP=SHR,DSN=hlq.BASE.H
/*
//BIND.SYSOBJ DD DISP=SHR,DSN=CEE.SCEEOBJ
//            DD DISP=SHR,DSN=hlq.SCSQDEFS
//BIND.SYSLMOD DD DISP=SHR,DSN=hlq.LOAD(MYPROGAM)
//BIND.SYSIN  DD *
 ENTRY CEESTART
 INCLUDE SYSOBJ(CSQBMQ1)
 NAME MYPROGAM(R)
//
```

Not: Derleme işlemi **DLL** seçeneğini kullanır. Bağlantı düzenleme **DYNAM=DLL** seçeneğini kullanır ve **CSQBMQ1** kitaplığına başvurur.

Aşağıdaki örnek JCL, 31 bit XPLINK DLL çağıran arabirimi kullanmak üzere bir C programını nasıl derleyebileceğinizi ve bağlayabileceğinizi göstermektedir:

```
//CLG  EXEC EDCXCB,
//  INFILE=MYPROGS.CPROGS(MYPROGRAM),
//  CPARM='OPTF(DD:OPTF)',
//  BPARM='XREF,MAP,DYNAM=DLL'           < LINKEDIT OPTIONS
//COMPILE.OPTF DD *
RENT,CHECKOUT(ALL),SSCOM,DEFINE(MVS),NOMARGINS,NOSEQ,XPLINK,DLL
SE(DD:SYSLIBV)
//COMPILE.SYSLIB DD
//               DD
//               DD DISP=SHR,DSN=hlq.SCSQC370
//COMPILE.SYSLIBV DD DISP=SHR,DSN=hlq.BASE.H
/*
//BIND.SYSOBJ DD DISP=SHR,DSN=CEE.SCEEOBJ
//            DD DISP=SHR,DSN=hlq.SCSQDEFS
//BIND.SYSLMOD DD DISP=SHR,DSN=hlq.LOAD(MYPROGAM)
//BIND.SYSIN  DD *
 ENTRY CEESTART
 INCLUDE SYSOBJ(CSQBMQ1X)
 NAME MYPROGAM(R)
//
```

Not: Derleme, **XPLINK** ve **DLL** seçeneklerini kullanır. Link-edit **DYNAM=DLL** seçeneğini kullanır ve **CSQBMQ1X** kitaplığına başvurur.

Birimdeki her programa DLL derleme seçeneğini eklediğinizden emin olun. IEW2456E 9207 SYMBOL CSQ1BAK UNÇÖZÜLMEMIŞ gibi iletiler, tüm programların DLL seçeneğiyle derlendiğini denetlemeniz gerektiğini gösterir.

z/OS *z/OS içinde CICS uygulamaları oluşturma*

z/OSiçinde CICS uygulamaları oluştururken bu bilgileri kullanın.

CICSaltında çalışan IBM MQ for z/OS için bir uygulama oluşturmak üzere aşağıdakileri yapmanız gerekir:

- Programınızdaki CICS komutlarını, programınızın geri kalanının yazıldığı dile çevirin.
- Nesne kodu üretmek için çevirmenin çıkışını derleyin ya da derleyin.
  - PL/I programları için, EXTRN (SHORT) derleyici seçeneğini kullanın.

  - C uygulamaları için, uygulama XPLINKkullanmıyorsa, DEFINE (MQ_OS_LINKAGE=1) derleyici seçeneğini kullanın.
- Bir yükleme modülü yaratmak için nesne kodunu düzenleyin.

CICS , desteklediği programlama dillerinin her biri için sırayla bu adımları yürütmek üzere bir yordam sağlar.

- CICS Transaction Server for z/OSiçin *CICS Transaction Server for z/OS System Definition Guide* , bu yordamların nasıl kullanılacağını açıklar ve *CICS/ESA Application Programming Guide* , çeviri işlemiyle ilgili daha fazla bilgi verir.

Şunları eklemeniz gerekir:

- Derleme (ya da derleme) aşamasının SYSLIB deyiminde, ürün verileri tanımlama dosyalarını derleyicinin kullanımına sağlayan deyimler. Veri tanımlamaları aşağıdaki IBM MQ for z/OS kitaplıklarında sağlanır:
  - COBOL için, **thlqual**.SCSQCOBC
  - Çevirici dili için, **thlqual**.SCSQMACS
  - C için, **thlqual**.SCSQC370
  - PL/I için, **thlqual**.SCSQPLIC
- IBM MQ for z/OS CICS sınırlı kod öbeği programı (CSQCSTUB), bağlantı düzenleme JCL ' inizde. Şekil 115 sayfa 984 , bunu yapmak için JCL kodunun parçalarını gösterir. Sınırlı kod öbeği dilden bağımsızdır ve **thlqual**.SCSQLOAD kitaplığında sağlanır.

```
⋮
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING CICS STUB
//*
//CSQSTUB   DD  DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
//*
⋮
//LKED.SYSIN DD *
  INCLUDE CSQSTUB(CSQCSTUB)
  ⋮
/*
```

Şekil 115. CICS ortamında nesne modülünü bağlamak için JCL parçaları

- CICS TS 3.2sürümünden sonraki CICS sürümleri için ya da IBM MQ ileti özelliği API 'lerini ya da IBM MQ API' lerini kullanmak istiyorsanız, MQCB, MQCTL, MQSTAT, MQSUB ya da MQSUBR, nesne kodunuzu, IBM MQ tarafından sağlanan CSQCSTUB ile değil, CICS tarafından sağlanan sınırlı kod öbeğiyle (DFHMQSTB) eşlemelisiniz. CICSiçin IBM MQ programları oluşturulmasıyla ilgili daha fazla bilgi için CICS ürün belgelerinde API stub program to access IBM MQ MQI çağrılarına bakın.

Bu adımları tamamladığınızda, yükleme modülünü bir uygulama yükleme kitaplığında saklayın ve programı CICS olarak tanımlayın.

Bir CICS programını çalıştırmadan önce, sistem yöneticiniz programı CICS olarak IBM MQ program ve hareket olarak tanımlamalıdır; daha sonra bu programı tipik olarak çalıştırabilirsiniz.

z/OS *IMS (BMP ya da MPP) uygulamalarını oluşturma*

IMS (BMP ya da MPP) uygulamalarını oluştururken bu bilgileri kullanın.

Toplu DL/I programları oluşturuyorsanız, bkz. “z/OS toplu iş uygulamaları oluşturma” sayfa 980.
IMS altında (BMP ya da MPP olarak) çalışan başka uygulamalar oluşturmak için, aşağıdaki görevleri gerçekleştiren JCL oluşturun:

1. Nesne kodu üretmek için programı derleyin (ya da birleştirin). Derlemeniz için JCL, ürün verileri tanımlama dosyalarını derleyicinin kullanımına sağlayan SYSLIB deyimlerini içermelidir. Veri tanımlamaları aşağıdaki IBM MQ for z/OS kitaplıklarında sağlanır:
   - COBOL için, **thlqual**.SCSQCOBC
   - Çevirici dili için, **thlqual**.SCSQMACS

- C için, **thlqual**.SCSQC370
- PL/I için, **thlqual**.SCSQPLIC

2. Bir C uygulaması için, “1” sayfa 984. adımda yaratılan nesne modülünü önceden bağlayın.
3. PL/I programları için, EXTRN (SHORT) derleyici seçeneğini kullanın.
4. Bir C uygulaması için, uygulama XPLINKkullanmıyorsa, DEFINE (MQ_OS_LINKAGE=1) derleyici seçeneğini kullanın.
5. Bir yükleme modülü oluşturmak için “1” sayfa 984 . adımda (ya da C/370 uygulaması için “2” sayfa 985 . adımda) yaratılan nesne kodunu düzenleyin:
   a. IMS dil arabirimi modülünü (DFSLI000) ekleyin.
   b. IBM MQ for z/OS IMS sınırlı kod öbeği programını (CSQQSTUB) ekleyin. Şekil 116 sayfa 985 , bunu yapmak için JCL parçalarını gösterir. Sınırlı kod öbeği dilden bağımsızdır ve **thlqual**.SCSQLOAD kitaplığında sağlanır.

      Not: COBOL kullanıyorsanız, “IBM MQ sınırlı kod öbeğini dinamik olarak çağırma” sayfa 986başlıklı konuda açıklandığı şekilde dinamik bağlantı oluşturmayı planlamıyorsanız, bağlantı düzenleyicisinin CSQQSTUB ' ye yönelik başvuruları çözmesini sağlamak için NODYNAM derleyici seçeneğini belirleyin.
6. Yükleme modülünü bir uygulama yükleme kitaplığında saklayın.

```
⋮
//*
//* WEBSPHERE MQ FOR Z/OS LIBRARY CONTAINING IMS STUB
//*
//CSQSTUB   DD  DSN=thlqual.SCSQLOAD,DISP=SHR
//*
⋮
//LKED.SYSIN DD *
  INCLUDE CSQSTUB(CSQQSTUB)
  ⋮
/*
```

Şekil 116. IMS ortamında nesne modülünü bağlamak için JCL parçaları

Bir IMS programını çalıştırmadan önce, sistem yöneticiniz programı IMS olarak IBM MQ program ve hareket olarak tanımlamalıdır: Daha sonra bu programı olağan şekilde çalıştırabilirsiniz.

z/OS *z/OS UNIX System Services uygulamaları oluşturma*

z/OS UNIX System Services uygulamaları oluştururken bu bilgileri kullanın.

z/OS UNIX System Servicesaltında çalışan IBM MQ for z/OS için bir C uygulaması oluşturmak üzere uygulamanızı aşağıdaki gibi derleyin ve bağlayın:

```
cc -o mqsamp -W c,DLL -I "//' thlqual.SCSQC370'" mqsamp.c "//' thlqual.SCSQDEFS(CSQBMQ1)'"
```

Burada **thlqual** , kuruluşunuz tarafından kullanılan üst düzey niteleyicidir.

C programını çalıştırmak için, aşağıdakileri `.profile` dosyanıza eklemeniz gerekir; bu, kök dizininizde olmalıdır:

```
STEPLIB= thlqual.SCSQANLE:thlqual.SCSQAUTH: STEPLIB
```

Değişikliğin tanınması için z/OS UNIX System Servicesiçinden çıkmanız ve z/OS UNIX System Services komutunu yeniden girmeniz gerektiğini unutmayın.

Birden çok kabuk çalıştırmak istiyorsanız, satırın başına aşağıdaki gibi bir sözcük dışa aktarma ekleyin:

```
export STEPLIB= thlqual.SCSQANLE:thlqual.SCSQAUTH: STEPLIB
```

Bu işlem başarıyla tamamlandığında CSQBSTUB ve IBM MQ çağrılarını bağlayabilirsiniz.

"IBM MQ sınırlı kod öbeğini dinamik olarak çağırma" sayfa 986 , bir IBM MQ sınırlı kod öbeğine bağlantı düzenlemenize gerek kalmaması için programlarınızda MQI çağrıları yapmanın alternatif bir yöntemini açıklar. Bu yöntem tüm diller ve ortamlar için kullanılamaz.

Programınızın çalıştığı IBM MQ for z/OS sürümünden daha yüksek düzeydeki bir sınırlı kod öbeği programını bağlamayın. Örneğin, IBM WebSphere MQ for z/OS 7.1 üzerinde çalışan bir program, IBM MQ for z/OS 8.0ile verilen bir sınırlı kod öbeği programıyla bağlantı düzenlenmemelidir.

z/OS *IBM MQ sınırlı kod öbeğini dinamik olarak çağırma*

IBM MQ sınırlı kod öbeği programını nesne kodunuzla birlikte düzenlemek yerine, sınırlı kod öbeğini programınızdan dinamik olarak çağırabilirsiniz.

Bunu toplu iş, IMSve CICS ortamlarında yapabilirsiniz. Bu olanak RRS ortamında desteklenmiyor. Uygulama programınız güncellemeleri koordine etmek için RRS kullanıyorsa, bkz. "RRS İle İlgili Önemli Noktalar" sayfa 990.

Ancak, bu yöntem:

- Programlarınızın karmaşıklığını artırır
- Yürütme sırasında programlarınızın gerektirdiği depolamayı artırır
- Programlarınızın performansını azaltır
- Aynı programları diğer ortamlarda kullanamayacağınız anlamına gelir

Sınırlı kod öbeğini devingen olarak çağırırsanız, yürütme sırasında uygun sınırlı kod öbeği programı ve diğer adları kullanılabilir olmalıdır. Bunu sağlamak için IBM MQ for z/OS SCSQLOAD veri kümesini ekleyin:

- Toplu iş ve IMSiçin, JCL ' nin STEPLIB birleşiminde.
- CICSiçin, CICS DFHRPL birleştirmesinde.

IMSiçin, dinamik sınırlı kod öbeğini içeren kitaplığın (IMS bağdaştırıcısının ayarlanması başlıklı konuda IMS bağdaştırıcısının takılmasına ilişkin bilgilerde açıklandığı şekilde oluşturulduğundan emin olun) JCL bölgesinin STEPLIB birleşiminde SCSQLOAD veri kümesinden önce.

Sınırlı kod öbeğini dinamik olarak çağırdığınızda Çizelge 152 sayfa 986 içinde gösterilen adları kullanın. PL/I dilinde, yalnızca programınızda kullanılan çağrı adlarını bildirin.

Çizelge 152. Dinamik bağlantı için arama adları

MQI çağrısı	Toplu (RRS olmayan) dinamik çağrı adları	CICS dinamik arama adları	IMS dinamik arama adları
MQBACK	CSQBBACK	desteklenmiyor	Desteklenmiyor
MQBUFMH	CSQBBFMH	CSQCBFMH [1]	MQBUFMH
MQCB	CSQBCB	CSQCCB [1]	Desteklenmiyor
MQCLOSE	CSQBCLOS	CSQCCLOS	MQCLOSE
MQCMIT	CSQBCOMM	desteklenmiyor	Desteklenmiyor
MQCONN	CSQBCONN	CSQCCONN	MQCONN
MQCONNX	CSQBCONX	CSQCCONX	MQCONNX
MQCRTMH	CSQBCTMH	CSQCCTMH [1]	MQCRTMH
MQCTL	CSQBCTL	CSQCCTL [1]	Desteklenmiyor
MQDISC	CSQBDISC	CSQCDISC	MQDISC
MQDLTMH	CSQBDTMH	CSQCDTMH [1]	MQDLTMH
MQDLTMP	CSQBDTMP	CSQCDTMP [1]	MQDLTMP
MQGet	CSQBGET	CSQCGET	MQGet

Çizelge 152. Dinamik bağlantı için arama adları (devamı var)			
MQI çağrısı	**Toplu (RRS olmayan) dinamik çağrı adları**	**CICS dinamik arama adları**	**IMS dinamik arama adları**
MQINQ	CSQBINQ	CSQCINQ	MQINQ
MQINQMP	CSQBIQMP	CSQCIQMP [1]	MQINQMP
MQMHBUF	CSQBMHBF	CSQCMHBF [1]	MQMHBUF
MQOPEN	CSQBOPEN	CSQCOPEN	MQOPEN
MQPUT	CSQBPUT	CSQCPUT	MQPUT
MQPUT1	CSQBPUT1	CSQCPUT1	MQPUT1
MQSET	CSQBSET	CSQCSET	MQSET
MQSETMP	CSQBSTMP	CSQCSTMP [1]	MQSETMP
MQSTAT	CSQBSTAT	CSQCSTAT [1]	MQSTAT
MQSUB	CSQBSUB	CSQCSUB [1]	MQSUB
MQSUBRQ	CSQBSUBR	CSQCSUBR [1]	MQSUBRQ

Not: 1. Bu API çağrıları yalnızca CICS TS 3.2 ya da üstü kullanıldığında kullanılabilir ve CICS ile birlikte gönderilen CSQCSTUB kullanılmalıdır. CICS TS 3.2için APAR PK66866 uygulanmalıdır. CICS TS 4.1için, APAR PK89844 uygulanmalıdır.

Bu tekniğin nasıl kullanılacağına ilişkin örnekler için aşağıdaki şekillere bakın:

- Toplu iş ve COBOL: bkz. Şekil 117 sayfa 987
- CICS ve COBOL: bkz. Şekil 118 sayfa 988
- IMS ve COBOL: bkz. Şekil 119 sayfa 988
- Toplu iş ve çevirici: bkz. Şekil 120 sayfa 988
- CICS ve çevirici: bkz. Şekil 121 sayfa 988
- IMS ve çevirici: bkz. Şekil 122 sayfa 989
- Toplu İş ve C: Şekil 123 sayfa 989
- CICS ve C: bkz. Şekil 124 sayfa 989
- IMS ve C: bkz. Şekil 125 sayfa 989
- Toplu iş ve PL/I: bkz. Şekil 126 sayfa 989
- IMS ve PL/I: bkz. Şekil 127 sayfa 990

```
...
      WORKING-STORAGE SECTION.
...
         05 WS-MQOPEN                    PIC X(8) VALUE 'CSQBOPEN'.
...
      PROCEDURE DIVISION.
...
          CALL WS-MQOPEN WS-HCONN
                         MQOD
                         WS-OPTIONS
                         WS-HOBJ
                         WS-COMPCODE
                         WS-REASON.
...
```

Şekil 117. Toplu iş ortamında COBOL kullanarak dinamik bağlantı oluşturma

```
...
       WORKING-STORAGE SECTION.
...
          05 WS-MQOPEN                     PIC X(8) VALUE 'CSQCOPEN'.
...
       PROCEDURE DIVISION.
...
           CALL WS-MQOPEN WS-HCONN
                          MQOD
                          WS-OPTIONS
                          WS-HOBJ
                          WS-COMPCODE
                          WS-REASON.
...
```

Şekil 118. CICS ortamında COBOL kullanarak dinamik bağlantı oluşturma

```
...
       WORKING-STORAGE SECTION.
...
          05 WS-MQOPEN                     PIC X(8) VALUE 'MQOPEN'.
...
       PROCEDURE DIVISION.
...
           CALL WS-MQOPEN WS-HCONN
                          MQOD
                          WS-OPTIONS
                          WS-HOBJ
                          WS-COMPCODE
                          WS-REASON.
...
      * ------------------------------------------------------------ *
      *
      *    If the compilation option 'DYNAM' is specified
      *    then you may code the MQ calls as follows
      *
      * ------------------------------------------------------------ *
...
           CALL 'MQOPEN'  WS-HCONN
                          MQOD
                          WS-OPTIONS
                          WS-HOBJ
                          WS-COMPCODE
                          WS-REASON.
...
```

Şekil 119. IMS ortamında COBOL kullanarak dinamik bağlantı oluşturma

```
...
         LOAD   EP=CSQBOPEN
...
         CALL (15),(HCONN,MQOD,OPTIONS,HOBJ,COMPCODE,REASON),VL
...
         DELETE EP=CSQBOPEN
...
```

Şekil 120. Toplu iş ortamında birleştirme dili kullanılarak dinamik bağlantı oluşturma

```
...
         EXEC CICS LOAD PROGRAM('CSQCOPEN') ENTRY(R15)
...
         CALL (15),(HCONN,MQOD,OPTIONS,HOBJ,COMPCODE,REASON),VL
...
         EXEC CICS RELEASE PROGRAM('CSQCOPEN')
...
```

Şekil 121. CICS ortamında birleştirme dili kullanılarak dinamik bağlantı oluşturma

```
...
         LOAD   EP=MQOPEN
...
         CALL (15),(HCONN,MQOD,OPTIONS,HOBJ,COMPCODE,REASON),VL
...
         DELETE EP=MQOPEN
...
```

Şekil 122. IMS ortamında birleştirme dili kullanılarak dinamik bağlantı oluşturma

```
...
typedef void CALL_ME();
#pragma linkage(CALL_ME, OS)
...
main()
{
CALL_ME * csqbopen;
...
csqbopen = (CALL_ME *) fetch("CSQBOPEN");
(*csqbopen)(Hconn,&ObjDesc,Options,&Hobj,&CompCode,&Reason);
...
```

Şekil 123. Toplu iş ortamında C dili kullanılarak dinamik bağlantı oluşturma

```
...
typedef void CALL_ME();
#pragma linkage(CALL_ME, OS)
...
main()
{
CALL_ME * csqcopen;
...
  EXEC CICS LOAD PROGRAM("CSQCOPEN") ENTRY(csqcopen);
(*csqcopen)(Hconn,&ObjDesc,Options,&Hobj,&CompCode,&Reason);
...
```

Şekil 124. CICS ortamında C dili kullanılarak dinamik bağlantı oluşturma

```
...
typedef void CALL_ME();
#pragma linkage(CALL_ME, OS)
...
main()
{
CALL_ME * mqopen;
...
mqopen = (CALL_ME *) fetch("MQOPEN");
(*mqopen)(Hconn,&ObjDesc,Options,&Hobj,&CompCode,&Reason);
...
```

Şekil 125. IMS ortamında C dili kullanılarak dinamik bağlantı oluşturma

```
...
      DCL CSQBOPEN ENTRY EXT OPTIONS(ASSEMBLER INTER);
...
      FETCH CSQBOPEN;

      CALL CSQBOPEN(HQM,
                    MQOD,
                    OPTIONS,
                    HOBJ,
                    COMPCODE,
                    REASON);

      RELEASE CSQBOPEN;
```

Şekil 126. Toplu iş ortamında PL/I kullanılarak dinamik bağlantı oluşturma

```
...
      DCL MQOPEN   ENTRY EXT OPTIONS(ASSEMBLER INTER);
...
      FETCH MQOPEN;

      CALL   MQOPEN(HQM,
                    MQOD,
                    OPTIONS,
                    HOBJ,
                    COMPCODE,
                    REASON);

      RELEASE   MQOPEN;
```

Şekil 127. IMS ortamında PL/I kullanılarak dinamik bağlantı oluşturma

z/OS *RRS İle İlgili Önemli Noktalar*

Uygulama programınız güncellemeleri koordine etmek için RRS kullanıyorsa, bu bilgileri kullanmayı göz önünde bulundurun.

IBM MQ , RRS eşgüdümü gereken toplu programlar için iki farklı sınırlı kod öbeği sağlar-bkz. “RRS toplu iş bağdaştırıcısı” sayfa 854. Sonraki API çağrılarının davranışındaki fark, toplu iş bağdaştırıcısı tarafından MQCONN ya da MQCONNX API üzerindeki sınırlı kod öbeği yordamı tarafından geçirilen bilgilerle MQCONN zamanında saptanır. Bu, IBM MQ ile ilk bağlantının uygun sınırlı kod öbeği kullanılarak yapılması koşuluyla, RRS eşgüdümü gereken toplu iş programları için dinamik API çağrıları kullanılabileceği anlamına gelir. Aşağıdaki örnek bunu göstermektedir:

```
       WORKING-STORAGE SECTION.
           05 WS-MQOPEN       PIC X(8) VALUE 'MQOPEN' .
.
.
.
       PROCEDURE DIVISION.
.
.
.
       *
       * Static call to MQCONN must be resolved by linkage edit to
       * CSQBRSTB or CSQBRRSI for RRS coordination
       *
           CALL 'MQCONN' USING W00-QMGR
                               W03-HCONN
                               W03-COMPCODE
                               W03-REASON.
.
.
.
       *
           CALL WS-MQOPEN  WS-HCONN
                           MQOD
                           WS-OPTIONS
                           WS-HOBJ
                           WS-COMPCODE
                           WS-REASON.
```

z/OS *Programlarınızda hata ayıklama*

TSO ve CICS programlarında hata ayıklanması ve CICS izlemesine ilişkin bir öngörü hakkında bilgi edinmek için bu bilgileri kullanın.

IBM MQ for z/OS uygulama programlarında hata ayıklamaya ilişkin ana yardım, her API çağrısı tarafından döndürülen neden kodlarıdır. Düzeltici eylem fikirleri de içinde olmak üzere bunların bir listesi için bkz:

- IBM MQ for z/OS iletileri, tamamlama ve neden kodları - IBM MQ for z/OS
- Diğer tüm IBM MQ platformları için İletiler ve neden kodları

Bu konu, belirli ortamlarda kullanılacak diğer hata ayıklama araçlarını da önerir.

TSO programlarında hata ayıklanması

TSO programları için aşağıdaki etkileşimli hata ayıklama araçları kullanılabilir:

- TEST aracı
- VS COBOL II etkileşimli hata ayıklama aracı
- C ve PL/I programları için INSPECT etkileşimli hata ayıklama aracı

CICS programlarında hata ayıklama

Program ya da program hazırlama yordamını değiştirmek zorunda kalmadan CICS programlarınızı etkileşimli olarak sınamak için CICS Execution Diagnostic Facility (CEDF) olanağını kullanabilirsiniz.

EDF hakkında daha fazla bilgi için bkz. *CICS Transaction Server for z/OS CICS Application Programming Guide*.

CICS İz

Büyük olasılıkla, CICS izleme etkinliğini denetlemek için CICS İzleme Denetimi hareketini (CETR) kullanmanız da yararlı olacaktır.

CETR hakkında daha fazla bilgi için bkz. *CICS Transaction Server for z/OS CICS-Supplied Transactions* elkitabı.

CICS izlemesinin etkin olup olmadığını belirlemek için CKQC panosunu kullanarak bağlantı durumunu görüntüleyin. Bu pano, izleme numarasını da gösterir.

CICS izleme girişlerini yorumlamak için bkz. Çizelge 153 sayfa 991.

Bu değerlere ilişkin CICS izleme girişi AP0 *xxx* ' dir (burada *xxx* , CICS bağdaştırıcısı etkinleştirildiğinde belirlenen izleme numarasıdır). CSQCTEST dışındaki tüm izleme girişleri CSQCTRUE tarafından yayınlanır. CSQCTEST, CSQCRST ve CSQCDSP tarafından verilir.

Çizelge 153. CICS bağdaştırıcı izleme girişleri

Ad	Açıklama	İzleme sırası	İzleme verileri
CSQCABNT	Olağandışı sonlandırma	IBM MQiçin END_THREAD ABNORMAL komutu vermeden önce. Bunun nedeni, görevin sona ermesi ve uygulama tarafından örtük bir geriletme gerçekleştirilebilir. Bu durumda, END_THREAD çağrısında bir ROLLBACK isteği bulunur.	İş birimi bilgileri. İşin durumu hakkında bilgi edinirken bu bilgileri kullanabilirsiniz. (Örneğin, bu, DISPLAY THREAD komutu ya da IBM MQ for z/OS günlük yazdırma yardımcı programı tarafından üretilen çıkışa göre doğrulanabilir.)
GERI DöNüş	Eşitleme noktası geriletme	IBM MQ for z/OSiçin BACKOUT yayınlamadan önce. Bunun nedeni, uygulamadan gelen belirtik bir geriletme isteğidir.	İş birimi bilgileri.
CSQCCCRC	Tamamlanma kodu ve neden kodu	API çağrısından başarısız döndükten sonra.	Tamamlanma kodu ve neden kodu.

Çizelge 153. CICS bağdaştırıcı izleme girişleri (devamı var)

Ad	Açıklama	İzleme sırası	İzleme verileri
CSQCCOMM	Eşitleme noktası kesinleştirme	IBM MQ for z/OSiçin COMMIT çalıştırılmadan önce. Bunun nedeni, tek aşamalı bir kesinleştirme isteği ya da iki aşamalı bir kesinleştirme isteğinin ikinci aşaması olabilir. İstek, uygulamadan gelen belirtik bir eşitleme noktası isteğinden kaynaklanıyor.	İş birimi bilgileri.
CSQCEXER	Yürütme çözümlemesi	EXECUTE_RESOLVE IBM MQ for z/OSolarak yayınlanmadan önce.	EXECUTE_RESOLVE komutunu yayınlayan iş biriminin iş birimi bilgileri. Bu, yeniden eşzamanlama sürecindeki son belirsiz iş birimidir.
CSQCGETW	GET WAIT	CICS komutunu vermeden önce bekleyin.	Beklenecek ECB ' nin adresi.
CSQCGMGD	GET ileti verileri	MQGET başarılı bir şekilde geri döndükten sonra.	İleti verilerinin 40 bayta kadar.
CSQCGMGH	GET ileti tanıtıcısı	IBM MQ for z/OSiçin MQGET çalıştırılmadan önce.	Nesne tanıtıcısı.
CSQCGMGI	İleti tanıtıcısını al	MQGET başarılı bir şekilde geri döndükten sonra.	İletinin ileti tanıtıcısı ve ilinti tanıtıcısı.
CSQCINDL	Belirsiz liste	İkinci INQUIRE_INDOUBT 'tan başarıyla döndükten sonra.	İş listesinin belirsiz birimleri.
CSQCINDO	Yalnızca IBM kullan		
CSQCINDS	Belirsiz liste boyutu	İlk INQUIRE_INDOUBT 'tan başarıyla döndükten sonra belirsiz liste boş değil.	Listenin uzunluğu. 64 'e bölünmesi, belirsiz iş birimi sayısını verir.
CSQCINQH	INQ tanıtıcısı	IBM MQ for z/OSiçin MQINQ yayınlamadan önce.	Nesne tanıtıcısı.
CSQCLOSH	CLOSE tutamacı	MQCLOSE komutunu IBM MQ for z/OS' e vermeden önce.	Nesne tanıtıcısı.
CSQCLOST	Yok etme kayboldu	Yeniden eşzamanlama işlemi sırasında CICS , bağdaştırıcıya yeniden başlatıldığını bildirir; bu nedenle yeniden eşzamanlanmakta olan iş birimiyle ilgili atma bilgisi yoktur.	Yeniden eşzamanlanmakta olan iş birimine ilişkin CICS tarafından bilinen iş birimi tanıtıcısı.
CSQCNIND	Yok etme belirsiz değil	Yeniden eşzamanlama işlemi sırasında CICS , bağdaştırıcıya yeniden eşzamanlanmakta olan iş biriminin belirsiz olmaması gerektiğini bildirir (yani, hala çalışıyor olabilir).	Yeniden eşzamanlanmakta olan iş birimine ilişkin CICS tarafından bilinen iş birimi tanıtıcısı.

Çizelge 153. CICS bağdaştırıcı izleme girişleri (devamı var)

Ad	Açıklama	İzleme sırası	İzleme verileri
CSQCNORT	Normal sonlandırma	IBM MQ for z/OSiçin END_THREAD NORMAL komutu vermeden önce. Bu, görevin sona ermesinden kaynaklanır ve bu nedenle uygulama örtük bir uyumlulaştırma noktası kesinleştirme gerçekleştirebilir. Bu durumda, END_THREAD çağrısında bir COMMIT isteği bulunur.	İş birimi bilgileri.
CSQCOPNH	OPEN tutamacı	MQOPEN ' den başarıyla döndükten sonra.	Nesne tanıtıcısı.
CSQCOPNO	Nesneyi aç	IBM MQ for z/OSiçin MQOPEN yayınlamadan önce.	Nesne adı.
CSQCPMGD	PUT ileti verileri	MQPUT IBM MQ for z/OS' e verilmeden önce.	İleti verilerinin 40 bayta kadar.
CSQCPMGH	PUT iletisi tanıtıcısı	MQPUT IBM MQ for z/OS' e verilmeden önce.	Nesne tanıtıcısı.
CSQCPMGI	PUT iletisi tanıtıcısı	IBM MQ for z/OSiçinden başarılı bir MQPUT sonrasında.	İletinin ileti tanıtıcısı ve ilinti tanıtıcısı.
CSQCPREP	Eşitleme noktası hazırlığı	İki aşamalı kesinleştirme işleminin birinci aşamasında IBM MQ for z/OS için PREPARE komutunu vermeden önce. Bu çağrı, dağıtılmış kuyruğa alma bileşeninden bir API çağrısı olarak da verilebilir.	İş birimi bilgileri.
CSQCP1MD	PUTONE ileti verileri	IBM MQ for z/OSiçin MQPUT1 çalıştırılmadan önce.	İletinin 40 bayta kadar verisi.
CSQCP1MI	PUTONE ileti tanıtıcısı	MQPUT1' den başarıyla döndükten sonra.	İletinin ileti tanıtıcısı ve ilinti tanıtıcısı.
CSQCP1ON	PUTONE nesne adı	IBM MQ for z/OSiçin MQPUT1 çalıştırılmadan önce.	Nesne adı.
CSQCRBAK	Çözümlenen geriletme	RESOLVE_ROLLBACK komutunu IBM MQ for z/OS' e vermeden önce.	İş birimi bilgileri.
CSQCRCMT	Çözülmüş kesinleştirme	IBM MQ for z/ OSiçin RESOLVE_COMMIT yayınlamadan önce.	İş birimi bilgileri.
CSQCRMIR	RMI yanıtı	Belirli bir çağrıdan CICS RMI ' ya (kaynak yöneticisi arabirimi) dönmeden önce.	Mimari RMI yanıt değeri. Anlamı, çağırma tipine bağlıdır. Bu değerler, *CICS Transaction Server for z/OS Customization Guide*belgelenmiştir. Çağırma tipini belirlemek için CICS RMI bileşeni tarafından üretilen önceki izleme girişlerine bakın.

Çizelge 153. CICS bağdaştırıcı izleme girişleri (devamı var)			
Ad	**Açıklama**	**İzleme sırası**	**İzleme verileri**
CSQCRSYN	Yeniden eşzamanlama	Görev için yeniden eşzamanlama işlemi başlamadan önce.	Yeniden eşzamanlanmakta olan iş birimine ilişkin CICS tarafından bilinen iş birimi tanıtıcısı.
CSQCSETH	SET tanıtıcısı	MQSET, IBM MQ for z/OSolarak yayınlanmadan önce.	Nesne tanıtıcısı.
CSQCTASE	Yalnızca IBM kullan		
CSQCTEST	İzleme sınaması	Kullanıcı tarafından sağlanan izleme numarasını ya da bağlantının izleme durumunu doğrulamak için EXEC CICS ENTER TRACE çağrısında kullanılır.	Veri yok.
CSQCDCFF	Yalnızca IBM kullan		

Yordamsal program hatalarının işlenmesi

Bu bilgiler, bir çağrı yaptığında ya da iletisi son hedefine teslim edildiğinde, uygulamalarınızla ilişkili MQI çağrılarıyla ilgili hataları açıklar.

Mümkün olduğunda, kuyruk yöneticisi bir MQI çağrısı yapılır yapılmaz hata döndürür. Bunlar *yerel olarak saptanan hatalardır*.

Uzak kuyruğa ileti gönderilirken, MQI çağrısı yapıldığında hatalar görünmeyebilir. Bu durumda, hataları tanımlayan kuyruk yöneticisi bunları kaynak programa başka bir ileti göndererek bildirir. Bunlar *uzaktan belirlenen hatalardır*.

Yerel olarak belirlenen hatalar

Yerel olarak saptanan hatalarla ilgili bilgi: MQI çağrısında hata, sistem kesintileri ve yanlış veri içeren iletiler.

Kuyruk yöneticisinin hemen bildirebileceği hataların en sık rastlanan üç nedeni şunlardır:

- MQI çağrısı başarısız oldu; örneğin, bir kuyruk dolu olduğundan
- Uygulamanızın bağlı olduğu sistemin bir kısmının çalışmasının kesilmesi; örneğin, kuyruk yöneticisi
- Başarıyla işlenemeyen verileri içeren iletiler

Zamanuyumsuz koyma olanağını kullanıyorsanız, hatalar hemen bildirilmez. Önceki zamanuyumsuz koyma işlemlerine ilişkin durum bilgilerini almak için MQSTAT çağrısına bakın.

MQI çağrısı başarısız oldu

Kuyruk yöneticisi, bir MQI çağrısının kodlamasındaki hataları hemen bildirebilir. Bunu, önceden tanımlanmış dönüş kodları kümesini kullanarak yapar. Bunlar, tamamlama kodlarına ve neden kodlarına ayrılmıştır.

Bir aramanın başarılı olup olmadığını göstermek için, çağrı tamamlandığında kuyruk yöneticisi bir *tamamlanma kodu* döndürür. Başarı, kısmi tamamlanma ve aramanın başarısız olduğunu gösteren üç tamamlama kodu vardır. Kuyruk yöneticisi, kısmi tamamlanma nedenini ya da çağrı hatasını belirten bir *neden kodu* da döndürür.

Her aramaya ilişkin tamamlanma ve neden kodları, Dönüş kodları' ndaki aramanın açıklamasıyla birlikte listelenir. Düzeltici eylem fikirleri de içinde olmak üzere daha ayrıntılı bilgi için bkz:

- z/OS IBM MQ for z/OS iletileri, tamamlama ve neden kodları - IBM MQ for z/OS

- Diğer tüm IBM MQ platformları için İletiler ve neden kodları

Programlarınızı, her aramadan ortaya çıkabilecek tüm dönüş kodlarını işleyecek şekilde tasarlayın.

System interruptions

Bağlı olduğu kuyruk yöneticisinin bir sistem hatasından kurtulması gerekirse, uygulamanız herhangi bir kesintiden haberdar olmayabilir. Ancak, böyle bir kesinti oluşursa verilerinizin kaybolmadığından emin olmak için uygulamanızı tasarlamanız gerekir.

Verilerinizin tutarlılığını sağlamak için kullanabileceğiniz yöntemler, kuyruk yöneticinizin çalıştığı altyapıya bağlıdır:

z/OS
CICS ve IMS ortamlarında, CICS ya da IMStarafından yönetilen iş birimleri içinde MQPUT ve MQGET çağrıları yapabilirsiniz. Toplu iş ortamında MQPUT ve MQGET çağrılarını aynı şekilde yapabilirsiniz, ancak eşitleme noktalarını aşağıdakileri kullanarak bildirmeniz gerekir:

- IBM MQ for z/OS MQCMIT ve MQBACK çağrıları (bkz. "İş birimlerinin kesinleştirilmesi ve geri çekilmesi" sayfa 818) ya da
- İki aşamalı eşitleme noktası desteği sağlamak için z/OS Hareket Yönetimi ve Kurtarılabilir Resource Manager Hizmetleri (RRS). RRS, tek bir mantıksal iş birimi içinde hem IBM MQ hem de Db2 saklanmış yordam kaynakları gibi RRS etkin ürün kaynaklarını güncellemenizi sağlar. RRS eşitleme noktası desteği hakkında bilgi için bkz. "Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri" sayfa 822.

IBM i
IBM i kesinleştirme denetimi tarafından yönetilen genel iş birimlerinde MQPUT ve MQGET çağrılarınızı yapabilirsiniz. Yerel IBM i COMMIT ve ROLLBACK komutlarını ya da dile özgü komutları kullanarak eşitleme noktalarını bildirebilirsiniz. Yerel iş birimleri, MQCMIT ve MQBACK çağrıları kullanılarak IBM MQ tarafından yönetilir.

AIX, Linux, and Windows sistemleri
Bu ortamlarda MQPUT ve MQGET çağrılarınızı olağan şekilde yapabilirsiniz, ancak MQCMIT ve MQBACK çağrılarını kullanarak eşitleme noktalarını bildirmeniz gerekir (bkz. "İş birimlerinin kesinleştirilmesi ve geri çekilmesi" sayfa 818). CICS ortamında MQCMIT ve MQBACK komutları devre dışı bırakılır; MQPUT ve MQGET çağrılarınızı CICStarafından yönetilen iş birimleri içinde yapabilirsiniz.

Kaybetmeyi göze alamayacağınız tüm verileri taşımak için kalıcı iletiler kullanın. Kuyruk yöneticisinin bir hatadan kurtulması gerekirse, kuyruklardaki kalıcı iletiler geri getirilir. ALW IBM MQ on AIX, Linux, and Windowsile, uygulamanız içindeki bir MQGET ya da MQPUT çağrısı, tüm günlük dosyalarını doldurma noktasında MQRC_RESOURCE_PROBLEM iletisiyle başarısız olur. AIX, Linux, and Windowsüzerindeki günlük dosyalarıyla ilgili daha fazla bilgi için bkz. IBM MQ Yönetimi. z/OS z/OS için bkz. z/OS üzerinde planlama.

Bir uygulama çalışırken kuyruk yöneticisi bir işleç tarafından durdurulursa, genellikle susturma seçeneği kullanılır. Kuyruk yöneticisi, uygulamaların çalışmaya devam edebileceği bir susturma durumuna girer, ancak uygun olduğu anda sonlandırmaları gerekir. Küçük, hızlı uygulamalar büyük olasılıkla susturma durumunu yoksayabilir ve normal olarak sonlandırılıncaya kadar devam edebilir. Daha uzun süre çalışan uygulamalar ya da iletilerin gelmesini bekleyenler, MQOPEN, MQPUT, MQPUT1ve MQGET çağrılarını kullandıklarında *susturulursa başarısız olur* seçeneğini kullanmalıdır. Bu seçenekler, kuyruk yöneticisi susturulduğunda çağrıların başarısız olduğu, ancak uygulamanın susturma durumunu yoksayan çağrılar vererek düzgün bir şekilde sonlandırmak için hala zamanı olabileceği anlamına gelir. Bu tür uygulamalar, yaptıkları değişiklikleri kesinleştirebilir ya da geri çekilebilir ve daha sonra sonlandırabilir.

Kuyruk yöneticisi durdurulmaya zorlandıysa (yani, susturulmadan durdurulursa), uygulamalar MQI çağrıları yaptıklarında MQRC_CONNECTION_BROKEN neden kodunu alır. Uygulamadan çıkın ya da diğer bir seçenek olarak, IBM i IBM MQ for IBM i, AIX, Linux, and Windows sistemlerinde bir MQDISC çağrısı yayınlayın.

Yanlış veri içeren iletiler

Uygulamanızda iş birimlerini kullandığınızda, bir program kuyruktan aldığı bir iletiyi başarıyla işleyemezse, MQGET çağrısı geriletir.

Kuyruk yöneticisi, oluşma sayısını (ileti tanımlayıcısının *BackoutCount* alanında) tutar. Etkilenen her iletinin tanımlayıcısında bu sayıyı korur. Bu sayı, bir uygulamanın verimliliğine ilişkin değerli bilgiler sağlayabilir. Zaman içinde artan geri dönüş sayıları olan iletiler sürekli olarak reddedilir; uygulamanızı bunun nedenlerini analiz etmek ve bu tür iletileri buna göre işlemesi için tasarlayın.

z/OS IBM MQ for z/OS' ta, gerileme sayısının kuyruk yöneticisini yeniden başlatmaya devam etmesi için **HardenGetBackout** özniteliğini MQQA_BACKOUT_HARDENED olarak ayarlayın; tersi durumda, kuyruk yöneticisinin yeniden başlatılması gerekirse, her ileti için doğru bir geriletme sayısı korunmaz. Özniteliğin bu şekilde ayarlanması, fazladan işlemenin cezasını ekler.

IBM MQ for IBM i IBM i, AIX, Linux, and Windows sistemlerinde, geriletme sayısı her zaman kuyruk yöneticisini yeniden başlatmaya devam eder.

z/OS Ayrıca, IBM MQ for z/OSişletim birimindeki bir iş birimindeki bir kuyruktan iletileri kaldırdığınızda, bir iletiyi, iş birimi uygulama tarafından yedeklenirse bir daha kullanılabilir kılınmayacak şekilde işaretleyebilirsiniz. İşaretli ileti, yeni bir iş birimi altında alınmış gibi işlenir. MQGMO_MARK_SKIP_BACKOUT seçeneğini kullanarak geriletme işlemini atlanacak iletiyi işaretleyebilirsiniz(MQGMO yapısında) MQGET çağrıyı kullandığınızda. Bu teknik hakkında daha fazla bilgi için bkz. “Geriletme atlanıyor” sayfa 766 .

Sorun belirleme için rapor iletilerinin kullanılması

Uzak kuyruk yöneticisi, MQI çağrıyı yaptığınızda kuyruğa ileti konmaması gibi hataları bildiremez, ancak iletinizi nasıl işlediğini belirten bir rapor iletisi gönderebilir.

Uygulamanızda rapor iletileri yaratabilir (MQPUT) ve bunları alma seçeneğini belirleyebilirsiniz (bu durumda bunlar başka bir uygulama ya da kuyruk yöneticisi tarafından gönderilir).

Rapor iletileri oluşturma

Rapor iletileri, bir uygulamanın başka bir uygulamaya gönderilen iletiyle başa çıkamayacağını söylemesini sağlar.

Ancak, iletiyi gönderen uygulamanın herhangi bir sorunla ilgili olarak bilgilendirilmekle ilgilenip ilgilenmediğini belirlemek için başlangıçta *Report* alanının çözümlenmesi gerekir. Bir rapor iletisinin gerekli olduğunu belirledikten sonra, aşağıdakilere karar vermeniz gerekir:

- Özgün iletinin tamamının, yalnızca ilk 100 baytlık verinin ya da özgün iletinin hiçbirinin dahil edilmemesini isteyip istemediğinizi belirler.
- Özgün iletiyle ne yapilacak? Bunu atabilir ya da gitmeyen iletiler kuyruğuna gitmesine izin verebilirsiniz.
- *MsgId* ve *CorrelId* alanlarının içeriğinin de gerekli olup olmadığını belirler.

Oluşturulmakta olan rapor iletisinin nedenini belirtmek için *Feedback* alanını kullanın. Rapor iletilerinizi bir uygulamanın yanıt kuyruğuna koyun. Daha fazla bilgi için bkz. Geribildirim .

Rapor iletilerinin istenmesi ve alınması (MQGET)

Başka bir uygulamaya ileti gönderdiğinizde, gereksinim duyduğunuz geribildirimi belirtmek için *Report* alanını doldurmadığınız sürece herhangi bir sorun hakkında bilgilendirilmezsiniz. Kullanılabilir seçenekler için bkz. Rapor alanının yapısı .

Kuyruk yöneticileri her zaman bir uygulamanın yanıt kuyruğuna rapor iletileri koyar ve kendi uygulamalarınızın da aynısını yapması önerilir. Rapor iletisi olanağını kullandığınızda, iletinizin ileti tanımlayıcısında yanıt kuyruğunun adını belirtin; tersi durumda MQPUT çağrısı başarısız olur.

Uygulamanız, yanıt kuyruğunuzu izleyen yordamları içermelidir ve üzerine gelen iletileri işlemelidir. Bir rapor iletisinin tüm özgün iletiyi, özgün iletinin ilk 100 baytını ya da özgün iletinin hiçbirini içermediğini unutmayın.

Kuyruk yöneticisi, hatanın nedenini belirtmek için rapor iletisinin *Feedback* alanını ayarlar; örneğin, hedef kuyruk yok. Senin programların da aynısını yapmalı.

Rapor iletileri hakkında daha fazla bilgi için bkz. “Rapor iletileri” sayfa 19.

Uzaktan belirlenen hatalar

Uzak bir kuyruğa ileti gönderdiğinizde, yerel kuyruk yöneticisi MQI çağrılarınızı hata bulmadan işlediğinde bile, başka etkenler iletinizin uzak bir kuyruk yöneticisi tarafından nasıl işlendiğini etkileyebilir.

Örneğin, hedeflemekte olduğunuz kuyruk dolu olabilir ya da var olmayabilir. İletinizin hedef kuyruğa giden rotadaki diğer ara kuyruk yöneticileri tarafından işlenmesi gerekirse, bunlardan herhangi biri bir hata bulabilir.

İleti teslim edildiğinde birden çok sorun ortaya çıktı

Bir MQPUT çağrısı başarısız olduğunda, iletiyi kuyruğa yeniden yerleştirmeyi deneyebilir, gönderene geri gönderebilir ya da gönderilemeyen iletiler kuyruğuna koyabilirsiniz.

Her seçeneğin kendi erdemleri vardır, ancak MQPUT ' nin başarısız olmasının nedeni hedef kuyruğun dolu olması ise, bir iletiyi yeniden yerleştirmeyi denemek istemeyebilirsiniz. Bu örnekte, bu kuyruğun teslim edilmeyen iletiler kuyruğuna konması, kuyruğun daha sonra doğru hedef kuyruğuna teslim edilmesine olanak sağlar.

İleti tesliminin yeniden denenmesi

İleti gitmeyen iletiler kuyruğuna konmadan önce, kanal için *MsgRetryCount* ve *MsgRetryInterval* öznitelikleri ayarlandıysa ya da kanal için yeniden deneme çıkış programı varsa (adı *MsgRetryExitId* kanal özniteliği alanında tutulan), uzak kuyruk yöneticisi iletiyi kuyruğa yeniden yerleştirmeyi dener.

MsgRetryExitId alanı boşsa, *MsgRetryCount* ve *MsgRetryInterval* özniteliklerindeki değerler kullanılır.

MsgRetryExitId alanı boş değilse, bu adın çıkış programı çalışır. Kendi çıkış programlarınızı kullanma hakkında daha fazla bilgi için bkz. “İleti sistemi kanalları için kanal çıkış programları” sayfa 920.

İletiyi gönderene döndür

Özgün iletinin tümünü içerecek bir rapor iletisinin oluşturulmasını isteyerek gönderene bir ileti döndürürsiniz.

Rapor iletisi seçenekleriyle ilgili ayrıntılar için bkz. “Rapor iletileri” sayfa 19 .

Teslim edilmeyen ileti kuyruğunun kullanılması

Bir kuyruk yöneticisi bir iletiyi teslim edemediği zaman, iletiyi teslim edilmeyen iletiler kuyruğuna yerleştirmeyi dener. Kuyruk yöneticisi kurulduğunda bu kuyruk tanımlanmalıdır.

Programlarınız, kuyruk yöneticisinin kullandığı gibi, teslim mektubu kuyruğunu da kullanabilir. Kuyruk yöneticisi nesnesini (MQOPEN çağrısıyla) açarak ve **DeadLetterQName** özniteliği (MQINQ çağrısıyla) hakkında bilgi alarak, teslim edilmeyen ileti kuyruğunun adını bulabilirsiniz.

Kuyruk yöneticisi bu kuyruğa bir ileti koyduğunda, iletiye biçimi MQDLH (ölü harf üstbilgisi) yapısı tarafından tanımlanan bir üstbilgi ekler; bkz. MQDLH-Ölü harf üstbilgisi. Bu üstbilgi, hedef kuyruğun adını ve iletinin gönderilmeyen iletiler kuyruğuna konma nedenini içerir. İleti istenen kuyruğa konmadan önce kaldırılmalı ve sorun çözülmelidir. Kuyruk yöneticisi, iletinin bir MQDLH yapısı içerdiğini belirtmek için ileti tanımlayıcısının (MQMD) *Format* alanını da değiştirir.

MQDLH yapısı

Teslim edilmeyen iletiler kuyruğuna yerleştirdiğiniz tüm iletilere bir MQDLH yapısı eklemeniz önerilir; ancak, belirli IBM MQ ürünleri tarafından sağlanan teslim edilmeyen mektup işleyicisini kullanmayı planlıyorsanız, iletilerinize bir MQDLH yapısı eklemeniz gerekir.

Üstbilginin bir iletiye eklenmesi iletiyi teslim edilmeyen ileti kuyruğu için çok uzun yapabilir; bu nedenle, iletilerinizin en azından MQ_MSG_HEADER_LENGTH değişmezinin değerine göre, teslim edilmeyen ileti kuyruğu için izin verilen boyut üst sınırından daha kısa olduğundan emin olun. Bir kuyrukta izin verilen ileti büyüklüğü üst sınırı, kuyruğun **MaxMsgLength** özniteliğinin değerine göre belirlenir. Teslim edilmeyen ileti kuyruğu için, bu özniteliğin kuyruk yöneticisi tarafından izin verilen üst sınıra ayarlandığından emin olun. Uygulamanız bir iletiyi teslim edemezse ve ileti, gitmeyen iletiler kuyruğuna konamayacak kadar uzunsa, MQDLH yapısının tanımında belirtilen öneriyi izleyin.

Gitmeyen iletiler kuyruğunun izlendiğinden ve gelen iletilerin işlendiğinden emin olun. Gelmeyen iletiler kuyruk işleyicisi toplu iş yardımcı programı olarak çalışır ve teslim edilmeyen iletiler kuyruğunda seçilen iletiler üzerinde çeşitli işlemler gerçekleştirmek için kullanılabilir. Daha fazla ayrıntı için bkz. “Teslim mektubu kuyruğunun işlenmesi” sayfa 998.

Veri dönüştürme gerekiyorsa, MQGET çağrısında MQGMO_CONVERT seçeneğini kullandığınızda kuyruk yöneticisi üstbilgiyi dönüştürür. İletiyi koyan işlem bir MCA ise, üstbilgiyi özgün iletinin tüm metni izler.

Bu kuyruk için çok uzunsa, ileti kuyruğuna konan iletiler kesilebilir. Bu durumun olası bir göstergesi, teslim edilmeyen ileti kuyruğundaki iletilerin, kuyruğun **MaxMsgLength** özniteliğinin değeriyle aynı uzunlukta olması olabilir.

Teslim mektubu kuyruğunun işlenmesi
Bu bilgiler, ileti kuyruğunun işlenmesini kullanırken genel kullanım programlama arabirimi bilgilerini içerir.

Gönderilmeyen iletiler kuyruğunun işlenmesi yerel sistem gereksinimlerine bağlıdır, ancak belirtimi çizerken aşağıdaki şeyleri göz önünde bulundurun:

- MQMD ' deki biçim alanının değeri MQFMT_DEAD_LETTER_HEADER olduğundan, ileti bir teslim edilmeyen ileti kuyruğu üstbilgisine sahip olarak tanımlanabilir.
- IBM MQ for z/OS kullanılıyorsa CICS, bir MCA bu iletiyi teslim edilmeyen ileti kuyruğuna koyarsa, *PutApplType* alanı MQAT_CICS olur ve *PutApplName* alanı CICS sisteminin *ApplId* alanıdır ve ardından MCA ' nın hareket adı gelir.
- İletinin teslim edilmeyen ileti kuyruğuna yöneltilmesinin nedeni, teslim edilmeyen ileti kuyruğu üstbilgisinin *Reason* alanında bulunur.
- Gönderilmeyen ileti kuyruğu üstbilgisi, hedef kuyruk adı ve kuyruk yöneticisi adının ayrıntılarını içerir.
- Teslim edilmeyen ileti kuyruğu üstbilgisi, ileti hedef kuyruğa yerleştirilmeden önce ileti tanımlayıcısında yeniden belirtilmesi gereken alanları içerir. Bunlar:
  1. *Encoding*
  2. *CodedCharSetId*
  3. *Format*
- Gösterilen üç alan (Kodlama, CodedCharSetIdve Biçim) dışında, ileti tanımlayıcısı özgün uygulama tarafından PUT ile aynıdır.

Gönderilmeyen iletiler için kuyruk uygulamanızın aşağıdakilerden birini ya da birkaçını yapması gerekir:

- *Reason* alanını inceleyin. Aşağıdaki nedenlerden ötürü MCA tarafından bir ileti konmuş olabilir:
  - İleti, kanala ilişkin ileti büyüklüğü üst sınırından daha uzun

    Nedeni: MQRC_MSG_TOO_BIG_FOR_CHANNEL
  - İleti hedef kuyruğuna konamadı

    Bunun nedeni, MQPUT işlemi tarafından döndürülecek herhangi bir MQRC_ * neden kodudur.
  - Bir kullanıcı çıkışı bu işlemi istedi

Neden kodu, kullanıcı çıkışı tarafından sağlanan ya da varsayılan MQRC_SUPPRESSED_BY_EXIT

- İletiyi, mümkün olduğu yerde, hedeflenen hedefine iletmeyi deneyin.
- Sapma nedeni saptandığında, ancak hemen düzeltilemeyecek şekilde, iletiyi atmadan önce belirli bir süre boyunca alıkoyun.
- Sistem yöneticilerine, bunların belirlendiği yerde sorunları düzeltmek için yönergeler verin.
- Bozulmuş ya da işlenebilir olmayan iletileri atın.

Gitmeyen iletiler kuyruğundan kurtardığınız iletilerle başa çıkmanın iki yolu vardır:

1. İleti yerel bir kuyruk içinse:
   - Uygulama verilerini çıkarmak için gereken kod çevirilerini yürüt
   - Bu bir yerel işlevse, o veriler üzerinde kod dönüştürmelerini gerçekleştir
   - Sonuçtaki iletiyi, ileti tanımlayıcısının tüm ayrıntısı geri yüklenerek yerel kuyruğa koy
2. İleti uzak bir kuyruk içinse, iletiyi kuyruğa koyun.

Dağıtılmış bir kuyruğa alma ortamında teslim edilmemiş iletilerin nasıl işlendiğine ilişkin bilgi için İleti teslim edilemediğinde ne olur?başlıklı konuya bakın.

Çok hedefli programlama

Bir kuyruk yöneticisine bağlanma ve kural dışı durum raporlaması gibi IBM MQ Multicast programlama görevleri hakkında bilgi edinmek için bu bilgileri kullanın.

IBM MQ Multicast, kullanıcı için mümkün olduğu kadar saydam olacak ve yine de var olan uygulamalarla uyumlu olacak şekilde tasarlanmıştır. COMMINFO nesnesi tanımlanması ve TOPIC nesnesinin **MCAST** ve **COMMINFO** değiştirgelerinin ayarlanması, var olan IBM MQ uygulamalarının çoklu yayın kullanabilmesi için önemli ölçüde yeniden yazma gerektirmediği anlamına gelir. Ancak, bazı sınırlamalar (daha fazla bilgi için bkz. “Çoklu yayın ve MQI” sayfa 999) ve dikkate alınacak bazı güvenlik sorunları olabilir (daha fazla bilgi için bkz. Çoklu yayın güvenliği).

Çoklu yayın ve MQI

Ana Message Queue Interface (MQI) kavramlarını ve bunların IBM MQ Multicast ile ilişkisini anlamak için bu bilgileri kullanın.

Çoklu yayın abonelikleri sürekli değildir; fiziksel kuyruk olmadığı için, sürekli abonelikler tarafından oluşturulan çevrimdışı iletilerin saklanacağı yer yoktur.

Bir uygulama çok hedefli bir konuya abone olduktan sonra, uygulamaya, bir kuyruk tanıtıcısı gibi kullanabileceği ya da MQGET ' i kullanabileceği bir nesne tanıtıcısı verilir. Bu, yalnızca yönetilen çoklu yayın aboneliklerinin (MQSO_MANAGED ile oluşturulan abonelikler) desteklendiği anlamına gelir; bir abonelik yapılamaz ve iletileri bir kuyrukta 'gösteremez'. Bu, iletilerin abonelik çağrısında döndürülen nesne tanıtıcısından tüketilmesi gerektiği anlamına gelir. İstemcide, iletiler istemci tarafından tüketilinceye kadar ileti arabelleğinde saklanır; ek bilgi için bkz. İstemci yapılanış kütüğünün MessageBuffer kısmı . İstemci yayınlama hızına ayak uydurmazsa, iletiler gerektiği gibi atılır ve en eski iletiler atılır.

Olağan durumda, bir uygulamanın TOPIC nesnesinin MCAST özniteliğini ayarlayarak, Multicast kullanıp kullanmamasına ilişkin bir yönetim kararıdır. Bir yayınlama uygulamasının çoklu yayın kullanılmamasını sağlaması gerekiyorsa, `MQOO_NO_MULTICAST` seçeneğini kullanabilir. Benzer şekilde, abone olan bir uygulama, `MQSO_NO_MULTICAST` seçeneğine abone olarak çoklu yayın kullanılmamasını güvenceye alabilir.

IBM MQ Multicast, ileti seçicilerinin kullanımını destekler. Seçici, bir uygulama tarafından yalnızca seçim dizgisinin temsil ettiği SQL92 sorgusuna uyan özelliklere sahip iletilere olan ilgisini kaydetmek için kullanılır. İleti seçiciler hakkında daha fazla bilgi için bkz. “Seçiciler” sayfa 29.

Aşağıdaki çizelge, tüm ana MQI kavramlarını ve bunların Multicast ile nasıl ilişkilendirildiğini listeler:

Çizelge 154. MQI kavramları ve bunların çoklu yayınla ilişkisi

MQI Kavramı	Çoklu yayın kullanılmaya çalışıldığında yapılacak işlem	Neden Kodu
Sıfır uzunluklu bir ileti konması	Reddedildi	2005 (07D5) (RC2005): MQRC_BUFFER_LENGTH_ERROR
Gruplandırma	Reddedildi	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
Bölümleme	Reddedildi	2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWED
Dağıtım listeleri	Reddedildi	2154 (086A) (RC2154): MQRC_RECS_PRESENT_ERROR
MQINQ	Konu tanıtıcıları için reddedildi: Konuların MQINQ ve MQSET desteklenmiyor.	2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_INQUIRE
MQINQ	Yönetilen tanıtıcı için kabul edildi. Yalnızca `Yürürlükteki Derinlik` sorulabiliyor.	• Değer Yürürlükteki Derinlik ise, geçerli neden kodu yoktur. • Değer Yürürlükteki Derinlik dışında bir değerse, neden kodu: 2067 (0813) (RC2067): MQRC_SELECTOR_ERROR.
MQSET	Tüm tanıtıcıları için reddedildi.	2040 (07F8) (RC2040): MQRC_NOT_OPEN_FOR_SET
Hareketler (XA ya da değil)	Reddedildi	2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_VAR
İletiye göz at	Reddedildi	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
İletileri kilitle	Reddedildi	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
İşaretle Göz At	Reddedildi	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Bağlamı geçir	Reddedildi	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPUT1	Reddedildi. Çok hedefli bir konuda MQPUT1 ve deneme işlemi geçersizdir.	2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
Sürekli abonelik	Konu "Yalnızca çok hedefli" olarak işaretlenirse reddedilir, tersi durumda çok hedefli olmayan bir abonelik yapılır.	2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED

Çizelge 154. MQI kavramları ve bunların çoklu yayınla ilişkisi (devamı var)		
MQI Kavramı	**Çoklu yayın kullanılmaya çalışıldığında yapılacak işlem**	**Neden Kodu**
TopicString > 255	Reddedildi. Konu dizgisi 255 karakterden fazlaysa, istemcide reddedilir.	2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR
Yönetilmeyen abonelik yapıldı	Konu "Yalnızca çok hedefli" olarak işaretlenirse reddedilir, tersi durumda çok hedefli olmayan bir abonelik yapılır.	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPMO_NOT_OWN_SUBS	Reddedildi	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Aşağıdaki öğeler, önceki çizelgedeki bazı MQI kavramlarını genişletir ve çizelgede olmayan MQI kavramlarına ilişkin bilgi sağlar:

İleti kalıcılığı
Sürekli olmayan çoklu yayın aboneleri için, yayınlayıcıdan gelen kalıcı iletiler kurtarılamaz bir şekilde teslim edilir.

İleti kesilmesi
İleti kesilmesi desteklenir; bu, bir uygulamanın aşağıdakileri yapabileceği anlamına gelir:

1. MQGET komutunu verin.
2. MQRC_TRUNCATED_MSG_FAILED değerini alın.
3. Daha büyük bir arabellek ayırın.
4. İletiyi almak için MQGET ' i yeniden yayınlayın.

Abonelik süre bitimi
Abonelik süre bitimi desteklenmiyor. Süre bitimini ayarlama girişimi yoksayılır.

Çoklu yayın için yüksek kullanılabilirlik

IBM MQ Multicast continuous peer-to-peer operation; IBM MQ bir IBM MQ kuyruk yöneticisine bağlansa da, iletiler o kuyruk yöneticisinden geçmez.

Çok hedefli konu nesnesini MQOPEN ya da MQSUB için bir kuyruk yöneticisiyle bağlantı kurulması gerekse de, iletilerin kendileri kuyruk yöneticisinden geçmez. Bu nedenle, çok hedefli konu nesnesinde MQOPEN ya da MQSUB tamamlandıktan sonra, kuyruk yöneticisiyle bağlantı kesilse bile çok hedefli iletileri iletmeye devam edilebilir. İki işletim kipi vardır:

Kuyruk yöneticisiyle olağan bir bağlantı kurulur
Kuyruk yöneticisiyle bağlantı kurulurken çok hedefli iletişim kurulabilecektir. Bağlantı başarısız olursa, normal MQI kuralları uygulanır; örneğin, çok hedefli nesne tanıtıcısı için MQPUT, 2009 (07D9) (RC2009): MQRC_CONNECTION_BROKENdeğerini döndürür.

Kuyruk yöneticisine yeniden bağlanan bir istemci bağlantısı yapıldı
Çok hedefli iletişim, yeniden bağlantı döngüsü sırasında bile mümkündür. Bu, kuyruk yöneticisiyle bağlantı kesildiğinde bile çoklu yayın iletilerinin yerleştirilmesi ve tüketilmesinden etkilenmediği

anlamına gelir. İstemci bir kuyruk yöneticisine yeniden bağlanmayı dener ve yeniden bağlanma başarısız olursa, bağlantı tanıtıcısı bozulur ve çok hedefli olanlar da içinde olmak üzere tüm MQI çağrıları başarısız olur. Daha fazla bilgi için bakınız: Automatic client reconnection

Herhangi bir uygulama belirtik olarak bir MQDISC yayınlarsa, tüm çok hedefli abonelikler ve nesne tanıtıcıları kapatılır.

Çok hedefli sürekli eşler arası işlem

İstemciler arasındaki eşler arası iletişimin avantajlarından biri, iletilerin kuyruk yöneticisinden akmasına gerek olmamasıdır; bu nedenle, kuyruk yöneticisine yönelik bağlantı kesilirse, ileti aktarımı devam eder. Bu kipin sürekli ileti gereksinimleri için aşağıdaki kısıtlamalar geçerlidir:

- Sürekli işlem için MQCNO_RECONNECT_ * seçeneklerinden biri kullanılarak bağlantı kurulmalıdır. Bu işlem, iletişim oturumu bozulmuş olsa da, gerçek bağlantı tanıtıcısı bozuk değildir ve bunun yerine yeniden bağlanma durumunda olduğu anlamına gelir. Yeniden bağlanma başarısız olursa, bağlantı tanıtıcısı kesilir ve diğer tüm MQI çağrılarını engeller.
- Bu kipte yalnızca MQPUT, MQGET, MQINQ ve Zamanuyumsuz Tüketim desteklenir. Herhangi bir MQOPEN, MQCLOSE ya da MQDISC fiilinin tamamlanması için kuyruk yöneticisiyle yeniden bağlantı kurulması gerekir.
- Kuyruk yöneticisine giden durum akışları durur; bu nedenle, kuyruk yöneticisindeki herhangi bir durum eski ya da eksik olabilir. Bu, istemcilerin ileti gönderiyor ve alıyor olabileceği ve kuyruk yöneticisinde bilinen bir durum olmadığı anlamına gelir. Daha fazla bilgi için bakınız: Multicast application monitoring

Çok hedefli ileti alışverişi için MQI 'da veri dönüştürme

Veri dönüştürmenin IBM MQ Multicast ileti sistemi için nasıl çalıştığını anlamak için bu bilgileri kullanın.

IBM MQ Multicast, paylaşılan, bağlantısız bir iletişim kuralıdır ve her bir istemcinin veri dönüştürme için belirli isteklerde bulunması mümkün değildir. Aynı çoklu yayın akımına abone olan her istemci aynı ikili verileri alır; bu nedenle, IBM MQ veri dönüştürmesi gerekiyorsa, dönüştürme her istemcide yerel olarak gerçekleştirilir.

Veriler istemcide IBM MQ Multicast trafiği için dönüştürülür. **MQGMO_CONVERT** seçeneği belirtilirse, veri dönüştürme istendiği gibi yapılır. Kullanıcı tanımlı biçimler, istemcide veri dönüştürme çıkışının kurulu olması gerekir; istemci ve sunucu paketlerinde bulunan kitaplıklara ilişkin bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941 .

Veri dönüştürmeyi yönetme hakkında bilgi için bkz. Multicast ileti sistemi için veri dönüştürmeyi etkinleştirme.

Veri dönüştürme hakkında daha fazla bilgi için bkz. Veri dönüştürme.

Veri dönüştürme çıkışları ve `ClientExitPath`ile ilgili daha fazla bilgi için bkz. ClientExitİstemci yapılandırma dosyasının yol kısmı.

Çok hedefli kural dışı durum raporlaması

IBM MQ Çok hedefli olay işleyicileri ve raporlama IBM MQ Çok hedefli kural dışı durumları hakkında bilgi edinmek için bu bilgileri kullanın.

IBM MQ Multicast, standart IBM MQ olay işleyici mekanizması kullanılarak bildirilen çoklu yayın olaylarını bildirmek için olay işleyiciyi çağırarak sorun belirlemeye yardımcı olur.

Tek bir Çoklu yayın olayı, aynı çoklu yayın ileticisini ya da alıcısını kullanan birden çok `MQHCONN` bağlantı tanıtıcısı olabileceği için birden çok IBM MQ olayının çağrılmasına neden olabilir. Ancak, her çoklu yayın kural dışı durumu, IBM MQ bağlantısı başına yalnızca bir olay işleyicinin çağrılmasına neden olur.

IBM MQ `MQCBDO_EVENT_CALL` sabiti, uygulamaların yalnızca IBM MQ olaylarını almak için bir geri çağırma kaydetmesini sağlar ve `MQCBDO_MC_EVENT_CALL` , uygulamaların yalnızca çok hedefli olayları almak için bir geri çağırma kaydetmesini sağlar. Her iki değişmez de kullanılırsa, her iki olay tipi de alınır.

Çok hedefli olaylar isteniyor

IBM MQ Çoklu yayın olayları, `cbd.Options` alanında `MQCBDO_MC_EVENT_CALL` sabitini kullanır. Aşağıdaki örnek, çoklu yayın olaylarının nasıl isteneceğini gösterir:

```
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);
```

`cbd.Options` alanı için `MQCBDO_MC_EVENT_CALL` seçeneği belirtildiğinde, olay işleyici bağlantı düzeyi olayları yerine yalnızca IBM MQ Multicast olayları gönderilir. Her iki olay tipinin olay işleyiciye gönderilmesini istemek için, uygulamanın `cbd.Options` alanında `MQCBDO_EVENT_CALL` değişmezini ve aşağıdaki örnekte gösterildiği gibi `MQCBDO_MC_EVENT_CALL` değişmezini belirtmesi gerekir:

```
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);
```

Bu sabitlerden hiçbiri kullanılmazsa, olay işleyiciye yalnızca bağlantı düzeyi olayları gönderilir.

`Options` alanına ilişkin değerlerle ilgili daha fazla bilgi için Options (MQLONG)konusuna bakın.

Çok hedefli olay biçimi

IBM MQ Çok hedefli kural dışı durumlar, geri çağırma işlevinin **Buffer** değiştirgesinde döndürülen bazı destekleyici bilgileri içerir. **Buffer** işaretçisi bir işaretçi dizisini gösterir ve `MQCBC.DataLength` alanı dizinin bayt cinsinden boyutunu belirtir. Dizinin ilk öğesi her zaman olayın kısa metin açıklamasını gösterir. Olayın tipine bağlı olarak daha fazla parametre sağlanabilir. Aşağıdaki çizelge kural dışı durumları listeler:

Çizelge 155. Çok hedefli olay kodu açıklamaları

Olay Kodu	**Açıklama**	**Ek veriler**
MQMCEV_PACKET_LOSS	Kurtarılamaz paket kaybı	Kayıp paket sayısı
MQMCEV_HEARTBEAT_TIMEOUT	Sağlıklı işletim denetimi paketinin uzun süre yokluğu	Yok
MQMCEV_VERSION_CONFLICT	Daha yeni protokol sürümü paketlerinin alımı	Yok
MQMCEV_GÜVENILIRLIK	İleleyicinin ve alıcının farklı güvenilirlik kipleri	Yok
MQMCEV_CLOSED_TRANS	Konu iletimi 1 kaynak tarafından kapatıldı	Yok
MQMCEV_STREAM_ERROR	Akışta hata saptandı	Yok
MQMCEV_YENI_KAYNAK	Yeni bir kaynak konuyla ilgili iletim yapmaya başlıyor	Kaynak yapı
MQMCEV_RECEIVE_QUEUE_TRIMMED	Zaman ya da alan süresi dolması nedeniyle PacketQ ' dan kaldırılan paketler	Kırpılmış paket sayısı
MQMCEV_PACKET_LOSS_NACK_SÜRE bitimi	NACK süre bitimi nedeniyle kurtarılamaz paket kaybı	Kayıp paket sayısı
MQMCEV_ACK_RETRIES_AŞILDI	**max_ack_retries** aşıldıktan sonra geçmişten kaldırılan paketler	Kaldırılan paket sayısı

Çizelge 155. Çok hedefli olay kodu açıklamaları (devamı var)

Olay Kodu	Açıklama	Ek veriler
MQMCEV_STREAM_SUSPEND_NACK	Bu konu tarafından kabul edilen bir akışta NACK ' ler askıya alındı	Akış tanıtıcısını askıya al Akışın askıya alındığı süre (milisaniye)
MQMCEV_STREAM_RESUME_NACK	Bir akışta askıya alındıktan sonra NACK ' ler sürdürüldü	Akış Tanıtıcısı
MQMCEV_STREAM_EXPAT_TR	Bu konu tarafından kabul edilen bir akış, bir ihraç isteği nedeniyle reddedildi	Akış Tanıtıcısı
MQMCEV_FIRST_MESSAGE	Bir kaynaktan gelen ilk ileti	İleti numarası
MQMCEV_LATE_JOIN_FAILURE	Geç birleştirme oturumu başlatılamadı	Yok
MQMCEV_MESSAGE_LOSS	Kurtarılamaz ileti kaybı	Kaybolan iletilerin sayısı
MQMCEV_SEND_PACKET_FAILURE	Çok hedefli iletici çok hedefli bir paket gönderemedi	Yok
MQMCEV_REPAIR_DELAY	Çok hedefli alıcı, bekleyen bir NAK için onarım paketi almadı	Yok
MQMCEV_MEMORY_ALERT_ON	Alıcı alma arabellekleri doluyor	Arabellek havuzu kullanım yüzdesi
MQMCEV_MEMORY_ALERT_OFF	Alıcı alma arabellekleri normale döndü	Arabellek havuzu kullanım yüzdesi
MQMCEV_NACK_ALERT_ON	Alıcı onarım paketi istek hızı yüksek filigran değerine ulaştı	Paket/saniye cinsinden geçerli onarım isteği oranı
MQMCEV_NACK_ALERT_OFF	Alıcı onarım paketi istek hızı normal.	Paket/saniye cinsinden geçerli onarım isteği oranı
MQMCEV_REPAIR_ALERT_ON	Verici onarım paketi gönderme hızı yüksek su işaretine ulaştı	Yok
MQMCEV_REPAIR_ALERT_OFF	Verici onarım paketi gönderme hızı normale düştü	Yok
MQMCEV_SHM_DEST_KULLANILAMAZ	Bir iletici konu hedefi tarafından kullanılan Paylaşılan Bellek bölgesinin kullanılamaz olduğu saptandı	Yok
MQMCEV_SHM_PORT_KULLANILAMAZ	Bir alıcı yönetim ortamı tarafından kullanılan Paylaşılan Bellek kapısının kullanılamaz olduğu saptandı	Yok
MQMCEV_CCT_GETTIME_FAILED	Eşgüdümlü Küme Saatinden alma zamanı başarısız oldu	Yok
MQMCEV_DEST_INTERFACE_FAILURE	Bir iletici konu hedefi tarafından kullanılan ağ arabirimi başarısız oldu ve bir yedek ağ arabirimi kullanılamıyor	

Çizelge 155. Çok hedefli olay kodu açıklamaları (devamı var)		
Olay Kodu	**Açıklama**	**Ek veriler**
MQMCEV_DEST_INTERFACE_FAILOVER	Bir iletici konu hedefi tarafından kullanılan ağ arabirimi başarısız oldu ve başka bir Arabirime başarılı bir hata durumunda geçiş işlemi tamamlandı	
MQMCEV_PORT_INTERFACE-HATA	Bir günlük nesnesi rmmPort tarafından kullanılan ağ arabirimi başarısız oldu ve bir yedek ağ arabirimi kullanılamıyor (ya da başarısız oldu)	RMM yapılandırması
MQMCEV_PORT_INTERFACE_FAILOVER	Bir günlük nesnesi rmmPort tarafından kullanılan ağ arabirimi başarısız oldu ve başka bir Arabirime başarılı bir hata durumunda geçiş işlemi tamamlandı	RMM yapılandırması

C dilinde kodlama

C içindeki IBM MQ programlarını kodlarken aşağıdaki bölümlerdeki bilgileri not edin.

- "MQI çağrılarına ilişkin değiştirgeler" sayfa 1005
- "Tanımlanmamış veri tipine sahip parametreler" sayfa 1005
- "Veri türleri" sayfa 1006
- "İkili dizgiler işleniyor" sayfa 1006
- "Karakter dizilimlerini işleme" sayfa 1006
- "Yapılara ilişkin ilk değerler" sayfa 1006
- "Dinamik yapılar için ilk değerler" sayfa 1007
- "C++ dilinden kullan" sayfa 1007

MQI çağrılarına ilişkin değiştirgeler

yalnızca giriş tipindeki ve MQHCONN, MQHOBJ, MQHMSG ya da MQLONG tipindeki değiştirgeler değere göre geçirilir; diğer tüm değiştirgeler için, değiştirgenin *adresi* değer temelinde geçirilir.

Adres tarafından geçirilen tüm değiştirgelerin, bir işlev her çağrıldığında belirtilmesi gerekmez. Belirli bir parametrenin gerekli olmadığı durumlarda, parametre verilerinin adresi yerine işlev çağrısında parametre olarak boş bir gösterge belirtilebilir. Bunun mümkün olduğu parametreler, çağrı açıklamalarında tanımlanır.

İşlevin değeri olarak hiçbir parametre döndürülmez; C terminolojisinde bu, tüm işlevlerin geçersiz olduğu anlamına gelir.

İşlevin öznitelikleri MQENTRY makro değişkeniyle tanımlanır; bu makro değişkeninin değeri ortama bağlıdır.

Tanımlanmamış veri tipine sahip parametreler

MQGET, MQPUT ve MQPUT1 işlevlerinin her birinin, tanımlanmamış veri tipi olan bir **Buffer** değiştirgesi vardır. Bu parametre, uygulamanın ileti verilerini göndermek ve almak için kullanılır.

Bu sıralamaya ilişkin parametreler, C örneklerinde MQBYTE dizileri olarak gösterilir. Parametreleri bu şekilde bildirebilirsiniz, ancak genellikle bunları iletideki verilerin düzenini açıklayan yapı olarak

bildirmeniz daha uygundur. İşlev değiştirgesi bir gösterge-to-void olarak bildirildiğinden, işlev çağrıldığında herhangi bir verinin adresi parametre olarak belirtilebilir.

Veri türleri

Tüm veri tipleri `typedef` deyimiyle tanımlanır.

Her veri tipi için, ilgili gösterge veri tipi de tanımlanır. İşaretçi veri tipinin adı, bir işaretçiyi göstermek için öneki P harfi olan temel veri ya da yapı veri tipinin adıdır. Göstergenin öznitelikleri MQPOINTER makro değişkeniyle tanımlanır; bu makro değişkeninin değeri ortama bağlıdır. Aşağıdaki kod, işaretçi veri tiplerinin nasıl bildirileceğini gösterir:

```
#define MQPOINTER                   /* depends on environment */
...
typedef MQLONG  MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;     /* pointer to MQMD */
```

İkili dizgiler işleniyor

İkili veri dizgileri, MQBYTEn veri tiplerinden biri olarak bildirilir.

Bu tipteki alanları kopyaladığınızda, karşılaştırdığınızda ya da ayarladığınızda, C işlevlerini `memcpy`, `memcmp`ya da `memset`kullanın:

```
#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));
```

MQBYTE24olarak bildirilen verilerle doğru çalışmadığından, strcpy, strcmp, strncpy ya da strncmp dizgi işlevlerini kullanmayın.

Karakter dizilimlerini işleme

Kuyruk yöneticisi uygulamaya karakter verileri döndürdüğünde, kuyruk yöneticisi karakter verilerini her zaman alanın tanımlı uzunluğuna kadar boşluklarla doldurur. Kuyruk yöneticisi boş sonlandırılmış dizgiler döndürmez, ancak bunları girişinizde kullanabilirsiniz. Bu nedenle, bu tür dizgileri kopyalarken, karşılaştırırken ya da birleştirirken strncpy, strncmp ya da strncat dizgi işlevlerini kullanın.

Dizginin boş değerle (strcpy, strcmp ve strcat) sonlandırılmasını gerektiren dizgi işlevlerini kullanmayın. Ayrıca, dizginin uzunluğunu saptamak için strlen işlevini kullanmayın; alanın uzunluğunu saptamak için sizeof işlevini kullanın.

Yapılara ilişkin ilk değerler

İçerme dosyası <cmqc.h>, bu yapıların eşgörünümlerini bildirirken yapılara ilişkin ilk değerleri sağlamak için kullanabileceğiniz çeşitli makro değişkenlerini tanımlar. Bu makro değişkenlerinin adları MQxxx_DEFAULT biçiminde olur; burada MQxxx, yapının adını gösterir. Onları şöyle kullan:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

Bazı karakter alanları için MQI, geçerli olan belirli değerleri tanımlar (örneğin, *StrucId* alanları ya da MQMD ' deki *Format* alanı için). Geçerli değerlerin her biri için iki makro değişkeni sağlanır:

- Bir makro değişkeni, değeri, alanın tanımlı uzunluğuyla tam olarak eşleşen, örtük boş değer dışında bir uzunluğa sahip bir dizgi olarak tanımlar. Aşağıdaki örneklerde, ¬ simgesi tek bir boş karakteri temsil eder:

```
#define MQMD_STRUC_ID "MD¬¬"
#define MQFMT_STRING "MQSTR¬¬¬"
```

  Bu formu memcpy ve memcmp işlevleriyle kullanın.
- Diğer makro değişkeni değeri bir karakter dizisi olarak tanımlar; bu makro değişkeninin adı, _ARRAY ile eklenen dizgi biçimi sonekinin adıdır. Örneğin:

```
#define MQMD_STRUC_ID_ARRAY 'M','D','¬','¬'
#define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
```

  Bir yapı eşgörünümü MQMD_DEFAULT makro değişkeni tarafından sağlananlardan farklı değerlerle bildirildiğinde alanı kullanıma hazırlamak için bu formu kullanın.

Dinamik yapılar için ilk değerler

Bir yapının değişken sayıda eşgörünümü gerektiğinde, eşgörünümler tipik olarak, calloc ya da malloc işlevleri kullanılarak dinamik olarak elde edilen ana bellekte yaratılır.

Bu tür yapılardaki alanları kullanıma hazırlamak için aşağıdaki teknik önerilir:

1. Yapıyı kullanıma hazırlamak için uygun MQxxx_DEFAULT makro değişkenini kullanarak yapının bir eşgörünümünü bildirin. Bu eşgörünüm, diğer eşgörünümler için *model* olur:

```
MQMD ModelMsgDesc = {MQMD_DEFAULT};
                                    /* declare model instance */
```

   Bildirimdeki durağan ya da otomatik anahtar sözcükleri, model eşgörünümüne gereken şekilde durağan ya da devingen geçerlik süresi verecek şekilde kodlayın.
2. Yapının dinamik bir eşgörünümü için depolama alanı elde etmek üzere calloc ya da malloc işlevlerini kullanın:

```
PMQMD InstancePtr;
InstancePtr = malloc(sizeof(MQMD));
                                    /* get storage for dynamic instance */
```

3. Model eşgörünümünü devingen örneğe kopyalamak için memcpy işlevini kullanın:

```
memcpy(InstancePtr,&ModelMsgDesc,sizeof(MQMD));
                                    /* initialize dynamic instance */
```

C++ dilinden kullan

C++ programlama dili için, üstbilgi dosyaları yalnızca C++ derleyicisi kullanıldığında içerilen aşağıdaki ek deyimleri içerir:

```
#ifdef __cplusplus
  extern "C" {
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Windows Visual Basic içinde kodlama

Microsoft Visual Basiciçinde IBM MQ programlarını kodlarken göz önünde bulundurulması gereken bilgiler. Visual Basic yalnızca Windowsüzerinde desteklenir.

Not:

Stabilized IBM WebSphere MQ 7.0' den .NET ortamının dışında, Visual Basic (VB) desteği IBM WebSphere MQ 6.0 düzeyinde dengelenmiştir. IBM WebSphere MQ 7.0 ya da sonraki sürümlere eklenen yeni işlevin çoğu VB uygulamaları tarafından kullanılamaz. VB.NET' de programlama kullanıyorsanız, .NETiçin IBM MQ sınıflarını kullanın. Daha fazla bilgi için bkz .NET uygulamaları geliştirilmesi.

Deprecated IBM MQ 9.0' den Microsoft Visual Basic 6.0 desteği kullanımdan kaldırılmıştır. .NET için IBM MQ sınıfları önerilen değiştirme teknolojisidir.

Visual Basic ile IBM MQarasında geçen ikili verilerin istenmeyen çevirisini önlemek için MQSTRING yerine MQBYTE tanımlaması kullanın. CMQB.BAS , bir C byte tanımlamasına eşdeğer birkaç yeni MQBYTE tipini tanımlar ve bunları IBM MQ yapıları içinde kullanır. Örneğin, MQMD (ileti tanımlayıcı) yapısı için MsgId (ileti tanıtıcısı) MQBYTE24olarak tanımlanır.

Visual Basic ' in gösterge veri tipi yoktur; bu nedenle, diğer IBM MQ veri yapılarına yönelik başvurular işaretçi yerine göreli konuma göre olur. İki bileşen yapısından oluşan bir bileşik yapı bildirin ve çağrıda bileşik yapıyı belirtin. IBM MQ support for Visual Basic , bunu olanaklı kılmak ve istemci uygulamalarının bir istemci bağlantısında kanal özelliklerini belirtmesine izin vermek için bir MQCONNXAny çağrısı sağlar. Tipik MQCNO yapısı yerine tip atanmamış bir yapı (MQCNOCD) kabul eder.

MQCNOCD yapısı, MQCD 'yi izleyen bir MQCNO' dan oluşan bileşik bir yapıdır. Bu yapı, CMQXB çıkış üstbilgi dosyasında bildirilir. Bir MQCNOCD yapısını kullanıma hazırlamak için MQCNOCD_DEFAULTS yordamını kullanın. MQCONNX çağrıları yapan bir örnek sağlanmıştır (amqscnxb.vbp).

MQCONNXAny 'nin değiştirgeleri MQCONNX ile aynıdır; ancak, **ConnectOpts** değiştirgesi MQCNO veri tipi yerine Any veri tipinde olarak bildirilir. Bu, işlevin MQCNO ya da MQCNOCD yapısını kabul etmesini sağlar. Bu işlev, ana üstbilgi dosyası CMQB ' de bildirilir.

İlgili kavramlar

“Windows ' da Visual Basic programlarının hazırlanması” sayfa 977
Windowsüzerinde Microsoft Visual Basic programlarını kullanırken göz önünde bulundurulması gereken bilgiler.

İlgili başvurular

“Visual Basic uygulamalarının IBM MQ MQI client koduyla bağlanması” sayfa 881
Microsoft Visual Basic uygulamalarını Windowsüzerindeki IBM MQ MQI client koduyla bağlayabilirsiniz.

COBOL dilinde kodlama

COBOL dilinde IBM MQ programlarını kodlarken aşağıdaki bölümdeki bilgileri not edin.

Adlandırılmış sabitler

Değişmezlerin adları, adın bir parçası olarak alt çizgi karakterini (_) içeren şekilde gösterilir. COBOL dilinde, alt çizgi yerine kısa çizgi karakterini (-) kullanmanız gerekir. Karakter dizilimi değerleri olan sabitler, dizilim sınırlayıcı olarak tek tırnak işareti karakterini (') kullanır. Derleyicinin bu karakteri kabul etmesini sağlamak için, APOST derleyici seçeneğini kullanın.

CMQV kopya dosyası, adlandırılmış değişmezlerin bildirimlerini level-10 öğe olarak içerir. Değişmezleri kullanmak için, level-01 öğesini belirtik olarak bildirin ve değişmezlerin bildirimlerini kopyalamak için COPY deyimini kullanın:

```
WORKING-STORAGE SECTION.
```

```
01 MQM-CONSTANTS.
COPY CMQV.
```

Ancak, bu y " ntem, de§iŸmez de§iŸmezlerin, bu de§iŸmezlere gönderide bulunulmasa da programdaki saklama alanların kullanmasına neden olur. Değişmezler aynı çalıştırma birimi içindeki birçok ayrı programda yer alıyorsa, değişmezlerin birden çok kopyası vardır; bu, önemli miktarda ana saklama alanının kullanılmasıyla sonuçlanabilir. Bunu önlemek için, level-01 bildirimine GLOBAL yantümcesini ekleyin:

```
* Declare a global structure to hold the constants
01 MQM-CONSTANTS GLOBAL.
COPY CMQV.
```

Bu işlem, saklama alanını çalıştırma birimi içindeki yalnızca *tek bir* değişmez kümesi için ayırır; ancak, değişmezlere, yalnızca level-01 bildirimini içeren program değil, çalıştırma birimi içindeki *herhangi bir* programı gönderme yapabilir.

Yapı hizalamasının sağlanması

MQ çağrısında başlamak üzere geçirilen IBM MQ yapılarının sözcük sınırlarına hizalanması gerektiğinden emin olmak için dikkatli olunmalıdır. Bir sözcük sınırı 32 bit işlemler için 4 bayt, 64 bit işlemler için 8 bayt ve 128 bit işlemler için 16 bayttır (IBM i).

Olanaklı olduğu yerlerde, tüm IBM MQ yapılarını, tümü sınırla hizalanacak şekilde bir araya yerleştirin.

System/390 çevirici dilinde kodlama (İleti kuyruğu arabirimi)

Çevirici dilinde IBM MQ for z/OS programlarını kodlarken aşağıdaki bölümlerdeki bilgilere dikkat edin.

Adlar

Çağrıların tanımlarındaki parametrelerin adları ve yapı tanımlarındaki alanların adları büyük ve küçük harf karışık olarak gösterilir. IBM MQile verilen çevirici dili makrolarında tüm adlar büyük harfli olur.

MQI çağrılarını kullanma

MQI bir çağrı arabirimidir, bu nedenle çevirici dili programları işletim sistemi bağlantı kuralını izlemelidir.

Özellikle, bir MQI çağrısı yayınlamadan önce, çevirici dil programlarının R13 ' ü en az 18 tam sözcükten oluşan bir saklama alanına kaydetmeleri gerekir. Bu saklama alanı, çağrılan program için saklama alanı sağlar. İçerikleri yok edilmeden önce çağıranın kayıtlarını saklar ve geri dönüşte çağıranın kayıt defterlerini geri yükler.

Not: Bu, dinamik saklama alanını ayarlamak için DFHEIENT makrosunu kullanan, ancak R13 'ten diğer kayıt dosyalarına varsayılan DATAREG' yi geçersiz kılmayı seçen CICS çevirici dili programları için önemlidir. CICS Resource Manager Arabirimi sınırlı kod öbeğinden denetim aldığında, kayıtların yürürlükteki içeriğini R13 ' ün gösterdiği adreste saklar. Bu amaçla bir saklama alanı ayırmanın başarısız olması beklenmeyen sonuçlar verir ve CICSiçinde olağandışı bir sona neden olabilir.

Değişmezlerin bildirilmesi

Çoğu sabit, makro CMQA ' da eşit olarak bildirilir.

Ancak, aşağıdaki sabitler eşit olarak tanımlanamaz ve varsayılan seçenekleri kullanarak makroyu çağırdığınızda bunlar dahil edilmez:

- MQACT_NONE
- MQCI_NONE
- MQFMT_NONE
- MQFMT_ADMIN
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- MQFMT_EVENT
- MQFMT_IMS
- MQFMT_IMS_VAR_STRING
- MQFMT_PCF
- MQFMT_STRING
- MQFMT_TRIGGER
- MQFMT_XMIT_Q_HEADER
- MQMI_NONE

Bunları içermek için, makroyu çağırdığınızda EQUONLY=NO anahtar sözcüğünü ekleyin.

CMQA birden çok bildirime karşı korunduğundan, bunu birçok kez ekleyebilirsiniz. Ancak, EQUONLY anahtar sözcüğü yalnızca, makro ilk kez eklendiğinde yürürlüğe girer.

Bir yapının adının belirtilmesi

Bir yapının birden çok örneğinin bildirilmesine izin vermek için, yapı oluşturan makro, kullanıcı tarafından belirtilen bir dizgi ve altçizgi karakteri (_) ile her alanın adının önekini ekler.

Makroyu çağırdığınızda dizgiyi belirtin. Dizgi belirtmezseniz, makro öneki oluşturmak için yapının adını kullanır:

```
* Declare two object descriptors
CMQODA         Prefix used="MQOD_" (the default)
MY_MQOD CMQODA         Prefix used="MY_MQOD_"
```

Çağrı açıklamaları içindeki yapı bildirimleri varsayılan öneki gösterir.

Yapı formunun belirtilmesi

Makrolar, DSECT parametresi tarafından denetlenen iki biçimden birinde yapı bildirimleri oluşturabilir:

DSECT = EVET
Yeni bir veri bölümü başlatmak için çevirici dili DSECT yönergesi kullanılır; yapı tanımı DSECT deyimini hemen izler. Saklama alanı ayrılmadığı için kullanıma hazırlama yapılamaz. Makro çağrısındaki etiket, veri kısmının adı olarak kullanılır; etiket belirtilmezse, yapının adı kullanılır.

DSECT = HAYıR
Derleyici dili DC yönergeleri, yapıyı yordamın yürürlükteki konumunda tanımlamak için kullanılır. Alanlar, makro çağrısında ilgili parametreleri kodlayarak belirtebileceğiniz değerlerle kullanıma hazırlanır. Makro çağrısında değer belirtilmeyen alanlar varsayılan değerlerle kullanıma hazırlanır.

DSECT parametresi belirlenmezse, DSECT = NO varsayılır.

Listelemeyi denetleme

Çevirici dili listelemesinde yapı bildiriminin görünüşünü LIST parametresiyle denetleyebilirsiniz:

LIST = EVET
Yapı bildirimi, çevirici dili listelemesinde görünür.

LISTE = HAYıR
Yapı bildirimi, çevirici dili listelemesinde görünmüyor. LIST parametresi belirtilmezse bu kabul edilir.

Alanlar için başlangıç değerlerinin belirtilmesi

Bir yapıdaki bir alanı kullanıma hazırlamak için kullanılacak değeri, o alanın adını (önek olmadan) makro çağrısında parametre olarak kodlayarak (gerekli değerle birlikte) belirtebilirsiniz.

Örneğin, bir ileti tanımlayıcı yapısını MQMT_REQUEST ile kullanıma hazırlanan *MsgType* alanıyla ve MY_REPLY_TO_QUEUE dizgisiyle kullanıma hazırlanan *ReplyToQ* alanıyla bildirmek için aşağıdaki kodu kullanın:

```
MY_MQMD    CMQMDA      MSGTYPE=MQMT_REQUEST,       X
REPLYTOQ=MY_REPLY_TO_QUEUE
```

Makro çağrısında bir değer olarak adlandırılan bir değişmez (ya da eşit) belirtirseniz, adı belirtilen değişmezi tanımlamak için CMQA makrosunu kullanın. Karakter dizgisi olan tek tırnak işareti ('') değerlerine girmemelisiniz.

Yeniden girilebilir programlar yazılıyor

IBM MQ , hem giriş hem de çıkış için yapılarını kullanır. Programınızın yeniden ağırlanabilmesini istiyorsanız:

1. Yapıların çalışan depolama sürümlerini DSECT olarak tanımlayın ya da önceden tanımlanmış bir DSECT içindeki yapıları tanımlayın. Daha sonra, DSECT ' yi aşağıdaki yöntemle elde edilen depolama alanına kopyalayın:
   - Toplu iş ve TSO programları için, STORAGE ya da GETMAIN z/OS çevirici makroları
   - CICSiçin, çalışan depolama DSECT (DFHEISTG) ya da EXEC CICS GETMAIN komutu

   Bu çalışan depolama yapılarını doğru bir şekilde başlatmak için, ilgili yapının sabit bir sürümünü çalışan depolama sürümüne kopyalayın.

   Not: MQMD ve MQXQH yapılarının her biri 256 byte 'tan uzun. Bu yapıları saklama alanına kopyalamak için MVCL çevirici yönergesini kullanın.
2. CALL makrosunun LIST formunu (MF=L) kullanarak saklama alanında yer ayırın. Bir MQI çağrısı yapmak için CALL makrosunu kullandığınızda, “CEDF Kullanılması” sayfa 1012örneğinde gösterildiği gibi, daha önce ayrılmış depolamayı kullanarak makronun EXECUTE formunu (MF=E) kullanın. Bunun nasıl yapılacağını gösteren daha fazla örnek için, IBM MQile birlikte gönderilen çevirici dili örnek programlarına bakın.

Programınızın yeniden girilebilir olup olmadığını belirlemenize yardımcı olması için çevirici dili RENT seçeneğini kullanın.

Yeniden girilebilir programlar yazma hakkında bilgi için bkz. *z/OS MVS Application Development Guide: Assembler Language Programs*.

CEDF Kullanılması

Programınızda hata ayıklamanıza yardımcı olması için CICStarafından sağlanan CEDF (CICS Execution Diagnostic Facility) hareketini kullanmak istiyorsanız, `,VL` anahtar sözcüğünü her `CALL` deyimine ekleyin; örneğin:

```
CALL MQCONN,(NAME,HCONN,COMPCODE,REASON),MF=(E,PARMAREA),VL
```

Önceki örnek, yeniden girilebilir çevirici dil kodudur; burada `PARMAREA` , belirttiğiniz çalışma deposunda bulunan bir alandır.

MQI çağrılarını kullanma

MQI bir çağrı arabirimidir, bu nedenle çevirici dili programları işletim sistemi bağlantı kuralını izlemelidir. Özellikle, bir MQI çağrısı yayınlamadan önce, çevirici dil programlarının R13 ' ü en az 18 tam sözcükten oluşan bir saklama alanına kaydetmeleri gerekir. Bu saklama alanı, çağrılan program için saklama alanı sağlar. İçerikleri yok edilmeden önce çağıranın kayıtlarını saklar ve geri dönüşte çağıranın kayıt defterlerini geri yükler.

Not: Bu, dinamik saklama alanını ayarlamak için DFHEIENT makrosunu kullanan, ancak R13 'ten diğer kayıt dosyalarına varsayılan DATAREG' yi geçersiz kılmayı seçen CICS çevirici dili programları için önemlidir. CICS Resource Manager Arabirimi sınırlı kod öbeğinden denetim aldığında, kayıtların yürürlükteki içeriğini R13 ' ün gösterdiği adreste saklar. Bu amaçla uygun bir saklama alanı ayırmamanız beklenmedik sonuçlar verir ve büyük olasılıkla CICS' da olağandışı bir sona neden olur.

IBM i RPG ' de IBM MQ programlarının kodlanması (yalnızcaIBM i)

IBM MQ belgelerinde, çağrıların parametreleri, veri tiplerinin adları, yapı alanları ve sabit adların tümü uzun adları kullanılarak açıklanır. RPG dilinde, bu adlar altı ya da daha az büyük harfli karaktere kısaltılır.

Örneğin, RPG ' de *MsgType* alanı *MDMT* olur. Daha fazla bilgi için bkz. IBM i Application Programming Reference (ILE/RPG).

PL/I dilinde kodlama (yalnızcaz/OS)

PL/I içinde IBM MQ için kodlama yaparken yararlı bilgiler

Yapılar

Yapılar BASED özniteliğiyle bildirilir ve bu nedenle, program bir yapının bir ya da daha fazla örneğini bildirmediği sürece herhangi bir depolama alanı kaplamaz.

Bir yapının eşgörünümü, `like` özniteliği kullanılarak bildirilebilir; örneğin:

```
dcl my_mqmd    like MQMD; /* one instance */
dcl my_other_mqmd like MQMD; /* another one */
```

Yapı alanları INITIAL özniteliğiyle bildirilir; bir yapının eşgörünümünü bildirmek için `like` özniteliği kullanıldığında, o eşgörünüm o yapı için tanımlanan ilk değerleri devralır. Yalnızca, gerekli değerin ilk değerden farklı olduğu alanları ayarlamanız gerekir.

PL/I büyük/küçük harfe duyarlı değildir; bu nedenle, çağrıların, yapı alanlarının ve değişmezlerin adları küçük harfli, büyük harfli ya da büyük harfli olarak kodlanabilir.

Adlandırılmış sabitler

Adı belirtilen değişmezler makro değişkenleri olarak bildirilir; sonuç olarak, program tarafından gönderme yapılmayan değişmezler, derlenen yordamda herhangi bir saklama yeri kaplamaz.

Ancak, kaynağın makro ön işlemcisi tarafından işlenmesine neden olan derleyici seçeneği, program derlendiğinde belirtilmelidir.

Tüm makro değişkenleri, sayısal değerleri temsil eden değişkenler de dahil, karakter değişkenleridir. Bu, sezgisel olmayan bir durum gibi görünse de, makro değişkenleri makro işlemcisi tarafından değiştirildikten sonra herhangi bir veri tipi çakışmasıyla sonuçlanmaz; örneğin:

```
%dcl MQMD_STRUC_ID char;
%MQMD_STRUC_ID = '''MD ''';

%dcl MQMD_VERSION_1 char;
%MQMD_VERSION_1 = '1';
```

IBM MQ örnek yordam programlarının kullanılması

Bu örnek programlar yordamsal dillerde yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir. Farklı platformlarda IBM MQ programları.

Bu görev hakkında

İki örnek kümesi vardır:

- Dağıtılmış sistemler ve IBM iiçin örnek programlar.
- z/OSiçin örnek programlar.

Yordam

- Örnek programlar hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:
  - “Multiplatforms üzerinde örnek programların kullanılması” sayfa 1014
  - z/OS “z/OS için örnek programların kullanılması” sayfa 1111

İlgili kavramlar

“Uygulama geliştirme kavramları” sayfa 6
IBM MQ uygulamaları yazmak için bir yordam ya da nesne yönelimli dil seçeneği kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarını tanıyın.

“IBM MQ için uygulama geliştirilmesi” sayfa 5
İleti göndermek ve almak için uygulamalar geliştirebilir ve kuyruk yöneticilerinizi ve ilgili kaynakları yönetebilirsiniz. IBM MQ , birçok farklı dilde ve çerçevede yazılmış uygulamaları destekler.

“IBM MQ uygulamaları için tasarımla ilgili önemli noktalar” sayfa 47
Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğine karar verdiğinizde, IBM MQtarafından sunulan özellikleri nasıl kullanacağınıza karar vermeniz gerekir.

“Kuyruğa alma için yordam uygulaması yazılması” sayfa 692
Kuyruğa alma uygulamaları yazma, bir kuyruk yöneticisine bağlanma ve bağlantı kesme, yayınlama/abone olma ve nesneleri açma ve kapatma hakkında bilgi edinmek için bu bilgileri kullanın.

“İstemci yordamsal uygulamaları yazılıyor” sayfa 875
IBM MQ üzerinde yordam dili kullanarak istemci uygulamaları yazmak için bilmeniz gerekenleri.

“Yayınlama/abone olma uygulamaları yazılıyor” sayfa 776
Yayınlama/abone olma IBM MQ uygulamaları yazmaya başlayın.

“Yordamsal uygulama oluşturma” sayfa 957
Birkaç yordam dilinden birinde bir IBM MQ uygulaması yazabilir ve uygulamayı birkaç farklı platformda çalıştırabilirsiniz.

“Yordamsal program hatalarının işlenmesi” sayfa 994
Bu bilgiler, bir çağrı yaptığında ya da iletisi son hedefine teslim edildiğinde, uygulamalarınızla ilişkili MQI çağrılarıyla ilgili hataları açıklar.

Multi Multiplatforms üzerinde örnek programların kullanılması

Bu örnek yordam programları ürünle birlikte teslim edilir. Örnekler C ve COBOL dilinde yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir.

Bu görev hakkında

Örneklerin genel programlama tekniklerini göstermesi amaçlanmamıştır, bu nedenle bir üretim programına dahil etmek isteyebileceğiniz bazı hata kontrolleri atlanır.

Tüm örneklerin kaynak kodu ürünle birlikte sağlanır; bu kaynak, programlarda gösterilen ileti kuyruklama tekniklerini açıklayan açıklamalar içerir.

IBM i RPG programlaması için bkz. IBM i Application Programming Reference (ILE/RPG).

Örneklerin adları amqönekiyle başlar. Dördüncü karakter, programlama dilini ve gerektiğinde derleyicisini gösterir:

- s: C dili
- 0: IBM ve Micro Focus derleyicilerinin her ikisinde de COBOL dili
- i: Yalnızca IBM derleyicilerindeki COBOL dili
- m: Yalnızca Micro Focus derleyicilerindeki COBOL dili

Yürütülür dosyanın sekizinci karakteri, örneğin yerel bağ tanımlama kipinde mi, yoksa istemci kipinde mi çalıştığını gösterir. Sekizinci karakter yoksa, örnek yerel bağ tanımlama kipinde çalışır. Sekizinci karakter ' c' ise, örnek istemci kipinde çalışır.

Örnek uygulamaları çalıştırabilmeniz için önce bir kuyruk yöneticisi yaratmanız ve yapılandırmanız gerekir. Kuyruk yöneticisini istemci bağlantılarını kabul edecek şekilde ayarlamak için bkz. “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023.

Yordam

- Örnek programlar hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:
  - “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015
  - “Örnek programların hazırlanması ve çalıştırılması” sayfa 1023
  - “API çıkış örnek programı” sayfa 1029
  - “Zamanuyumsuz tüketim örnek programı” sayfa 1030
  - “Zamanuyumsuz Put örnek programı” sayfa 1031
  - “Göz At örnek programları” sayfa 1032
  - “Tarayıcı örnek programı” sayfa 1033
  - “CICS hareket örneği” sayfa 1034
  - “Connect örnek programı” sayfa 1034
  - “Data-Conversion örnek programı” sayfa 1036
  - “Veritabanı koordinasyonu örnekleri” sayfa 1036
  - “Gönderilmeyen ileti kuyruğu işleyicisi örneği” sayfa 1043
  - “Dağıtım Listesi örnek programı” sayfa 1043
  - “Echo örnek programları” sayfa 1044
  - “"Get" örnek programları” sayfa 1045
  - “Yüksek kullanılabilirlikli örnek programlar” sayfa 1046
  - “Inquire örnek programları” sayfa 1050
  - “Message Handle örnek programının Inquire Properties (İleti Tanıtıcısı Özellikleri)” sayfa 1051
  - “Yayınlama/Abone Olma örnek programları” sayfa 1051

- "Yayınlama Çıkışı örnek programı" sayfa 1056
- "Put örnek programları" sayfa 1057
- "Reference Message örnek programları" sayfa 1059
- "İstek örnek programları" sayfa 1066
- "Set örnek programları" sayfa 1071
- "TLS örnek programı" sayfa 1072
- "Triggering örnek programları" sayfa 1075
- "AIX, Linux, and Windows üzerinde TUXEDO örneklerinin kullanılması" sayfa 1077
- "Windows üzerinde SSPI güvenlik çıkışının kullanılması" sayfa 1086
- "Örneklerin uzak kuyruklar kullanılarak çalıştırılması" sayfa 1087
- "Küme Kuyruğu İzleme örnek programı (AMQSCLM)" sayfa 1087
- "Bağlantı Uç Noktası Arama (CEPL) için örnek program" sayfa 1096

İlgili kavramlar

"C++ örnek programları" sayfa 508
İletileri almayı ve yerleştirmeyi göstermek için dört örnek program sağlanır.

İlgili görevler

"z/OS için örnek programların kullanılması" sayfa 1111
IBM MQ for z/OS ile sağlanan örnek yordam uygulamaları, İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir.

Multi *Multiplatforms üzerinde örnek programlarda gösterilen özellikler*

IBM MQ örnek programları tarafından gösterilen teknikleri gösteren bir dizi çizelge.

MQOPEN ve MQCLOSE çağrılarını kullanan tüm örnekler açık ve kapalı kuyruklar, bu nedenle bu teknikler çizelgelerde ayrı olarak listelenmez. İlgilendiğiniz platformu içeren başlığın başlığına bakın.

z/OS z/OS platformu için bkz. "z/OS için örnek programların kullanılması" sayfa 1111.

Linux AIX *AIX and Linux sistemleri için örnekler*

IBM MQ for AIX or Linuxiçin örnek programlarla gösterilen teknikler.

IBM MQ for AIX or Linux örnek programlarının nerede saklandığını öğrenmek için bkz. "AIX and Linux üzerinde örnek programların hazırlanması ve çalıştırılması" sayfa 1027 .

Çizelge 156 sayfa 1015 Tablo, hangi C ve COBOL kaynak dosyalarının sağlandığı ve bir sunucu ya da istemci yürütülebilir dosyasının dahil edilip edilmediğini listeler.

Çizelge 156. AIX and Linux üzerinde MQI (C ve COBOL) kullanımını gösteren örnek programlar.

Dört sütunlu bir tablo. İlk sütunlar, örneklerle gösterilen teknikleri listeler. İkinci sütunda C örnekleri, üçüncü sütunda ise birinci sütunda listelenen tekniklerin her birini gösteren COBOL örnekleri listelenir. Dördüncü sütun, bir C sunucusu yürütülür dosyasının dahil edilip edilmediğini gösterir ve beşinci sütun, bir C istemcisi yürütülür dosyasının dahil edilip edilmediğini gösterir.

Teknik	C (kaynak) ("1" sayfa 1018)	COBOL (kaynak) ("2" sayfa 1018)	Sunucu (C yürütülür dosyası)	İstemci (C yürütülür dosyası)
Yayınlama/abone olma arabirimini kullanma	amqspuba amqssuba amqssbxa	örnek yok	amqspub amqssub amqssbx	örnek yok
İletilerin MQPUT çağrısı kullanılarak konmasını sağlar	amqsput0	amq0put0	amqsput	amqsputc

Çizelge 156. AIX and Linux üzerinde MQI (C ve COBOL) kullanımını gösteren örnek programlar.

Dört sütunlu bir tablo. İlk sütunlar, örneklerle gösterilen teknikleri listeler. İkinci sütunda C örnekleri, üçüncü sütunda ise birinci sütunda listelenen tekniklerin her birini gösteren COBOL örnekleri listelenir. Dördüncü sütun, bir C sunucusu yürütülür dosyasının dahil edilip edilmediğini gösterir ve beşinci sütun, bir C istemcisi yürütülür dosyasının dahil edilip edilmediğini gösterir.

(devamı var)

Teknik	**C (kaynak) (“1” sayfa 1018)**	**COBOL (kaynak) (“2” sayfa 1018)**	**Sunucu (C yürütülür dosyası)**	**İstemci (C yürütülür dosyası)**
MQPUT1 çağrısı kullanılarak tek bir ileti konması	amqsinqa amqsecha	amqminqx amqmechx amqiinqx amqiechx	amqsinq amqsech	amqsechc
İletilerin dağıtım listesine konması (“3” sayfa 1018)	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
İstek iletisinin yanıtlanması	amqsinqa	amqminqx amqiinqx	amqsinq	örnek yok
Göz atma kullanılarak ileti alma (bekleme yok)	amqsgbr0	amq0gbr0	amqsgbr	örnek yok
İleti alma (süre sınırı ile bekleme)	amqsget0	amq0get0	amqsget	amqsgetc
İletiler alınıyor (sınırsız bekleme)	amqstrg0	örnek yok	amqstrg	amqstrgc
İleti alma (veri dönüştürme ile)	amqsecha	örnek yok	amqsech	örnek yok
Başvuru İletilerinin Kuyruğa Konması (“3” sayfa 1018)	amqsprma	örnek yok	amqsprm	amqsprmc
Kuyruktan Başvuru İletileri Alma (“3” sayfa 1018)	amqsgrma	örnek yok	amqsgrm	amqsgrmc
Başvuru İletisi kanal çıkışı (“3” sayfa 1018)	amqsqrma amqsxrma	örnek yok	amqsxrm	örnek yok
İletinin ilk 20 karakterine göz atma	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Tam iletilere göz atma	amqsbcg0	örnek yok	amqsbcg	amqsbcgc
Paylaşılan giriş kuyruğunun kullanılması	amqsinqa	amqminqx amqiinqx	amqsinq	amqsinqc
Dışlayıcı giriş kuyruğunun kullanılması	amqstrg0	amq0req0	amqstrg	amqstrgc
MQINQ çağrıyı kullanma	amqsinqa	amqminqx amqiinqx	amqsinq	örnek yok
MQSET çağrıyı kullanma	amqsseta	amqmsetx amqisetx	amqsset	amqssetc
Yanıt Kuyruğunun Kullanılması	amqsreq0	amq0req0	amqsreq	amqsreqc
İleti kural dışı durumları isteniyor	amqsreq0	amq0req0	amqsreq	örnek yok
Kesilmiş bir iletiyi kabul etme	amqsgbr0	amq0gbr0	amqsgbr	örnek yok
Çözülmüş kuyruk adının kullanılması	amqsgbr0	amq0gbr0	amqsgbr	örnek yok
Süreci tetikleme	amqstrg0	örnek yok	amqstrg	amqstrgc

Çizelge 156. AIX and Linux üzerinde MQI (C ve COBOL) kullanımını gösteren örnek programlar.

Dört sütunlu bir tablo. İlk sütunlar, örneklerle gösterilen teknikleri listeler. İkinci sütunda C örnekleri, üçüncü sütunda ise birinci sütunda listelenen tekniklerin her birini gösteren COBOL örnekleri listelenir. Dördüncü sütun, bir C sunucusu yürütülür dosyasının dahil edilip edilmediğini gösterir ve beşinci sütun, bir C istemcisi yürütülür dosyasının dahil edilip edilmediğini gösterir.

(devamı var)

Teknik	**C (kaynak) (“1” sayfa 1018)**	**COBOL (kaynak) (“2” sayfa 1018)**	**Sunucu (C yürütülür dosyası)**	**İstemci (C yürütülür dosyası)**
Veri dönüştürmenin kullanılması	(“4” sayfa 1018)	örnek yok	örnek yok	örnek yok
IBM MQ (XA uyumlu veritabanı yöneticilerinin eşgüdümü) SQL kullanarak tek bir veritabanına erişme	amqsxas0.sqc Db2 amqsxas0.ec Informix	amq0xas0.sqb	örnek yok	örnek yok
IBM MQ (XA uyumlu veritabanı yöneticilerinin eşgüdümü) SQL kullanarak iki veritabanına erişilmesi	amqsxag0.c amqsxab0.sqc amqsxaf0.sqc	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	örnek yok	örnek yok
CICS hareket (“5” sayfa 1018)	amqscic0.ccs	örnek yok	amqscic0	örnek yok
Encina işlemi (“3” sayfa 1018)	amqsxae0	örnek yok	amqsxae0	örnek yok
İletileri koymak için TUXEDO işlemi “6” sayfa 1018)	amqstxpx	örnek yok	örnek yok	örnek yok
İleti almak için TUXEDO işlemi (“6” sayfa 1018)	amqstxgx	örnek yok	örnek yok	örnek yok
TUXEDO Sunucusu (“6” sayfa 1018)	amqstxsx	örnek yok	örnek yok	örnek yok
Gönderilmeyen ileti kuyruğu işleyicisi	Dizin `./tools/c/Samples/dlq` (“7” sayfa 1018)	örnek yok	amqsdlq	örnek yok
MQI istemcisinden ileti koyma	örnek yok	örnek yok	örnek yok	amqsputc
MQI istemcisinden ileti alma	örnek yok	örnek yok	örnek yok	amqsgetc
MQCONNX kullanılarak kuyruk yöneticisine bağlanılıyor	amqscnxc	örnek yok	örnek yok	amqscnxc
API çıkışlarını kullanma	amqsaxe0	örnek yok	amqsaxe	örnek yok
Küme iş yükü dengeleme çıkışı	amqswlm0	örnek yok	amqswlm	örnek yok
İletileri zamanuyumsuz olarak koyma ve MQSTAT çağrısı kullanılarak durum alma	amqsapt0	örnek yok	amqsapt	amqsaptc
Yeniden bağlanabilir istemciler	amqsphac amqsghac amqsmhac	örnek yok	geçerli değil	amqsphac amqsghac amqsmhac

Çizelge 156. AIX and Linux üzerinde MQI (C ve COBOL) kullanımını gösteren örnek programlar.

Dört sütunlu bir tablo. İlk sütunlar, örneklerle gösterilen teknikleri listeler. İkinci sütunda C örnekleri, üçüncü sütunda ise birinci sütunda listelenen tekniklerin her birini gösteren COBOL örnekleri listelenir. Dördüncü sütun, bir C sunucusu yürütülür dosyasının dahil edilip edilmediğini gösterir ve beşinci sütun, bir C istemcisi yürütülür dosyasının dahil edilip edilmediğini gösterir.

(devamı var)

Teknik	C (kaynak) (“1” sayfa 1018)	COBOL (kaynak) (“2” sayfa 1018)	Sunucu (C yürütülür dosyası)	İstemci (C yürütülür dosyası)
Birden çok kuyruktan gelen iletileri zamanuyumsuz olarak tüketmek için ileti tüketicilerinin kullanılması	amqscbf0	örnek yok	amqscbf	amqscbfc
MQCONNX 'te TLS bağlantı bilgilerinin belirtilmesi	amqssslc	örnek yok	geçerli değil	amqssslc

Notlar:

1. IBM MQ MQI client örneklerinin yürütülebilir sürümü, bir sunucu ortamında çalışan örneklerle aynı kaynağı paylaşır.
2. Micro Focus COBOL derleyicisiyle 'amqm' ile başlayan programları, IBM COBOL derleyicisiyle 'amqi' ile başlayan programları ve 'amq0' ile başlayan programları derleyin.
3. AIX Yalnızca IBM MQ for AIX üzerinde desteklenir.
4. AIX IBM MQ for AIX sistemlerinde bu program `amqsvfc0.c` olarak adlandırılır.
5. AIX CICS yalnızca IBM MQ for AIX tarafından desteklenir.
6. Linux TUXEDO, System püzerinde Linux için IBM MQ tarafından desteklenmez.
7. Gönderilmeyen ileti kuyruğu işleyicisinin kaynağı birkaç dosyadan oluşur ve ayrı bir dizinde sağlanır.

AIX and Linux sistemleri desteğiyle ilgili ayrıntılı bilgi için bkz. IBM MQ için Sistem Gereksinimleri.

Windows *IBM MQ for Windows için örnekler*

IBM MQ for Windowsiçin örnek programlarla gösterilen teknikler.

Çizelge 157 sayfa 1018 içinde hangi C ve COBOL kaynak dosyalarının sağlandığı ve bir sunucu ya da istemci yürütülebilir dosyasının dahil edilip edilmediği listelenir.

Çizelge 157. MQI (C ve COBOL) kullanımını gösteren IBM MQ for Windows örnek programları

Teknik	C (kaynak)	COBOL (kaynak)	Sunucu (C yürütülür dosyası)	İstemci (C yürütülür dosyası)
Yayınlama/abone olma arabirimini kullanma	amqspuba amqssuba amqssbxa	örnek yok	amqspub amqssub amqssbx	örnek yok
İletilerin MQPUT çağrısı kullanılarak konmasını sağlar	amqsput0	amq0put0	amqsput	amqsputc
MQPUT1 çağrısı kullanılarak tek bir ileti konması	amqsinqa amqsecha	amqminq2 amqmech2 amqiinq2 amqiech2	amqsinq amqsech	amqsinqc amqsechc
Dağıtım listesine ileti konması	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc

Çizelge 157. MQI (C ve COBOL) kullanımını gösteren IBM MQ for Windows örnek programları (devamı var)

Teknik	**C (kaynak)**	**COBOL (kaynak)**	**Sunucu (C yürütülür dosyası)**	**İstemci (C yürütülür dosyası)**
İstek iletisinin yanıtlanması	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
İletiler alınıyor (bekleme yok)	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
İleti alma (süre sınırı ile bekleme)	amqsget0	amq0get0	amqsget	amqsgetc
İletiler alınıyor (sınırsız bekleme)	amqstrg0	örnek yok	amqstrg	amqstrgc
İleti alma (veri dönüştürme ile)	amqsecha	örnek yok	amqsech	amqsechc
Başvuru İletilerinin Kuyruğa Konması	amqsprma	örnek yok	amqsprm	amqsprmc
Kuyruktan Başvuru İletileri Alma	amqsgrma	örnek yok	amqsgrm	amqsgrmc
Başvuru İletisi kanal çıkışı	amqsqrma amqsxrma	örnek yok	amqsxrm	örnek yok
İletinin ilk 20 karakterine göz atma	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Tam iletilere göz atma	amqsbcg0	örnek yok	amqsbcg	amqsbcgc
Paylaşılan giriş kuyruğunun kullanılması	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Dışlayıcı giriş kuyruğunun kullanılması	amqstrg0	amq0req0	amqstrg	amqstrgc
MQINQ çağrıyı kullanma	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
MQSET çağrıyı kullanma	amqsseta	amqmset2 amqiset2	amqsset	amqssetc
MQINQMP çağrıyı kullanma	amqsiqma	örnek yok	örnek yok	örnek yok
Yanıt Kuyruğunun Kullanılması	amqsreq0	amq0req0	amqsreq	amqsreqc
İleti kural dışı durumları isteniyor	amqsreq0	amq0req0	amqsreq	amqsreqc
Kesilmiş bir iletiyi kabul etme	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Çözülmüş kuyruk adının kullanılması	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Süreci tetikleme	amqstrg0	örnek yok	amqstrg	amqstrgc
Veri dönüştürmenin kullanılması	amqsvfc0	örnek yok	örnek yok	örnek yok
IBM MQ (XA uyumlu veritabanı yöneticilerinin eşgüdümü) SQL kullanarak tek bir veritabanına erişme	amqsxas0.sqc Db2 amqsxas0.ec Informix	amq0xas0.sqb	örnek yok	örnek yok
IBM MQ (XA uyumlu veritabanı yöneticilerinin eşgüdümü) SQL kullanarak iki veritabanına erişilmesi	amqsxag0.c amqsxab0.sqc Db2 amqsxaf0.sqc Db2	amq0xag0.cbl amq0xab0.sqb amq0xaf0.sqb	örnek yok	örnek yok

Çizelge 157. MQI (C ve COBOL) kullanımını gösteren IBM MQ for Windows örnek programları (devamı var)				
Teknik	**C (kaynak)**	**COBOL (kaynak)**	**Sunucu (C yürütülür dosyası)**	**İstemci (C yürütülür dosyası)**
İletileri koymak için TUXEDO işlemi	amqstxpx	örnek yok	örnek yok	örnek yok
İletileri almak için TUXEDO işlemi	amqstxgx	örnek yok	örnek yok	örnek yok
TUXEDO Sunucusu	amqstxsx	örnek yok	örnek yok	örnek yok
Gönderilmeyen ileti kuyruğu işleyicisi	Dizin `./tools/c/Samples/dlq` (“1” sayfa 1020)	örnek yok	amqsdlq	örnek yok
IBM MQ MQI client' den bir ileti koyma	örnek yok	örnek yok	örnek yok	amqsputc
IBM MQ MQI client' den ileti alma	örnek yok	örnek yok	örnek yok	amqsgetc
MQCONNX kullanılarak kuyruk yöneticisine bağlanılıyor	amqscnxc	örnek yok	örnek yok	amqscnxc
API çıkışlarını kullanma	amqsaxe0	örnek yok	amqsaxe	örnek yok
Küme iş yükü dengelemesi	amqswlm0	örnek yok	amqswlm	örnek yok
SSPI güvenlik yordamları	amqsspin	örnek yok	amqrspin.dll	amqrspin.dll
İletileri zamanuyumsuz olarak koyma ve MQSTAT çağrısı kullanılarak durum alma	amqsapt0	örnek yok	amqsapt	amqsaptc
Yeniden bağlanabilir istemciler	amqsphac amqsghac amqsmhac	örnek yok	Geçerli değildir	amqsphac amqsghac amqsmhac
Birden çok kuyruktan gelen iletileri zamanuyumsuz olarak tüketmek için ileti tüketicilerinin kullanılması	amqscbf0	örnek yok	amqscbf	amqscbfc
MQCONNX 'te TLS bağlantı bilgilerinin belirtilmesi	amqssslc	örnek yok	geçerli değil	amqssslc

Notlar:

1. Gönderilmeyen ileti kuyruğu işleyicisinin kaynağı birkaç dosyadan oluşur ve ayrı bir dizinde sağlanır.

Windows *IBM MQ for Windows için Visual Basic örnekleri*

Windows sistemlerinde IBM MQ için örnek programlarla gösterilen teknikler.

Çizelge 158 sayfa 1021 içinde IBM MQ for Windows örnek programları tarafından gösterilen teknikler gösterilmektedir.

Bir proje birkaç dosya içerebilir. Bir projeyi Visual Basic içinde açtığınızda, diğer dosyalar otomatik olarak yüklenir. Yürütülebilir program sağlanmaz.

mqtrivc.vbpdışında tüm örnek projeler, IBM MQ sunucusuyla çalışacak şekilde ayarlanır. IBM MQ istemcileriyle çalışmak üzere örnek projelerin nasıl değiştirileceğini öğrenmek için bkz. “Windows ' da Visual Basic programlarının hazırlanması” sayfa 977.

Çizelge 158. MQI kullanımını gösteren IBM MQ for Windows örnek programları (Visual Basic)	
Teknik	**Proje dosyası adı**
İletilerin MQPUT çağrısı kullanılarak konmasını sağlar	amqsputb.vbp
MQGET çağrıyı kullanarak iletileri alma	amqsgetb.vbp
MQGET çağrısını kullanarak kuyruğa göz atma	amqsbcgb.vbp
Yalın MQGET ve MQPUT örneği (istemci)	mqtrivc.vbp
Yalın MQGET ve MQPUT örneği (sunucu)	mqtrivs.vbp
MQPUT ve MQGET kullanarak dizgileri ve kullanıcı tanımlı yapıları koyma ve alma	strings.vbp
Bir kanalı başlatmak ve durdurmak için PCF yapılarının kullanılması	pcfsamp.vbp
MQAI kullanılarak kuyruk yaratılması	amqsaicq.vbp
MQAI kullanılarak bir kuyruk yöneticisinin kuyruklarının listelenmesi	amqsailq.vbp
MQAI kullanılarak olayların izlenmesi	amqsaiem.vbp

IBM i *IBM i için örnekler*

IBM i sistemlerinde IBM MQ için örnek programlarla gösterilen teknikler.

Çizelge 159 sayfa 1021 içinde IBM MQ for IBM i örnek programları tarafından gösterilen teknikler gösterilmektedir. Bazı teknikler birden çok örnek programda ortaya çıkar, ancak çizelgede yalnızca bir program listelenir.

Çizelge 159. IBM i üzerinde MQI (C ve COBOL) kullanımını gösteren örnek programlar				
Teknik	**C (kaynak) ("1" sayfa 1023)**	**COBOL (kaynak) ("2" sayfa 1023)**	**RPG (kaynak) ("3" sayfa 1023)**	**İstemci (C yürütülür dosyası) (4)**
İletilerin MQPUT çağrısı kullanılarak konmasını sağlar	AMQSPUT0	AMQ0PUT4	AMQ3PUT4	AMQSPUTC
MQPUT çağrısıyla bir veri dosyasından ileti konması	AMQSPUT4	örnek yok	örnek yok	örnek yok
MQPUT1 çağrısı kullanılarak tek bir ileti konması	AMQSINQ4, AMQSECH4	AMQ0INQ4, AMQ0ECH4	AMQ3INQ4, AMQ3ECH4	AMQSINQC, AMQSECHC
Dağıtım listesine ileti konması	AMQSPTL4	örnek yok	örnek yok	AMQSPTLC
İstek iletisinin yanıtlanması	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
İletiler alınıyor (bekleme yok)	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
İleti alma (süre sınırı ile bekleme)	AMQSGET4	AMQ0GET4	AMQ3GET4	AMQSGETC
İletiler alınıyor (sınırsız bekleme)	AMQSTRG4	örnek yok	AMQ3TRG4	AMQSTRGC
İleti alma (veri dönüştürme ile)	AMQSECH4	AMQ0ECH4	AMQ3ECH4	AMQSECHC
Başvuru İletilerinin Kuyruğa Konması	AMQSPRM4	örnek yok	örnek yok	AMQSPRMC
Kuyruktan Başvuru İletileri Alma	AMQSGRM4	örnek yok	örnek yok	AMQSGRMC
Başvuru İletisi kanal çıkışı	AMQSQRM4, AMQSXRM4	örnek yok	örnek yok	Örnek Yok
İleti çıkışı	AMQSCMX4	örnek yok	örnek yok	Örnek Yok

Çizelge 159. IBM i üzerinde MQI (C ve COBOL) kullanımını gösteren örnek programlar (devamı var)

Teknik	**C (kaynak) ("1" sayfa 1023)**	**COBOL (kaynak) ("2" sayfa 1023)**	**RPG (kaynak) ("3" sayfa 1023)**	**İstemci (C yürütülür dosyası) (4)**
İletinin ilk 49 karakterine göz atma	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Tam iletilere göz atma	AMQSBCG4	örnek yok	örnek yok	AMQSBCGC
Paylaşılan giriş kuyruğunun kullanılması	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
Dışlayıcı giriş kuyruğunun kullanılması	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQC
MQINQ çağrıyı kullanma	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
MQSET çağrıyı kullanma	AMQSSET4	AMQ0SET4	AMQ3SET4	AMQSSETC
Yanıt Kuyruğunun Kullanılması	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQC
İleti kural dışı durumları isteniyor	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQC
Kesilmiş bir iletiyi kabul etme	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Çözülmüş kuyruk adının kullanılması	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Süreci tetikleme	AMQSTRG4	örnek yok	AMQ3TRG4	AMQSTRGC
Tetikleyici sunucu	AMQSERV4	örnek yok	AMQ3SRV4	örnek yok
Tetikleyici sunucusunun kullanılması (CICS hareketleri de içinde olmak üzere)	AMQSERV4	örnek yok	AMQ3SRV4	örnek yok
Veri dönüştürmenin kullanılması	AMQSVFC4	örnek yok	örnek yok	örnek yok
API çıkışlarını kullanma	AMQSAXE0	örnek yok	örnek yok	örnek yok
Küme iş yükü dengelemesi	AMQSWLM0	örnek yok	örnek yok	örnek yok
İletileri zamanuyumsuz olarak koyma ve MQSTAT çağrısı kullanılarak durum alma	AMQSAPT0	örnek yok	örnek yok	AMQSAPTC
Yayınlama/abone olma arabirimini kullanma	AMQSPUBA, AMQSSUBA, AMQSSBXA	örnek yok	örnek yok	AMQSPUBC, AMQSSUBC, AMQSSBXC
Yeniden bağlanabilir istemciler (5)	AMQSPHAC, AMQSGHAC, AMQSMHAC	örnek yok	örnek yok	örnek yok
Birden çok kuyruktan gelen iletileri zamanuyumsuz olarak tüketmek için ileti tüketicilerinin kullanılması (5)	AMQSCBFO	örnek yok	örnek yok	örnek yok
MQCONNX 'te TLS bağlantı bilgilerinin belirtilmesi	AMQSSSLC	örnek yok	örnek yok	AMQSSSLC
MQCONNX kullanılarak kuyruk yöneticisine bağlanılıyor	AMQSCNXC	örnek yok	örnek yok	AMQSCNXC
İleti kuyruğundan MQINQMP kullanarak bir ileti tanıtıcısının özelliklerini sorma	AMQISQMA	örnek yok	örnek yok	AMQISQMC
MQSETMP kullanarak bir ileti tanıtıcısının özelliklerini ayarlayın ve ileti kuyruğuna koyun	AMQSSQMA	örnek yok	örnek yok	AMQSSQMC

Notlar:

1. C örneklerine ilişkin kaynak QMQMSAMP/QCSRC dosyasında. İçerilecek dosyalar QMQM/H dosyasında üye olarak var.
2. COBOL örneklerinin kaynağı QMQMSAMP/QCBLLESRC dosyalarında bulunur. Üyeler AMQ0 xxx 4 olarak adlandırılır; burada xxx örnek işlevi gösterir.
3. RPG örneklerinin kaynağı QMQMSAMP/QRPGLESRC 'dir. Üyeler AMQ3 xxx 4 olarak adlandırılır; burada xxx örnek işlevi gösterir. Kopya üyeleri QMQM/QRPGLESRC içinde var. Her üye adı G sonekine sahiptir.
4. IBM MQ MQI client örneklerinin yürütülebilir sürümü, bir sunucu ortamında çalışan örneklerle aynı kaynağı paylaşır. İstemci ortamındaki örnekler için kaynak, sunucuyla aynı. IBM MQ MQI client örnekleri, istemci kitaplığı LIBMQIC ile bağlantılıdır ve IBM MQ sunucu örnekleri, sunucu kitaplığı LIBMQM ile bağlantılıdır.
5. Yeniden Yapılabilir istemci ve zamanuyumsuz tüketici uygulamasının örnek uygulaması için istemci yürütülür dosyasının çalıştırılması gerekirse, derlenmesi ve LIBMQIC_R kitaplığı ile bağlantılandırılması gerekir. Bu nedenle, iş parçacıklı ortamda çalıştırılması gerekir. QIBM_MULTI_THREADED ortam değişkenini 'Y' olarak ayarlayın ve uygulamayı qsh içinden çalıştırın.

   Daha fazla bilgi için bkz. IBM MQ ürününü Java ve JMS ile ayarlama .

Ek bilgi için bkz. “IBM i üzerinde örnek programların hazırlanması ve çalıştırılması” sayfa 1025 .

Bunlara ek olarak, IBM MQ for IBM i örnek seçeneği örnek programlara, AMQSDATA ' ya ve yönetim görevlerini gösteren örnek CL programlarına giriş olarak kullandığınız bir örnek veri dosyası içerir. CL örnekleri, Yönetme IBM i içinde açıklanır. Bu konuda açıklanan örnek programlarla kullanmak üzere kuyruklar yaratmak için örnek CL programını amqsamp4 kullanabilirsiniz.

Multi

Örnek programların hazırlanması ve çalıştırılması

Bazı ilk hazırlıkları tamamladıktan sonra örnek programları çalıştırabilirsiniz.

Bu görev hakkında

Örnek programları çalıştırmadan önce, önce bir kuyruk yöneticisi yaratmanız ve gereksinim duyduğunuz kuyrukları yaratmanız gerekir. Örneğin, COBOL örneklerini çalıştırmak istiyorsanız, bazı ek hazırlıklar da yapmanız gerekebilir. Gerekli hazırlığı tamamladıktan sonra örnek programları çalıştırabilirsiniz.

Yordam

Örnek programları hazırlama ve çalıştırma hakkında bilgi için aşağıdaki konulara bakın:

- “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023
- “IBM i üzerinde örnek programların hazırlanması ve çalıştırılması” sayfa 1025
- “AIX and Linux üzerinde örnek programların hazırlanması ve çalıştırılması” sayfa 1027
- “Windows üzerinde örnek programların hazırlanması ve çalıştırılması” sayfa 1028

Multi

Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması

Örnek uygulamaları çalıştırabilmeniz için önce bir kuyruk yöneticisi yaratmanız gerekir. Bundan sonra, kuyruk yöneticisini istemci kipinde çalışan uygulamalardan gelen bağlantı isteklerini güvenli bir şekilde kabul edecek şekilde yapılandırabilirsiniz.

Başlamadan önce

Kuyruk yöneticisinin önceden var olduğunu ve başlatıldığını doğrulayın. MQSC komutunu vererek kanal kimlik doğrulama kayıtlarının önceden etkinleştirilip etkinleştirilmediğini saptayın:

```
DISPLAY QMGR CHLAUTH
```

Önemli: Bu görev, kanal kimlik doğrulama kayıtlarının etkinleştirilmesini bekler. Bu, diğer kullanıcılar ve uygulamalar tarafından kullanılan bir kuyruk yöneticisiyse, bu ayarın değiştirilmesi diğer tüm kullanıcıları ve uygulamaları etkiler. Kuyruk yöneticiniz kanal kimlik doğrulama kayıtlarından yararlanmazsa, 4 adımı, MCAUSER ' i “1” sayfa 1024. adımda edineceğiniz *ayrıcalıklı olmayan kullanıcı kimliği* olarak ayarlayan alternatif bir kimlik doğrulama yöntemiyle (örneğin, bir güvenlik çıkışı) değiştirilebilir.

Uygulamanın kanalı kullanmasına izin verilebilmesi için, uygulamanızın hangi kanal adını kullanmayı beklediğini bilmeniz gerekir. Uygulamanızın bunları kullanmasına izin verilebilmesi için, uygulamanızın hangi nesneleri (örneğin, kuyruklar ya da konular) kullanmayı beklediğini de bilmeniz gerekir.

Bu görev hakkında

Bu görev, kuyruk yöneticisine bağlanan bir istemci uygulaması için kullanılacak ayrıcalıklı olmayan bir kullanıcı kimliği yaratır. İstemci uygulamasına yalnızca bu kullanıcı kimliğini kullanarak gereksinim duyduğu kanalı ve kuyruğu kullanabilmesi için erişim verilir.

Yordam

1. Kuyruk yöneticinizin çalıştığı sistemde bir kullanıcı kimliği edinin. Bu görev için bu kullanıcı kimliği ayrıcalıklı bir yönetici kullanıcı olmamalıdır. Bu kullanıcı kimliği, istemci bağlantısının kuyruk yöneticisinde çalışacağı yetki olur.
2. Aşağıdaki komutlarla bir dinleyici programı başlatın:

   qmgr-name , kuyruk yöneticinizin adıdır
   nnnn , seçtiğiniz kapı numarasıdır

   a) ALW
   AIX, Linux, and Windows sistemleri için:

   ```
   runmqlsr -t tcp -m qmgr-name -p nnnn
   ```

   b) IBM i
   IBM iiçin:

   ```
   STRMQMLSR MQMNAME(qmgr-name) PORT(nnnn)
   ```

3. Uygulamanız SYSTEM.DEF.SVRCONN , daha sonra bu kanal önceden tanımlanmıştır. Uygulamanız başka bir kanal kullanıyorsa, aşağıdaki MQSC komutunu vererek yaratın:

   ```
   DEFINE CHANNEL(' channel-name ') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

   Burada *kanal-adı* , kanalın adıdır.
4. Aşağıdaki MQSC komutunu vererek, istemci sisteminizin yalnızca IP adresinin kanalı kullanmasını sağlayacak bir kanal kimlik doğrulama kuralı yaratın:

   ```
   SET CHLAUTH(' channel-name ') TYPE(ADDRESSMAP) ADDRESS(' client-machine-IP-address ') +
   MCAUSER(' non-privileged-user-id ')
   ```

   burada:

   kanal-adı , kanalınızın adıdır.
   istemci-makine-IP-adresi , istemci sisteminizin IP adresidir. Örnek istemci uygulamanız kuyruk yöneticisiyle aynı makinede çalışıyorsa, uygulamanız 'localhost' kullanarak bağlanacaksa '127.0.0.1' IP adresini kullanın. Birkaç farklı istemci makinesi bağlanacaksa, tek bir IP adresi yerine bir kalıp ya da aralık kullanabilirsiniz. Ayrıntılar için bkz. Soysal IP adresleri .
   ayrıcalıklı olmayan kullanıcı kimliği , adım “1” sayfa 1024 ' de edindiğiniz kullanıcı kimliğidir.

5. Uygulamanız SYSTEM.DEFAULT.LOCAL.QUEUE ise, bu kuyruk önceden tanımlanmıştır. Uygulamanız başka bir kuyruk kullanıyorsa, aşağıdaki MQSC komutunu vererek yaratın:

   ```
   DEFINE QLOCAL(' queue-name ') DESCR('Queue for use by sample programs')
   ```

   Burada *kuyruk-adı* , kuyruğunuzun adıdır.
6. Aşağıdaki MQSC komutunu vererek kuyruk yöneticisine bağlanma ve bu yöneticiye sorma erişimi verin:

   ```
   SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non-privileged-user-id ') +
   AUTHADD(CONNECT, INQ)
   ```

   Burada *ayrıcalıklı olmayan kullanıcı kimliği* , “1” sayfa 1024 . adımda edindiğiniz kullanıcı kimliğidir.
7. Uygulamanız noktadan noktaya iletişim uygulamasıysa, aşağıdaki MQSC komutlarını kullanarak, kuyruklardan yararlanır, sorgunun yapılmasına ve kuyruğunuzu kullanan iletilerin kullanıcı kimliğine göre konmasına ve alınmasına izin verir:

   ```
   SET AUTHREC PROFILE(' queue-name ') OBJTYPE(QUEUE) +
   PRINCIPAL(' non-privileged-user-id ') AUTHADD(PUT, GET, INQ, BROWSE)
   ```

   burada:

   kuyruk-adı , kuyruğunuzun adıdır
   ayrıcalıklı olmayan kullanıcı kimliği , adım “1” sayfa 1024 ' de edindiğiniz kullanıcı kimliğidir.
8. Uygulamanız bir yayınlama/abone olma uygulamasıysa, MQSC komutlarını kullanarak, konuları kullanır, kullanılacak kullanıcı kimliğiyle konunuzu kullanarak yayınlama ve abone olma izni verir:

   ```
   SET AUTHREC PROFILE('SYSTEM.BASE.TOPIC') OBJTYPE(TOPIC) +
   PRINCIPAL(' non-privileged-user-id ') AUTHADD(PUB, SUB)
   ```

   burada:

   ayrıcalıklı olmayan kullanıcı kimliği , adım “1” sayfa 1024 ' de edindiğiniz kullanıcı kimliğidir.
   Bu, *ayrıcalıklı olmayan kullanıcı kimliği* ' ne konu ağacındaki herhangi bir konuya ilişkin erişim verir; diğer bir seçenek olarak, **DEFINE TOPIC** komutunu kullanarak bir konu nesnesi tanımlayabilir ve yalnızca o konu nesnesinin başvurduğu konu ağacının kısmına erişim verebilirsiniz. Ayrıntılar için Konuların kullanıcı erişimini denetleme başlıklı konuya bakın.

Sonraki adım

İstemci uygulamanız artık kuyruk yöneticisine bağlanabilir ve kuyruğu kullanarak ileti koyabilir ya da alabilir.

İlgili kavramlar

ALW AIX, Linux, and Windows üzerinde bir IBM MQ nesnesine erişim verilmesi

İlgili başvurular

CHLAUTH AYARLA
KANAL TANıMLAYıN
QLOCAL TANıMLAYıN
AUTHREC DEğERINI AYARLA

IBM i IBM i üzerindeki IBM MQ yetkileri

IBM i *IBM i üzerinde örnek programların hazırlanması ve çalıştırılması*

Örnek programları IBM iüzerinde çalıştırmadan önce, önce bir kuyruk yöneticisi yaratmanız ve ayrıca gerek duyduğunuz kuyrukları yaratmanız gerekir. COBOL örneklerini çalıştırmak istiyorsanız, ek hazırlık yapmanız gerekebilir.

Bu görev hakkında

IBM MQ for IBM i örnek programlarının kaynağı QMQMSAMP kitaplığında QCSRC, QCLSRC, QCBLLESRC ve QRPGLESRC üyeleri olarak sağlanır.

Örnekleri çalıştırırken kendi kuyruklarınızı kullanabilir ya da örnek kuyruklar yaratmak için AMQSAMP4 örnek programını çalıştırabilirsiniz. AMQSAMP4 programının kaynağı, QMQMSAMP kitaplığındaki QCLSRC dosyasında bulunur. CRTCLPGM komutunu kullanarak derleyebilirsiniz.

Örnekleri çalıştırmak için, QMQM kitaplığında verilen C yürütülebilir sürümlerini kullanın ya da bunları başka bir IBM MQ uygulamasına benzer şekilde derleyin.

Yordam

1. Bir kuyruk yöneticisi yaratın ve varsayılan tanımlamaları ayarlayın.

   Örnek programlardan herhangi birini çalıştırmadan önce bunu yapmanız gerekir. Kuyruk yöneticisi oluşturma hakkında daha fazla bilgi için bkz. Yönetme IBM MQ. Bir kuyruk yöneticisinin istemci kipinde çalışan uygulamalardan gelen bağlantı isteklerini güvenli bir şekilde kabul edecek şekilde yapılandırılmasıyla ilgili bilgi için bkz. “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023.
2. QMQMSAMP kitaplığının AMQSDATA kütüğündeki PUT üyesinden alınan verileri kullanarak örnek programlardan birini çağırmak için aşağıdaki gibi bir komut kullanın:

   ```
   CALL PGM(QMQM/AMQSPUT4) PARM('QMQMSAMP/AMQSDATA(PUT)')
   ```

   Not: Derlenmiş bir modülün IFS dosya sistemini kullanması için, CRTCMOD üzerinde SYSIFCOPT (*IFSIO) seçeneğini belirleyin, ardından parametre olarak geçirilen dosya adı aşağıdaki biçimde belirlenmelidir:

   ```
   home/me/myfile
   ```
3. Inquire, Set ve Echo örneklerinin COBOL sürümlerini kullanmak istiyorsanız, bu örnekleri çalıştırmadan önce süreç tanımlamalarını değiştirin.

   Inquire, Set ve Echo örnekleri için, örnek tanımlamaları bu örneklerin C sürümlerini tetikler. COBOL sürümlerini istiyorsanız, süreç tanımlamalarını değiştirmeniz gerekir:

   - SYSTEM.SAMPLE.INQPROCESS
   - SYSTEM.SAMPLE.SETPROCESS
   - SYSTEM.SAMPLE.ECHOPROCESS

   IBM iişletim sistemlerinde, **CHGMQMPRC** komutunu kullanabilirsiniz (ayrıntılar için MQ İşleminin Değiştirilmesi (CHGMQMPRC) konusuna bakın) ya da **AMQSAMP4** komutunu diğer tanımlamayla düzenleyebilir ve çalıştırabilirsiniz.
4. Örnek programları çalıştırın.

   Her bir örneğin beklediği parametrelerle ilgili daha fazla bilgi için tek tek örneklerin açıklamalarına bakın.

   Not: COBOL örnek programları için, kuyruk adlarını parametre olarak geçirdiğinizde, gerekirse boş karakterlerle doldurarak 48 karakter sağlamanız gerekir. 48 karakterden başka bir karakter, programın başarısız olmasına neden olur. Neden kodu 2085.

İlgili başvurular

“IBM i için örnekler” sayfa 1021
IBM i sistemlerinde IBM MQ için örnek programlarla gösterilen teknikler.

Linux AIX *AIX and Linux üzerinde örnek programların hazırlanması ve çalıştırılması*

Örnek programları AIX and Linuxüzerinde çalıştırmadan önce, önce bir kuyruk yöneticisi yaratmanız ve ayrıca gerek duyduğunuz kuyrukları yaratmanız gerekir. COBOL örneklerini çalıştırmak istiyorsanız, ek hazırlık yapmanız gerekebilir.

Bu görev hakkında

IBM MQ on AIX and Linux sistemleri örnek dosyaları, kuruluş sırasında varsayılan değerler kullanıldıysa, Çizelge 160 sayfa 1027 içinde listelenen dizinlerdedir.

Çizelge 160. AIX and Linux sistemlerinde IBM MQ örnekleri nerede bulunur?

İçindekiler	**Dizin**
Kaynak dosyalar	*MQ_INSTALLATION_PATH*/samp
çıkmaz kuyruk işleyicisi kaynak dosyaları	*MQ_INSTALLATION_PATH*/samp/dlq
yürütülebilir dosyalar	*MQ_INSTALLATION_PATH*/samp/bin

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Örneklerle çalışmak için bir kuyruk kümesi gerekir. Kendi kuyruklarınızı kullanabilir ya da küme yaratmak için amqscos0.tst adlı örnek MQSC kütüğünü çalıştırabilirsiniz. Örnekleri çalıştırmak için, sağlanan yürütülebilir sürümleri kullanın ya da ANSI derleyicisini kullanarak diğer uygulamalarda olduğu gibi kaynak sürümleri derleyin.

Yordam

1. Bir kuyruk yöneticisi yaratın ve varsayılan tanımlamaları ayarlayın.

   Örnek programlardan herhangi birini çalıştırmadan önce bunu yapmanız gerekir. Kuyruk yöneticisi oluşturma hakkında daha fazla bilgi için bkz. Yönetme IBM MQ. Bir kuyruk yöneticisinin istemci kipinde çalışan uygulamalardan gelen bağlantı isteklerini güvenli bir şekilde kabul edecek şekilde yapılandırılmasıyla ilgili bilgi için bkz. “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023.
2. Kendi kuyruklarınızı kullanmıyorsanız, bir kuyruk kümesi yaratmak için amqscos0.tst adlı örnek MQSC kütüğünü çalıştırın.

   Bunu AIX and Linux sistemlerinde yapmak için şunu girin:

   ```
   runmqsc QManagerName <amqscos0.tst > /tmp/sampobj.out
   ```

   Hata olmadığından emin olmak için `sampobj.out` dosyasını denetleyin.
3. Inquire, Set ve Echo örneklerinin COBOL sürümlerini kullanmak istiyorsanız, bu örnekleri çalıştırmadan önce süreç tanımlamalarını değiştirin.

   Inquire, Set ve Echo örnekleri için, örnek tanımlamaları bu örneklerin C sürümlerini tetikler. COBOL sürümlerini istiyorsanız, süreç tanımlamalarını değiştirmeniz gerekir:

   - SYSTEM.SAMPLE.INQPROCESS
   - SYSTEM.SAMPLE.SETPROCESS
   - SYSTEM.SAMPLE.ECHOPROCESS

   Windowsişletim sisteminde bunu, bu örnekleri çalıştırmak için **runmqsc** komutunu kullanmadan önce `amqscos0.tst` dosyasını düzenleyerek ve C yürütülebilir dosya adlarını COBOL yürütülebilir dosya adlarına değiştirerek yapın.
4. Örnek programları çalıştırın.

   Bir örneği çalıştırmak için, adını ve ardından parametreleri girin, örneğin:

   ```
   amqsput myqueue qmanagername
   ```

Burada *myqueue* , iletilerin konacağı kuyruğun adı ve *qmanagername* , *myqueue*' un sahibi olan kuyruk yöneticisidir.

Her bir örneğin beklediği parametrelerle ilgili daha fazla bilgi için tek tek örneklerin açıklamalarına bakın.

Not: COBOL örnek programları için, kuyruk adlarını parametre olarak geçirdiğinizde, gerekirse boş karakterlerle doldurarak 48 karakter sağlamanız gerekir. 48 karakterden başka bir karakter, programın başarısız olmasına neden olur. Neden kodu 2085.

İlgili başvurular

"AIX and Linux sistemleri için örnekler" sayfa 1015
IBM MQ for AIX or Linuxiçin örnek programlarla gösterilen teknikler.

Windows *Windows üzerinde örnek programların hazırlanması ve çalıştırılması*

Örnek programları Windowsüzerinde çalıştırmadan önce, önce bir kuyruk yöneticisi yaratmanız ve ayrıca gerek duyduğunuz kuyrukları yaratmanız gerekir. COBOL örneklerini çalıştırmak istiyorsanız, ek hazırlık yapmanız gerekebilir.

Bu görev hakkında

Kuruluş sırasında varsayılan değerler kullanıldıysa, IBM MQ for Windows örnek dosyaları Çizelge 161 sayfa 1028içinde listelenen dizinlerdedir. Kuruluş sürücüsü varsayılan olarak < c: > değerine ayarlanır.

Çizelge 161. IBM MQ for Windows için örnekler nerede bulunacak?

İçerik	**Dizin**
C kaynak kodu	*MQ_INSTALLATION_PATH*\Araçlar \C\Örnekler
Gönderilmeyen harf işleyicisi örneği için kaynak kodu	*MQ_INSTALLATION_PATH*\Araçlar \C\Örnekler\DLQ
COBOL kaynak kodu	*MQ_INSTALLATION_PATH*\Araçlar \Cobol \ Örnekler
C yürütülür dosyaları [1]	*MQ_INSTALLATION_PATH*\ Tools\C\Samples \ Bin (32 bit sürümler) *MQ_INSTALLATION_PATH*\ Tools\C\Samples\Bin64 (64 bit sürümler)
Örnek MQSC dosyaları	*MQ_INSTALLATION_PATH*\Araçlar \MQSC\Örnekler
Visual Basic kaynak kodu	*MQ_INSTALLATION_PATH*\Tools\VB\SampVB6
.NET örnekleri	*MQ_INSTALLATION_PATH*\Araçlar \dotnet \ Örnekler

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Not: Bazı C yürütülebilir dosya örneklerinin 64 bitlik sürümleri vardır.

Örneklerle çalışmak için bir kuyruk kümesi gerekir. Kendi kuyruklarınızı kullanabilir ya da bir kuyruk kümesi yaratmak için amqscos0.tst adlı örnek MQSC kütüğünü çalıştırabilirsiniz. Örnekleri çalıştırmak için, sağlanan yürütülebilir sürümleri kullanın ya da diğer IBM MQ for Windows uygulamaları gibi kaynak sürümleri derleyin.

Yordam

1. Bir kuyruk yöneticisi yaratın ve varsayılan tanımlamaları ayarlayın.

   Örnek programlardan herhangi birini çalıştırmadan önce bunu yapmanız gerekir. Kuyruk yöneticisi oluşturma hakkında daha fazla bilgi için bkz. Yönetme IBM MQ. Bir kuyruk yöneticisinin istemci kipinde çalışan uygulamalardan gelen bağlantı isteklerini güvenli bir şekilde kabul edecek şekilde yapılandırılmasıyla ilgili bilgi için bkz. "Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması" sayfa 1023.

2. Kendi kuyruklarınızı kullanmıyorsanız, bir kuyruk kümesi yaratmak için amqscos0.tst adlı örnek MQSC kütüğünü çalıştırın.

   Windows sistemlerinde bunu yapmak için şunu girin:

   ```
   runmqsc QManagerName < amqscos0.tst > sampobj.out
   ```

   Hata olmadığından emin olmak için `sampobj.out` dosyasını denetleyin. Bu dosya geçerli dizininizde.
3. Inquire, Set ve Echo örneklerinin COBOL sürümlerini kullanmak istiyorsanız, bu örnekleri çalıştırmadan önce süreç tanımlamalarını değiştirin.

   Inquire, Set ve Echo örnekleri için, örnek tanımlamaları bu örneklerin C sürümlerini tetikler. COBOL sürümlerini istiyorsanız, süreç tanımlamalarını değiştirmeniz gerekir:

   - SYSTEM.SAMPLE.INQPROCESS
   - SYSTEM.SAMPLE.SETPROCESS
   - SYSTEM.SAMPLE.ECHOPROCESS

   Windowsişletim sisteminde bunu, bu örnekleri çalıştırmak için **runmqsc** komutunu kullanmadan önce `amqscos0.tst` dosyasını düzenleyerek ve C yürütülebilir dosya adlarını COBOL yürütülebilir dosya adlarına değiştirerek yapın.
4. Örnek programları çalıştırın.

   Bir örneği çalıştırmak için, adını ve ardından parametreleri girin, örneğin:

   ```
   amqsput myqueue qmanagername
   ```

   Burada *myqueue* , iletilerin konacağı kuyruğun adı ve *qmanagername* , *myqueue*' un sahibi olan kuyruk yöneticisidir.

   Her bir örneğin beklediği parametrelerle ilgili daha fazla bilgi için tek tek örneklerin açıklamalarına bakın.

   Not: COBOL örnek programları için, kuyruk adlarını parametre olarak geçirdiğinizde, gerekirse boş karakterlerle doldurarak 48 karakter sağlamanız gerekir. 48 karakterden başka bir karakter, programın başarısız olmasına neden olur. Neden kodu 2085.

İlgili başvurular

“IBM MQ for Windows için örnekler” sayfa 1018
IBM MQ for Windowsiçin örnek programlarla gösterilen teknikler.

“IBM MQ for Windows için Visual Basic örnekleri” sayfa 1020
Windows sistemlerinde IBM MQ için örnek programlarla gösterilen teknikler.

API çıkış örnek programı

Örnek API çıkışı, **MQAPI_TRACE_LOGFILE** ortam değişkeninde tanımlı bir önekle kullanıcı tarafından belirtilen bir dosya için MQI izlemesi oluşturur.

API çıkışları hakkında daha fazla bilgi için bkz. “Çoklu Platformda API çıkışlarının yazılması ve derlenmesi” sayfa 912.

Kaynak
amqsaxe0.c

İkili
amqsaxe

Örnek çıkış için yapılandırma

1. `qm.ini` dosyasının ApiExitLocal kısmına aşağıdaki bilgileri ekleyin.

   Windows dışındaki platformlar

   ```
   ApiExitLocal:
   Sequence=100
   ```

```
Function=EntryPoint
Module= MQ_INSTALLATION_PATH/samp/bin/amqsaxe
Name=SampleApiExit
```

Burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizini gösterir.

Windows

```
ApiExitLocal:
Sequence=100
Function=EntryPoint
Module= MQ_INSTALLATION_PATH\Tools\c\Samples\bin\amqsaxe
Name=SampleApiExit
```

Burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizini gösterir.

2. **MQAPI_TRACE_LOGFILE** ortam değişkenini ayarlama

```
MQAPI_TRACE_LOGFILE=/tmp/MqiTrace
```

3. Uygulamanızı çalıştırın.

   Çıkış dosyaları, `MqiTrace.pid.tid.log`gibi adlarla `/tmp` dizininde oluşturulur.

Zamanuyumsuz tüketim örnek programı

amqscbf örnek programı, birden çok kuyruktan gelen iletileri zamanuyumsuz olarak kullanmak için MQCB ve MQCTL ' nin kullanımını gösterir.

amqscbf, C kaynak kodu ve AIX, Linux, and Windows platformlarında yürütülebilir ikili istemci ve sunucu olarak sağlanır.

Program komut satırından başlatılır ve aşağıdaki isteğe bağlı parametreleri alır:

```
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
```

Birden çok kuyruktan ileti okumak için birden çok kuyruk adı sağlayın (örnek tarafından en çok on kuyruk desteklenir).

Not: `Reconnect type` yalnızca istemci programları için geçerlidir.

Örnek

Bu örnekte, amqscbf sunucu programı olarak çalıştırılıp `QL1` ' den bir ileti okunup daha sonra durdurulduğu gösterilmektedir.

`QL1`üzerinde bir test iletisi koymak için IBM MQ Explorer 'ı kullanın. Enter tuşuna basarak programı durdurun.

```
C:\>amqscbf QL1
Sample AMQSCBF0 start

Press enter to end
Message Call (9 Bytes) :
Message 1

Sample AMQSCBF0 end
```

amqscbf 'nin gösterdiği

Örnek, birden çok kuyruktan gelen iletilerin geliş sırasına göre nasıl okunacağını gösterir. Bu, zamanuyumlu MQGET kullanarak çok daha fazla kod gerektirir. Zamanuyumsuz tüketim durumunda yoklama gerekmez ve iş parçacığı ve depolama yönetimi IBM MQtarafından gerçekleştirilir. Bir "gerçek dünya" örneğinin hatalarla ilgilenmesi gerekir; örnek hatalarda konsola yazılır.

Örnek kod aşağıdaki adımları içerir:

1. Tek ileti tüketimi geri çağırma işlevini tanımlayın,

```
void MessageConsumer(MQHCONN      hConn,
                     MQMD    * pMsgDesc,
                     MQGMO   * pGetMsgOpts,
                     MQBYTE  * Buffer,
                     MQCBC   * pContext)
{ ... }
```

2. Kuyruk yöneticisine bağlanın,

```
MQCONNX(QMName,&cno,&Hcon,&CompCode,&CReason);
```

3. Giriş kuyruklarını açın ve her birini `MessageConsumer` geri çağırma işleviyle ilişkilendirin,

```
MQOPEN(Hcon,&od,O_options,&Hobj,&OpenCode,&Reason);
cbd.CallbackFunction = MessageConsumer;
MQCB(Hcon,MQOP_REGISTER,&cbd,Hobj,&md,&gmo,&CompCode,&Reason);
```

   `cbd.CallbackFunction` ' in her kuyruk için ayarlanması gerekmez; bu yalnızca giriş alanıdır. Ancak, her bir kuyrukla farklı bir geri çağırma işlevini ilişkilendirebilirsiniz.

4. İletilerin tüketimini başlat,

```
MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
```

5. Kullanıcı Enter tuşuna basıp iletilerin kullanımını durdurana kadar bekleyin.

```
MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
```

6. Son olarak kuyruk yöneticisiyle bağlantıyı kes,

```
MQDISC(&Hcon,&CompCode,&Reason);
```

Zamanuyumsuz Put örnek programı

amqsapt örneğini ve Zamanuyumsuz Put örnek programının tasarımını çalıştırma hakkında bilgi edinin.

Zamanuyumsuz koyma örnek programı, zamanuyumsuz MQPUT çağrıyı kullanarak iletileri kuyruğa koyar ve MQSTAT çağrısından durum bilgilerini alır. Bu programın farklı platformlardaki adı için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

amqsapt örneği çalıştırılıyor

Bu program en çok 6 parametre alır:

1. Hedef kuyruğun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)
3. Açma seçenekleri (isteğe bağlı)
4. Seçenekleri kapat (isteğe bağlı)
5. Hedef kuyruk yöneticisinin adı (isteğe bağlı)

6. Dinamik kuyruğun adı (isteğe bağlı)

Bir kuyruk yöneticisi belirtilmezse, amqsapt varsayılan kuyruk yöneticisine bağlanır.

Zamanuyumsuz Put örnek programının tasarımı

Program, çıkış seçenekleri sağlanan MQOPEN çağrısını ya da iletileri koymak üzere hedef kuyruğu açmak için MQOO_OUTPUT ve MQOO_FAIL_IF_QUIESCING seçeneklerini kullanır.

Kuyruğu açamazsa, program MQOPEN çağrısıyla döndürülen neden kodunu içeren bir hata iletisi verir. Programı basit tutmak için, bu ve sonraki MQI çağrılarında, program birçok seçenek için varsayılan değerleri kullanır.

Her giriş satırı için, program metni bir arabelleğe okur ve o satırın metnini içeren bir veri paketi iletisi yaratmak ve zamanuyumsuz olarak hedef kuyruğa koymak için MQPMO_ASYNC_RESPONSE ile MQPUT çağrıyı kullanır. Program, girişin sonuna ulaşıncaya ya da MQPUT çağrısı başarısız oluncaya kadar devam eder. Program girişin sonuna ulaşırsa, MQCLOSE çağrıyı kullanarak kuyruğu kapatır.

Program daha sonra MQSTAT çağrısını yayınlar, bir MQSTS yapısı verir ve başarıyla konan iletilerin sayısını, bir uyarıyla konan iletilerin sayısını ve başarısızlıkların sayısını içeren iletileri görüntüler.

Göz At örnek programları

Göz At örnek programları, MQGET çağrılarını kullanarak kuyruktaki iletilere göz atmanızı sağlar.

Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Göz At örnek programının tasarımı

Program, MQOO_BROWSE seçeneğiyle MQOPEN çağrısıyla hedef kuyruğu açar. Kuyruğu açamazsa, program MQOPEN çağrısıyla döndürülen neden kodunu içeren bir hata iletisi verir.

Kuyruktaki her ileti için, program iletiyi kuyruktan kopyalamak için MQGET çağrıyı kullanır ve iletide bulunan verileri görüntüler. MQGET çağrısı şu seçenekleri kullanır:

MQGMO_BROWSE_NEXT
MQOPEN çağrısından sonra, göz atma imleci mantıksal olarak kuyruktaki ilk iletiden önce konumlandırılır; bu nedenle, çağrı ilk yapıldığında ***ilk*** iletisinin döndürülmesine neden olur.

MQGMO_NO_WAIT
Kuyrukta ileti yoksa program beklemez.

MQGMO_ACCEPT_TRUNCATED_MSG
MQGET çağrısı, değişmez büyüklükte bir arabellek belirtiyor. Bir ileti bu arabellekten uzunsa, program kesilen iletiyi, iletinin kesildiğine ilişkin bir uyarıyla birlikte görüntüler.

Çağrı, bu alanları aldığı iletide bulunan değerlere ayarladığından, her MQGET çağrısından sonra MQMD yapısının *MsgId* ve *CorrelId* alanlarını nasıl temizlemeniz gerektiğini gösterir. Bu alanların temizlenmesi, ardışık MQGET çağrıları iletilerin kuyrukta tutulduğu sırada iletileri alır.

Program kuyruğun sonuna kadar devam eder; MQGET çağrısı MQRC_NO_MSG_AVAILABLE neden kodunu döndürür ve program bir uyarı iletisi görüntüler. MQGET çağrısı başarısız olursa, program neden kodunu içeren bir hata iletisi görüntüler.

Daha sonra program, MQCLOSE çağrılarını kullanarak kuyruğu kapatır.

AIX, Linux, and Windows için Göz At örnek programları
AIX, Linux, and Windowsüzerinde örnek programlara göz at hakkında bilgi edinirken bu konuyu kullanmayı düşünün.

Programın C sürümü 2 parametre alır

1. Kaynak kuyruğun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)

Bir kuyruk yöneticisi belirtilmezse, varsayılan yöneticiye bağlanır. Örneğin, aşağıdakilerden birini girin:

- `amqsgbr myqueue qmanagername`
- `amqsgbrc myqueue qmanagername`
- `amq0gbr0 myqueue`

Burada `myqueue` , iletilerin görüntüleneceği kuyruğun adı ve `qmanagername` , `myqueue`' un sahibi olan kuyruk yöneticisidir.

`qmanagername`eatlarsanız, C örneğini çalıştırırken, kuyruğun varsayılan kuyruk yöneticisinin sahibi olduğu varsayılır.

COBOL sürümünde parametre yok. Varsayılan kuyruk yöneticisine bağlanır ve çalıştırdığınızda size sorulur:

```
Please enter the name of the target queue
```

Bu durumda, her iletinin yalnızca ilk 50 karakteri görüntülenir ve bunu `- - - truncated` izler.

IBM i üzerinde Göz At örnek programları
Her program, programı çağırdığınızda belirlediğiniz kuyruktaki tüm iletilerin kopyalarını alır; iletiler kuyrukta kalır.

Sağlanan SYSTEM.SAMPLE.LOCAL; kuyruğa bazı iletiler koymak için önce Put örnek programını çalıştırın. SYSTEM.SAMPLE.ALIAS. Program, kuyruğun sonuna ulaşıncaya ya da bir MQI çağrısı başarısız oluncaya kadar devam eder.

C örnekleri, kuyruk yöneticisi adını (genellikle ikinci parametre olarak) Windows sistem örneklerine benzer bir şekilde belirtmenizi sağlar. Örneğin:

```
CALL PGM(QMQM/AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER' 'QM01')
```

Bir kuyruk yöneticisi belirtilmezse, varsayılan yöneticiye bağlanır. Bu RPG örnekleriyle de ilgilidir. Ancak, RPG örnekleriyle, varsayılan olarak izin vermek yerine bir kuyruk yöneticisi adı sağlamanız gerekir.

ALW *Tarayıcı örnek programı*

Tarayıcı örnek programı, bir kuyruktaki tüm iletilerin ileti tanımlayıcısı ve ileti içeriği alanlarını okur ve yazar.

Örnek program sadece bir tekniği göstermek için değil, bir yardımcı program olarak yazılır. Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Bu program şu konumsal parametreleri alır:

1. Kaynak kuyruğun adı (gerekli)
2. Kuyruk yöneticisinin adı (gerekli)
3. Özellikler için isteğe bağlı parametre (isteğe bağlı)

Kuyruk yöneticisiyle kimlik doğrulaması yapmak için kullanılan kimlik bilgilerini sağlamak üzere aşağıdaki ortam değişkenlerini kullanın:

MQSAMP_USER_ID
Kuyruk yöneticisiyle kimlik doğrulaması yapmak için bir kullanıcı kimliği ve parola kullanmak istiyorsanız, bağlantı kimlik doğrulaması için kullanılacak kullanıcı kimliğini ayarlayın. Program, kullanıcı kimliğiyle birlikte parolanın da girmesini ister.

Linux AIX V 9.3.4 **MQSAMP_TOKEN**
Kuyruk yöneticisiyle kimlik doğrulaması için bir kimlik doğrulama belirteci sağlamak istiyorsanız, boş olmayan bir değere ayarlayın. Program, kimlik doğrulama belirtecini ister. Kimlik doğrulama belirteçleri yalnızca istemci bağ tanımlarını kullanan **amqsbcgc** örneği tarafından kullanılabilir.

Bu programları çalıştırmak için aşağıdaki komutlardan birini girin:

- amqsbcg *myqueue qmanagername*
- amqsbcgc *myqueue qmanagername*

Burada *myqueue* , iletilere göz atılacak kuyruğun adı ve *qmanagername* , *myqueue*' un sahibi olan kuyruk yöneticisidir.

Kuyruktaki her iletiyi okur ve stdout 'a aşağıdakileri yazar:

- Biçimlendirilmiş ileti tanımlayıcı alanları
- İleti verileri (onaltılı ve mümkünse karakter biçiminde dökülür)

Çizelge 162. Özellik parametresi için izin verilebilir değerler

Değer	Davranış
0	Varsayılan davranış. Uygulamaya teslim edilen özellikler, iletinin alındığı **PropertyControl** kuyruk özniteliğine bağlıdır.

1	Bir ileti tanıtıcısı yaratılır ve MQGET ile kullanılır. İleti tanımlayıcısında (ya da uzantıda) bulunanlar dışında, iletinin özellikleri, ileti tanımlayıcısına benzer bir şekilde görüntülenir. Örneğin:
	`****Message properties****`
	`  property name: property value`
	Ya da herhangi bir özellik yoksa:
	`****Message properties****`
	`  None`
	Sayısal değerler `printf`kullanılarak, dize değerleri tek tırnak işareti içinde, bayt dizgileri ise ileti tanımlayıcısı olarak X ve tek tırnak işareti ile çevrelenir.

2	MQGMO_NO_PROPERTIES belirtildiğinden yalnızca ileti tanımlayıcı özellikleri döndürülecek.
3	MQGMO_PROPERTIES_FORCE_MQRFH2 , ileti verilerinde tüm özelliklerin döndürülmesi için belirtilir.
4	MQGMO_PROPERTIES_COMPATIBILITY belirtildiğinden, IBM MQ özelliğinin içerilip içerilmediğine bağlı olarak tüm özellikler döndürülebilir, tersi durumda özellikler atılır.

Program, iletinin ilk 65535 karakterini yazdırmayla sınırlıdır ve daha uzun bir ileti okunursa `truncated msg` nedeni ile başarısız olur.

Bu yardımcı programın çıkışına ilişkin bir örnek için Kuyruklara göz atmabaşlıklı konuya bakın.

CICS hareket örneği

Kaynak kod için amqscic0.ccs ve yürütülebilir sürüm için amqscic0 adlı örnek bir CICS hareket programı sağlanır. Standart CICS olanaklarını kullanarak işlem oluşturabilirsiniz.

Altyapınız için gereken komutlara ilişkin ayrıntılar için bkz. “Yordamsal uygulama oluşturma” sayfa 957 .

Hareket, iletileri SYSTEM.SAMPLE.CICSiletim kuyruğundan okur.WORKQUEUE varsayılan kuyruk yöneticisinde yer alır ve bunları, adı iletinin iletim üstbilgisinde bulunan yerel kuyruğa yerleştirir. Hatalar SYSTEM.SAMPLECICSkuyruğuna gönderilir.-DLQ.

Not: Bu kuyrukları ve örnek giriş kuyruklarını yaratmak için örnek bir MQSC komut dosyası amqscic0.tst kullanabilirsiniz.

Connect örnek programı

Connect örnek programı, MQCONNX çağrısını ve seçeneklerini bir istemciden keşfetmenizi sağlar. Örnek, MQCONNX çağrısı kullanılarak kuyruk yöneticisine bağlanır, MQINQ çağrısı kullanılarak kuyruk yöneticisinin adı hakkında bilgi alır ve bunu görüntüler. Ayrıca, amqscnxc örneğini çalıştırmayı öğrenin.

Not: Connect örnek programı bir istemci örneğidir. Bu işlevi bir sunucuda derleyebilir ve çalıştırabilirsiniz, ancak işlev yalnızca bir istemcide anlamlıdır ve yalnızca istemci tarafından yürütülebilir dosyalar sağlanır.

amqscnxc örneğinin çalıştırılması

Connect örnek programının komut satırı sözdizimi şöyledir:

```
amqscnxc [-x ConnName [-c SvrconnChannelName]] [-u User] [QMgrName]
```

Parametreler isteğe bağlıdır ve sıraları, belirtildiyse sonuncu olması gereken QMgrNamedışında önemli değildir. Değiştirgeler şunlardır:

ConnName
Sunucu kuyruk yöneticisinin TCP/IP bağlantısı adı

TCP/IP bağlantı adını belirtmezseniz, MQCONNX, *ClientConnPtr* ayarı NULL olarak ayarlanır.

SvrconnChannelAdı
Sunucu bağlantı kanalının adı

TCP/IP bağlantı adını belirtirseniz, ancak sunucu bağlantı kanalını belirlemezseniz (tersi kullanılamaz), örnek SYSTEM.DEF.SVRCONN.

Kullanıcı
Bağlantı doğrulaması için kullanılacak kullanıcı adı

Bu değeri belirlerseniz, program bu kullanıcı kimliğine eşlik edecek bir parola ister.

QMgrName
Hedef kuyruk yöneticisinin adı

Hedef kuyruk yöneticisini belirtmezseniz, örnek, belirtilen TCP/IP bağlantı adında dinleyen kuyruk yöneticisine bağlanır.

Not: Tek parametre olarak bir soru işareti girerseniz ya da yanlış parametreler girerseniz, programın nasıl kullanılacağını açıklayan bir ileti alırsınız.

Örneği komut satırı seçenekleri olmadan çalıştırırsanız, bağlantı bilgilerini saptamak için MQSERVER ortam değişkeninin içeriği kullanılır. (Bu örnekte MQSERVER, `SYSTEM.DEF.SVRCONN/TCP/machine.site.company.com`olarak ayarlanmıştır.) Çıkışı şu şekilde görürsünüz:

```
Sample AMQSCNXC start
Connecting to the default queue manager
with no client connection information specified.
Connection established to queue manager machine

Sample AMQSCNXC end
```

Örneği çalıştırır ve bir TCP/IP bağlantı adı ve bir sunucu bağlantı kanalı adı sağlarsanız, ancak hedef kuyruk yöneticisi adı yoksa, aşağıdaki gibi:

```
amqscnxc -x machine.site.company.com -c SYSTEM.ADMIN.SVRCONN
```

Varsayılan kuyruk yöneticisi adı kullanılır ve çıkışı şu şekilde görürsünüz:

```
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
```

Örneği çalıştırır ve bir TCP/IP bağlantı adı ve bir hedef kuyruk yöneticisi adı sağlarsanız, aşağıdaki gibi:

```
amqscnxc -x machine.site.company.com MACHINE
```

Çıkışı şu şekilde görürsünüz:

```
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
```

Data-Conversion örnek programı

Veri dönüştürme örnek programı, bir veri dönüştürme çıkış yordamının iskeleti. Veri dönüştürme örneğinin tasarımı hakkında bilgi edinin.

Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Veri dönüştürme örneğinin tasarımı

Her veri dönüştürme çıkış yordamı, tek bir adlandırılmış ileti biçimini dönüştürür. Bu iskelet, veri dönüştürme çıkış oluşturma yardımcı programı tarafından oluşturulan kod parçaları için bir sarıcı olarak tasarlanmıştır.

Yardımcı program her veri yapısı için bir kod parçası üretir; bu tür birkaç yapı bir biçim oluşturur; bu nedenle, tüm biçimin veri dönüşümünü gerçekleştirecek bir yordam oluşturmak için bu iskelete birkaç kod parçası eklenir.

Daha sonra program, dönüştürmenin başarılı mı, yoksa başarısız mı olduğunu denetler ve çağıran için gereken değerleri döndürür.

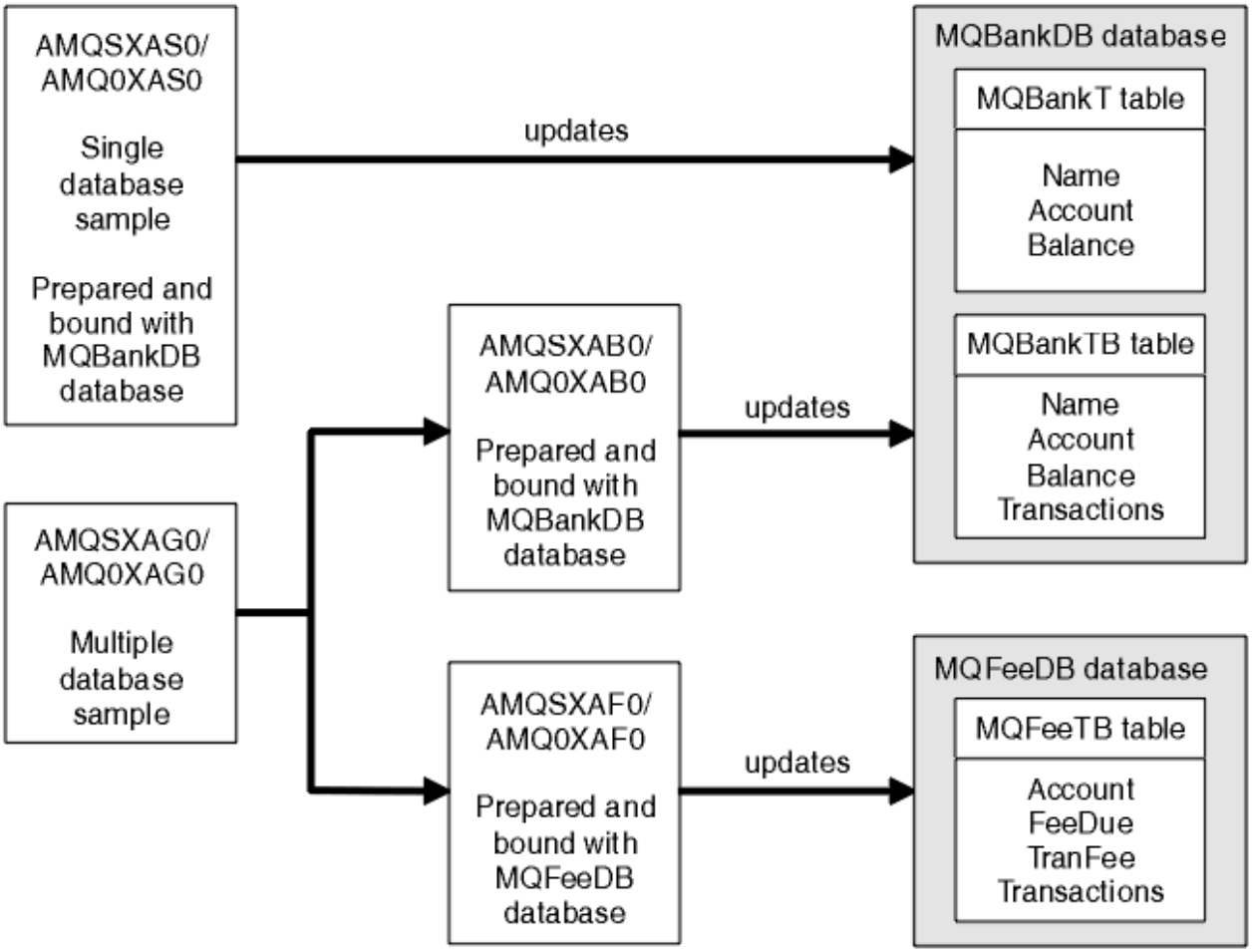
Veritabanı koordinasyonu örnekleri

IBM MQ ' in aynı iş birimi içinde hem IBM MQ güncellemelerini hem de veritabanı güncellemelerini nasıl koordine edebileceğini gösteren iki örnek sağlanır.

Bu örnekler şunlardır:

1. AMQSXAS0 (C) ya da AMQ0XAS0 (COBOL), IBM MQ iş birimindeki tek bir veritabanını günceller.
2. AMQSXAG0 (C dilinde) ya da AMQ0XAG0 (COBOL dilinde), AMQSXAB0 (C dilinde) ya da AMQ0XAB0 (COBOL dilinde) ve AMQSXAF0 (C içinde) ya da AMQ0XAF0 (COBOL içinde), bir IBM MQ iş birimindeki iki veritabanını birlikte güncelleyerek, birden çok veritabanına nasıl erişilebileceğini gösterir. Bu örnekler, MQBEGIN çağrısının, karma SQL ve IBM MQ çağrılarının kullanımını ve bir veritabanına nerede ve ne zaman bağlanacağımı göstermek için sağlanır.

Şekil 128 sayfa 1037 , sağlanan örneklerin veritabanlarını güncellemek için nasıl kullanıldığını gösterir:



Şekil 128. Veritabanı koordinasyonu örnekleri

Programlar bir kuyruktan (syncpoint altında) bir ileti okur, daha sonra iletideki bilgileri kullanarak veritabanından ilgili bilgileri alır ve günceller. Daha sonra, veritabanının yeni durumu yazdırılır.

Program mantığı aşağıdaki gibidir:

1. Program bağımsız değişkenindeki giriş kuyruğunun adını kullan
2. MQCONN kullanarak varsayılan kuyruk yöneticisine (ya da isteğe bağlı olarak C dilinde sağlanan ada) bağlan
3. Hata yokken giriş için bir kuyruk aç (MQOPEN kullanarak)
4. MQBEGIN ile bir iş birimi başlat
5. Eşitleme noktası altındaki kuyruktan sonraki iletiyi (MQGET kullanarak) al
6. Veritabanlarından bilgi al
7. Veritabanlarındaki bilgileri güncelle
8. MQCMIT kullanarak değişiklikleri kesinleştir
9. Güncellenen bilgileri yazdır (kullanılabilir ileti olmaması başarısızlık olarak sayılır ve döngü sona erer)
10. MQCLOSE komutunu kullanarak kuyruğu kapat
11. MQDISC kullanarak kuyruktan bağlantıyı kes

Örneklerde SQL imleçleri kullanılır; böylece, bir ileti işlenirken veritabanlarından (yani birden çok yönetim ortamından) okuma kilitlenir ve bu, bu programların birden çok eşgörünümünün eşzamanlı olarak çalışmasına olanak tanır. İmleçler belirtik olarak açıldı, ancak MQCMIT çağrısı tarafından örtük olarak kapatıldı.

Tek veritabanı örneğinde (AMQSXAS0 ya da AMQ0XAS0) SQL CONNECT deyimi yok ve veritabanıyla bağlantı IBM MQ tarafından MQBEGIN çağrısıyla örtük olarak yapıldı. Birden çok veritabanı örneği (AMQSXAG0 ya da AMQ0XAG0, AMQSXAB0 ya da AMQ0XAB0ve AMQSXAF0 ya da AMQ0XAF0) SQL CONNECT deyimlerine sahiptir; bazı veritabanı ürünleri yalnızca bir etkin bağlantıya izin verir. Veritabanı ürününüz için böyle değilse ya da birden çok veritabanı ürünündeki tek bir veritabanına erişiyorsanız, SQL CONNECT deyimleri kaldırılabilir.

Örnekler IBM Db2 veritabanı ürünüyle hazırlanır; bu nedenle, diğer veritabanı ürünleriyle çalışmak için bunları değiştirmeniz gerekebilir.

SQL hata denetimi, Db2tarafından sağlanan UTIL.C ve CHECKERR.CBL içindeki yordamları kullanır. Bunlar derlenmeden ve bağlanmadan önce derlenmeli ya da değiştirilmelidir.

Not: Micro Focus COBOL kaynağını kullanıyorsanız CHECKERR.MFC , AMQ0XAS0 için program tanıtıcısını büyük harfe çevirmeniz gerekir.

Veritabanlarının ve tabloların oluşturulması
Örnekleri derlemeden önce veritabanlarını ve çizelgeleri yaratın.

Veritabanlarını yaratmak için, veritabanı ürününüz için olağan yöntemi kullanın; örneğin:

```
DB2 CREATE DB MQBankDB
DB2 CREATE DB MQFeeDB
```

Aşağıdaki SQL deyimlerini kullanarak çizelgeleri yaratın:

C İçinde:

```
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));
```

COBOL dilinde:

```
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.
```

Aşağıdaki SQL deyimlerini kullanarak çizelgelere veri girin:

```
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);
⋮
```

Not: COBOL için, aynı SQL deyimlerini kullanın, ancak her satırın sonuna END_EXEC ekleyin.

Örneklerin ön derlenmesi, derlenmesi ve bağlanması
C ve COBOL ' da örnekleri ön derleme, derleme ve bağlama hakkında bilgi edinin.

.SQC dosyalarını (C) ve .SQB dosyalarını (COBOL içinde) ön derleyin ve bunları .C ya da .CBL dosyalarını üretmek için uygun veritabanına bağlayın. Bunu yapmak için, veritabanı ürününüze ilişkin tipik yöntemi kullanın.

C içinde ön derleniyor

```
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
```

COBOL ' da ön derleme

```
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
```

Derleniyor ve bağlantı kuruluyor

Aşağıdaki örnek komutlar *DB2TOP* ve *MQ_INSTALLATION_PATH*simgelerini kullanır. *DB2TOP* , Db2 ürününün kuruluş dizinini gösterir. *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

- AIX AIX' de dizin yolu şöyledir:

  ```
  /usr/lpp/db2_05_00
  ```

- Windows Windows sistemlerinde dizin yolu, ürünü kurarken seçilen yola bağlıdır. Varsayılan ayarları seçerseniz yol şöyledir:

```
c:\sqllib
```

Not: Windows sistemlerinde bağlantı komutunu vermeden önce, LIB ortam değişkeninin Db2 ve IBM MQ kitaplıklarının yollarını içerdiğinden emin olun.

Aşağıdaki dosyaları geçici bir dizine kopyalayın:

- IBM MQ kuruluşunuzda bulunan `amqsxag0.c` dosyası

  Not: Bu dosya aşağıdaki dizinlerde bulunabilir:

  – Linux AIX AIX and Linux sistemlerinde:

  ```
  MQ_INSTALLATION_PATH/samp/xatm
  ```

  – Windows Windows sistemlerinde:

  ```
  MQ_INSTALLATION_PATH\tools\c\samples\xatm
  ```

- `.sqc` kaynak dosyalarını, `amqsxas0.sqc`, `amqsxaf0.sqc`ve `amqsxab0.sqc`ön derleyerek edindiğiniz `.c` dosyaları.
- Db2 kuruluşunuzda bulunan `util.c` ve `util.h` dosyaları.

  Not: Bu dosyalar dizinde bulunabilir:

  ```
  DB2TOP/samples/c
  ```

Kullanmakta olduğunuz platform için aşağıdaki derleyici komutunu kullanarak her `.c` dosyası için nesne dosyalarını oluşturun:

- AIX AIX

```
xlc_r -I MQ_INSTALLATION_PATH/inc -I DB2TOP/include -c -o
FILENAME.o FILENAME.c
```

- Windows Windows sistemleri

```
cl /c /I MQ_INSTALLATION_PATH\tools\c\include /I DB2TOP\include
FILENAME.c
```

Kullandığınız platform için aşağıdaki bağlantı komutunu kullanarak `amqsxag0` yürütülür dosyasını oluşturun:

- AIX AIX

```
xlc_r -H512 -T512 -L DB2TOP/lib -ldb2 -L MQ_INSTALLATION_PATH/lib
-lmqm util.o amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

- Windows Windows sistemleri

```
link util.obj amqsxaf0.obj amqsxab0.obj amqsxag0.obj mqm.lib db2api.lib
/out:amqsxag0.exe
```

Kullandığınız platform için aşağıdaki derleme ve bağlama komutlarını kullanarak `amqsxas0` yürütülür dosyasını oluşturun:

- AIX AIX

```
xlc_r -H512 -T512 -L DB2TOP/lib -ldb2
-L MQ_INSTALLATION_PATH/lib -lmqm util.o amqsxas0.o -o amqsxas0
```

- Windows Windows sistemleri

```
link util.obj amqsxas0.obj mqm.lib db2api.lib /out:amqsxas0.exe
```

Ek bilgiler

AIX AIX üzerinde çalışıyorsanız ve Oracleolanağına erişmek istiyorsanız, xlc_r derleyicisini kullanın ve libmqm_r.abağlantısını kullanın.

Örnekleri çalıştırma
C ve COBOL üzerinde veritabanı koordinasyonu örneklerini çalıştırmadan önce kuyruk yöneticisinin nasıl yapılandırılacağını öğrenmek için bu bilgileri kullanın.

Örnekleri çalıştırmadan önce, kuyruk yöneticisini kullanmakta olduğunuz veritabanı ürünüyle yapılandırın. Bunun nasıl gerçekleştirileceğine ilişkin bilgi için bkz. Senaryo 1: Kuyruk yöneticisi eşgüdümü gerçekleştirir.

Aşağıdaki başlıklar, C ve COBOL ' da örneklerin nasıl çalıştırılacağı hakkında bilgi sağlar:

- “C örnekleri” sayfa 1041
- “COBOL örnekleri” sayfa 1042

C örnekleri

Bir kuyruktan okunabilmesi için iletilerin aşağıdaki biçimde olması gerekir:

```
UPDATE Balance change=nnn WHERE Account=nnn
```

AMQSPUT, iletileri kuyruğa koymak için kullanılabilir.

Veritabanı eşgüdüm örnekleri iki parametreyi alır:

1. Kuyruk adı (gerekli)
2. Kuyruk yöneticisi adı (isteğe bağlı)

singDBQMadlı tek veritabanı örneği için singDBQadlı bir kuyruk oluşturup yapılandırdığınızı varsayarak, Bay Fred Bloggs 'un hesabını aşağıdaki gibi 50 artırdınız:

```
AMQSPUT singDBQ singDBQM
```

Daha sonra aşağıdaki iletideki anahtar:

```
UPDATE Balance change=50 WHERE Account=1
```

Kuyruğa birden çok ileti koyabilirsiniz.

```
AMQSXAS0 singDBQ singDBQM
```

Daha sonra Bay Fred Bloggs 'un hesabının güncellenen durumu yazdırılır.

multDBQMadlı çoklu veritabanı örneği için multDBQadlı bir kuyruk oluşturup yapılandırdığınızı varsayarak, Ms Mary Brown 'un hesabını aşağıdaki gibi 75 oranında azaltırsınız:

```
AMQSPUT multDBQ multDBQM
```

Daha sonra aşağıdaki iletideki anahtar:

```
UPDATE Balance change=-75 WHERE Account=3
```

Kuyruğa birden çok ileti koyabilirsiniz.

```
AMQSXAG0 multDBQ multDBQM
```

Daha sonra Bayan Mary Brown 'un hesabının güncellenen durumu yazdırılır.

COBOL örnekleri

Bir kuyruktan okunabilmesi için iletilerin aşağıdaki biçimde olması gerekir:

```
UPDATE Balance change=snnnnnnnn WHERE Account=nnnnnnnn
```

Basitlik için, `Balance change` işaretli sekiz karakterli bir sayı olmalı ve `Account` sekiz karakterli bir sayı olmalıdır.

Örnek AMQSPUT, iletileri kuyruğa koymak için kullanılabilir.

Örnekler parametre kullanmaz ve varsayılan kuyruk yöneticisini kullanır. Aynı anda örneklerden yalnızca birini çalıştıracak şekilde yapılandırılabilir. Tek veritabanı örneği için varsayılan kuyruk yöneticisini singDBQadlı bir kuyrukla yapılandırdığınızı varsayarak, Bay Fred Bloggs 'un hesabını aşağıdaki gibi 50 artırın:

```
AMQSPUT singDBQ
```

Daha sonra aşağıdaki iletideki anahtar:

```
UPDATE Balance change=+00000050 WHERE Account=00000001
```

Kuyruğa birden çok ileti koyabilirsiniz:

```
AMQ0XAS0
```

Kuyruğun adını yazın:

```
singDBQ
```

Daha sonra Bay Fred Bloggs 'un hesabının güncellenen durumu yazdırılır.

Birden çok veritabanı örneği için varsayılan kuyruk yöneticisini multDBQadlı bir kuyrukla yapılandırdığınızı varsayarak, Bayan Mary Brown 'un hesabını aşağıdaki gibi 75 oranında azaltırsınız:

```
AMQSPUT multDBQ
```

Daha sonra aşağıdaki iletideki anahtar:

```
UPDATE Balance change=-00000075 WHERE Account=00000003
```

Kuyruğa birden çok ileti koyabilirsiniz:

```
AMQ0XAG0
```

Kuyruğun adını yazın:

```
multDBQ
```

Daha sonra Bayan Mary Brown 'un hesabının güncellenen durumu yazdırılır.

Gönderilmeyen ileti kuyruğu işleyicisi örneği

Örnek bir ileti kuyruğu işleyicisi sağlanmıştır; yürütülebilir sürümün adı `amqsdlq`. **RUNMQDLQ**' den farklı bir teslim olmayan ileti kuyruğu işleyicisi istiyorsanız, örneğin kaynağı temel olarak kullanabilirsiniz.

Örnek, ürün içinde sağlanan ölü harf işleyicisine benzer, ancak izleme ve hata raporlaması farklıdır. Kullanabileceğiniz iki ortam değişkeni vardır:

ODQ_TRACE
İzlemeyi açmak için YES ya da `yes` değerine ayarlayın.

ODQ_MSG
Hata ve bilgi iletilerini içeren dosyanın adını belirleyin. Sağlanan dosyanın adı `amqsdlq.msg`.

Bu değişkenleri, altyapınıza bağlı olarak **`export`** ya da **`set`** komutlarını kullanarak ortamınızda tanımanız gerekir; izleme **unset** komutu kullanılarak kapatılabilir.

Hata iletisi dosyasını (`amqsdlq.msg`) kendi gereksinimlerinize uyacak şekilde değiştirebilirsiniz. Örnek, iletileri IBM MQ hata günlüğü dosyasına **değil** standart çıkısına koyar.

Kullanılmayacak mektup işleyicisinin nasıl çalıştığına ve nasıl çalıştığına ilişkin daha fazla bilgi için, İletilerin IBM MQ gönderilmeyen iletiler kuyruğunda işlenmesi ya da altyapınıza ilişkin *Sistem Yönetimi Kılavuzu* ' na bakın.

Dağıtım Listesi örnek programı

Dağıtım Listesi örneği amqsptl0 , birkaç ileti kuyruğuna ileti koyma örneğini verir. MQPUT örneğine (amqsput0) dayalıdır.

amqsptl0 Dağıtım Listesi örneğinin çalıştırılması

Dağıtım Listesi örneği, Put örneklerine benzer bir şekilde çalışır.

Aşağıdaki parametreleri alır:

- Kuyrukların adları
- Kuyruk yöneticilerinin adları

Bu değerler çift olarak girilir. Örneğin:

```
amqsptl0 queue1 qmanagername1 queue2 qmanagername2
```

Kuyruklar MQOPEN kullanılarak açılır ve iletiler MQPUT kullanılarak kuyruklara konmuştur. Kuyruk ya da kuyruk yöneticisi adlarından herhangi biri tanınmazsa neden kodları döndürülür.

İletilerin aralarına akması için kuyruk yöneticileri arasında kanallar tanımlamayı unutmayın. Örnek program bunu sizin için yapmaz.

Dağıtım Listesi örneğinin tasarımı

Koyma İletisi Kayıtları (MQPMR), her hedef için ileti özniteliklerini belirtir. Örnek, *`MsgId`* ve *`CorrelId`*için değerler sağlar ve bunlar MQMD yapısında belirtilen değerleri geçersiz kılar.

MQPMO yapısındaki *PutMsgRecFields* alanı, MQPMR ' larda hangi alanların bulunduğunu gösterir:

```
MQLONG PutMsgRecFields=MQPMRF_MSG_ID + MQPMRF_CORREL_ID;
```

Daha sonra, örnek yanıt kayıtlarını ve nesne kayıtlarını ayırır. Nesne kayıtları (MQOR) için en az bir çift ad ve çift sayıda ad gerekir; *ObjectName* ve *ObjectQMgrName*.

Sonraki aşama, MQCONN kullanarak kuyruk yöneticilerine bağlanmayı içerir. Örnek, MQOR ' daki ilk kuyrukla ilişkilendirilmiş kuyruk yöneticisine bağlanmayı dener; bu başarısız olursa, sırayla nesne kayıtlarından geçer. Herhangi bir kuyruk yöneticisine bağlanıp bağlanamadığınız size bildirilir ve program çıkar.

Hedef kuyruklar MQOPEN kullanılarak açılır ve ileti MQPUT kullanılarak bu kuyruklara konmuştur. Yanıt kayıtlarında (MQRR) sorunlar ve hatalar bildirilir.

Son olarak, hedef kuyruklar MQCLOSE kullanılarak kapatılır ve program MQDISC kullanarak kuyruk yöneticisiyle bağlantısını keser. *CompCode* ve *Reason*belirten her arama için aynı yanıt kayıtları kullanılır.

Echo örnek programları

Echo örnek programları, ileti kuyruğundaki bir iletiyi yanıt kuyruğuna yansıtır.

Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Programların tetiklenen programlar olarak çalıştırılması amaçlanmıştır.

IBM i, AIX, Linux, and Windows sistemlerinde, bunların tek girişi, hedef kuyruğun adını ve kuyruk yöneticisini içeren bir MQTMC2 (tetikleyici iletisi) yapısıdır. COBOL sürümü, varsayılan kuyruk yöneticisini kullanır.

IBM i IBM iişletim sistemi üzerinde, tetikleme işleminin çalışması için, kullanmak istediğiniz Echo örnek programının SYSTEM.SAMPLE.ECHO. Bunu yapmak için, *ApplId* süreç tanımlamasının SYSTEM.SAMPLE.ECHOPROCESSalanında kullanmak istediğiniz Echo örnek programının adını belirtin. (Bunun için CHGMQMPRC komutunu kullanabilirsiniz; ayrıntılar için Change MQ Process (CHGMQMPRC)başlıklı konuya bakın.) Örnek kuyruk FIRST tetikleyici tipine sahiptir; bu nedenle, İstek örneğini çalıştırmadan önce kuyrukta önceden ileti varsa, gönderdiğiniz iletiler Echo örneğini tetiklemez.

Tanımı doğru ayarladığınızda, önce bir işte AMQSERV4 'ü başlatın, daha sonra başka bir işte AMQSREQ4 ' ü başlatın. AMQSERV4yerine AMQSTRG4 komutunu kullanabilirsiniz, ancak olası iş gönderimi gecikmeleri, olanların izlenmesini daha az kolaylaştıracaktır.

SYSTEM.SAMPLE.ECHO. Echo örnek programları, istek iletisindeki verileri içeren bir yanıt iletisini istek iletisinde belirtilen yanıt kuyruğuna gönderir.

Echo örnek programlarının tasarımı

Program, başlatılırken iletildiği tetikleyici ileti yapısında adı belirtilen kuyruğu açar. (Daha anlaşılır olması için, buna istek kuyruğu denir.) Program, paylaşılan giriş için bu kuyruğu açmak üzere MQOPEN çağrılmasını kullanır.

Program, iletileri bu kuyruktan kaldırmak için MQGET çağrısını kullanır. Bu çağrı, 5 saniyelik bekleme aralığıyla MQGMO_ACCEPT_TRUNCATED_MSG, MQGMO_CONVERT ve MQGMO_WAIT seçeneklerini kullanır. Program, bir istek iletisi olup olmadığını görmek için her iletinin tanımlayıcısını sınar; değilse, program iletiyi atar ve bir uyarı iletisi görüntüler.

Her giriş satırı için, program metni bir arabelleğe okur ve o satırın metnini içeren bir istek iletisini yanıt kuyruğuna koymak için MQPUT1 çağrıyı kullanır.

MQGET çağrısı başarısız olursa, program yanıt kuyruğuna bir rapor iletisi koyar ve ileti tanımlayıcısının *Feedback* alanını MQGET tarafından döndürülen neden koduna ayarlar.

İstek kuyruğunda kalan ileti yoksa, program o kuyruğu kapatır ve kuyruk yöneticisiyle olan bağlantısını keser.

IBM i IBM i' da, program kuyruğa IBM MQ for IBM idışındaki platformlardan gönderilen iletilere yanıt verebilir, ancak bu durum için örnek sağlanmaz. ECHO programının çalışmasını sağlamak için:

- Metin isteği iletileri göndermek için **Format**, **Encoding**ve **CCSID** parametrelerini doğru şekilde belirterek bir program yazın.

  ECHO programı, gerekiyorsa, kuyruk yöneticisinden ileti verilerini dönüştürmesini ister.
- Yazdığınız program yanıt için benzer bir dönüştürme sağlamıyorsa, IBM MQ for IBM i gönderme kanalında CONVERT (*YES) seçeneğini belirleyin.

"Get" örnek programları

Örnek programları al, MQGET çağrıyı kullanarak kuyruktan ileti alır.

Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Örnek Alma programının tasarımı

Program, MQOO_INPUT_AS_Q_DEF seçeneğiyle MQOPEN çağrısıyla hedef kuyruğu açar. Kuyruğu açamazsa, program MQOPEN çağrısıyla döndürülen neden kodunu içeren bir hata iletisi görüntüler.

Kuyruktaki her ileti için, program iletiyi kuyruktan kaldırmak için MQGET çağrısını kullanır ve iletide bulunan verileri görüntüler. MQGET çağrısı, kuyrukta ileti yoksa programın bu süreyi bekleyeceği 15 saniyelik bir *WaitInterval* belirterek MQGMO_WAIT seçeneğini kullanır. Bu aralığın süresi dolmadan hiçbir ileti gelmezse, çağrı başarısız olur ve MQRC_NO_MSG_AVAILABLE neden kodunu döndürür.

Çağrı, bu alanları aldığı iletide bulunan değerlere ayarladığından, her MQGET çağrısından sonra MQMD yapısının *MsgId* ve *CorrelId* alanlarını nasıl temizlemeniz gerektiğini gösterir. Bu alanların temizlenmesi, ardışık MQGET çağrıları iletilerin kuyrukta tutulduğu sırada iletileri alır.

MQGET çağrısı, değişmez büyüklükte bir arabellek belirtiyor. Bir ileti bu arabellekten daha uzunsa, çağrı başarısız olur ve program durur.

MQGET çağrısı MQRC_NO_MSG_AVAILABLE neden kodunu döndürünceye ya da MQGET çağrısı başarısız oluncaya kadar program devam eder. Çağrı başarısız olursa, program neden kodunu içeren bir hata iletisi görüntüler.

Daha sonra program, MQCLOSE çağrılarını kullanarak kuyruğu kapatır.

amqsget ve amqsgetc örneklerinin çalıştırılması

Bu programların her biri aşağıdaki konumsal parametreleri alır:

1. Kaynak kuyruğun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)

   Bir kuyruk yöneticisi belirtilmezse, **amqsget** varsayılan kuyruk yöneticisine bağlanır ve **amqsgetc** , *MQSERVER* ortam değişkeniyle ya da istemci kanal tanımlama dosyasıyla tanıtılan kuyruk yöneticisine bağlanır.
3. Açma seçenekleri (isteğe bağlı)

   Açma seçenekleri belirtilmezse, örnek şu iki seçeneğin birleşimi olan 8193 değerini kullanır:

   - MQOO_INPUT_AS_Q_DEF
   - MQOO_FAIL_IF_QUIESCING
4. Kapatma seçenekleri (isteğe bağlı)

   Kapatma seçenekleri belirtilmezse, örnek 0 değerini kullanır (MQCO_NONE).

Kuyruk yöneticisiyle kimlik doğrulaması yapmak için kullanılan kimlik bilgilerini sağlamak üzere aşağıdaki ortam değişkenlerini kullanın:

MQSAMP_USER_ID
Kuyruk yöneticisiyle kimlik doğrulaması yapmak için bir kullanıcı kimliği ve parola kullanmak istiyorsanız, bağlantı kimlik doğrulaması için kullanılacak kullanıcı kimliğini ayarlayın. Program, kullanıcı kimliğiyle birlikte parolanın da girmesini ister.

Linux AIX V 9.3.4 **MQSAMP_TOKEN**
Kuyruk yöneticisiyle kimlik doğrulaması için bir kimlik doğrulama belirteci sağlamak istiyorsanız, boş olmayan bir değere ayarlayın. Program, kimlik doğrulama belirtecini ister. Kimlik doğrulama belirteçleri yalnızca istemci bağ tanımlarını kullanan **amqsgetc** örneği tarafından kullanılabilir.

Bu programları çalıştırmak için aşağıdakilerden birini girin:

- `amqsget myqueue qmanagername`
- `amqsgetc myqueue qmanagername`

Burada *kuyruğum* , programın iletileri alacağı kuyruğun adı ve *qmanagername* , *kuyruğum*' un sahibi olan kuyruk yöneticisidir.

amqsget ve amqsgetc yöntemlerinin kullanılması

amqsget 'in kuyruk yöneticisine bağlanmak için paylaşılan bellek kullanarak kuyruk yöneticisine yerel bir bağlantı gerçekleştirdiğini ve bu nedenle yalnızca kuyruk yöneticisinin bulunduğu sistemde çalıştırılabileceğini, **amqsgetc** ' un ise istemci stili bir bağlantı gerçekleştirdiğini unutmayın (aynı sistemdeki bir kuyruk yöneticisine bağlansanız bile).

amqsgetc komutunu kullanırken, kuyruk yöneticisi anasistemi ya da IP adresi ve kuyruk yöneticisi dinleyici kapısı açısından kuyruk yöneticisine gerçekte nasıl ulaşılacağına ilişkin uygulama ayrıntılarını sağlamanız gerekir.

Olağan durumda bu, **MQSERVER** ortam değişkeni kullanılarak ya da ortam değişkenleri kullanılarak **amqsgetc** ' e sağlanabilen bir istemci kanal tanımlama çizelgesi kullanılarak bağlantı ayrıntıları tanımlanarak gerçekleştirilir; örneğin, MQCCDTURL.

1414 kapısında çalışan ve varsayılan sunucu bağlantı kanalını kullanan bir dinleyiciye sahip bir kuyruk yöneticisine yerel olarak bağlanmak için **MQSERVER**yönteminin kullanılması örneği:

```
export MQSERVER="SYSTEM.DEF.SVRCONN/TCP/ localhost(1414)"
```

Yüksek kullanılabilirlikli örnek programlar

amqsghac, **amqsphac**ve **amqsmhac** yüksek düzeyde kullanılabilirlik örnek programları, bir kuyruk yöneticisinin arızalanması sonrasında kurtarmayı göstermek için otomatik istemci yeniden bağlantısını kullanır. **amqsfhac** , ağ üzerinden depolama kullanan bir kuyruk yöneticisinin bir hatanın ardından veri bütünlüğünü korumasını denetler.

amqsghac, **amqsphac**ve **amqsmhac** programları komut satırından başlatılır ve çok eşgörünümlü bir kuyruk yöneticisinin bir eşgörünümü başarısız olduktan sonra yeniden bağlantıyı göstermek için birlikte kullanılabilir.

Diğer bir yöntem olarak, genellikle bir kuyruk yöneticisi grubunda yapılandırılmış olan tek eşgörünüm kuyruğu yöneticilerine istemci yeniden bağlantısını göstermek için **amqsghac**, **amqsphac**ve **amqsmhac** örneklerini de kullanabilirsiniz.

Örneği basit tutmak için, kolayca yapılandırılabilmesi için, başlatılan, durdurulan ve yeniden başlatılan tek bir yönetim ortamı kuyruk yöneticisine yeniden bağlanan örnek programlar gösterilir; bkz. “Kuyruk yöneticisinin ayarlanması ve denetlenmesi” sayfa 1048.

Dosya sistemi bütünlüğünü denetlemek için **amqmfsck** ile paralel olarak **amqsfhac** komutunu kullanın. Daha fazla bilgi için bkz. **amqmfsck** (dosya sistemi denetimi) ve Paylaşılan dosya sistemi davranışının doğrulanması .

amqsphac *queueName* [*qMgrAd*]

- **amqsphac** bir IBM MQ MQI client uygulamasıdır. Her ileti arasında iki saniyelik bir gecikme ile bir kuyruğa ileti dizisi koyar ve olay işleyicisine gönderilen olayları görüntüler.
- İletileri kuyruğa koymak için eşitleme noktası kullanılmaz.
- Aynı kuyruk yöneticisi grubundaki herhangi bir kuyruk yöneticisiyle yeniden bağlantı kurulabilir.

amqsghac *queueName* [*qMgrAd*]

- **amqsghac** bir IBM MQ MQI client uygulamasıdır. Bir kuyruktan ileti alır ve olay işleyicisine gönderilen olayları görüntüler.
- Kuyruktan ileti almak için eşitleme noktası kullanılmaz.
- Aynı kuyruk yöneticisi grubundaki herhangi bir kuyruk yöneticisiyle yeniden bağlantı kurulabilir.

amqsmhac -s *sourceQueueAdı* -t *targetQueueAdı* [-m *qMgrAdı*] [-w *waitInterval*]

- **amqsmhac** bir IBM MQ MQI client uygulamasıdır. Program tamamlanmadan önce alınan son iletiden 15 dakika sonra varsayılan bekleme aralığıyla iletileri bir kuyruktan diğerine kopyalar.
- İletiler eşitleme noktası içinde kopyalanır.
- Yalnızca aynı kuyruk yöneticisiyle yeniden bağlantı kurulabilir.

amqsfhac *QueueManagerAdı QueueName SideQueueName InTransactionCount RepeatCount* (0 | 1 | 2)

- **amqsfhac** bir IBM MQ MQI client uygulamasıdır. NAS ya da küme kütük sistemi gibi ağa bağlı depolama kullanan bir IBM MQ çok eşgörünümlü kuyruk yöneticisinin veri bütünlüğünü korumasını denetler. **amqsfhac** komutunu Paylaşılan dosya sistemi davranışının doğrulanmasıiçinde çalıştırmak için aşağıdaki adımları izleyin.
- *QueueManagerAd*' a bağlanırken MQCNO_RECONNECT_Q_MGR seçeneğini kullanır. Kuyruk yöneticisi başarısız olduğunda otomatik olarak yeniden bağlanır.
- *InTransactionCount*RepeatCount* kalıcı iletilerini *QueueName* kuyruğuna koyar; bu sırada kuyruk yöneticisi birçok kez başarısız olur. **amqsfhac** her seferinde kuyruk yöneticisine yeniden bağlanır ve devam eder. Sınama, hiçbir iletinin kaybolmadığından emin olmaktır.
- *InTransactionSayı* iletileri her işlem içine konmuştur. İşlem *RepeatCount* kez yinelenir. Bir hareket içinde bir hata oluşursa, **amqsfhac** kuyruk yöneticisine yeniden bağlandığında **amqsfhac** hareketi geriye işleyip yeniden sunar.
- Ayrıca, iletileri *SideQueueAd*' a da koyar. Tüm iletilerin *QueueName* kuyruğundan başarıyla kesinleştirilip kesinleştirilmediğini ya da geriye işlenip işlenmediğini denetlemek için *SideQueueName* komutunu kullanır. Bir tutarsızlık saptarsa, bir hata iletisi yazar.
- Son parametreyi (0|1|2)olarak ayarlayarak çıkış izleme miktarını **amqsfhac** değerine ayarlayın.

  0
  En azından çıktı.

  1
  Orta çıkış.

  2
  Çoğu çıktı.

İstemci bağlantısı yapılandırılması

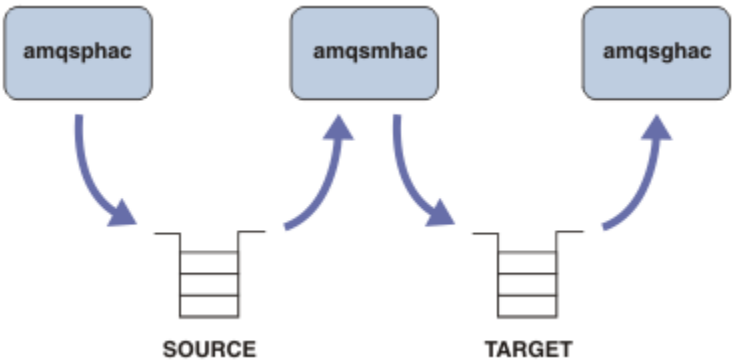
Örnekleri çalıştırmak için bir istemci ve sunucu bağlantı kanalı yapılandırmanız gerekir. İstemci doğrulama yordamı, bir istemci test ortamının nasıl ayarlanacağını açıklar.

Diğer bir seçenek olarak, aşağıdaki örnekte sağlanan yapılanışı kullanın.

amqsghac, amqsphacve amqsmhac Yöntemlerinin Kullanımı-Örnek

Bu örnek, tek bir yönetim ortamı kuyruk yöneticisini kullanarak yeniden bağlanabilir istemcileri gösterir.

İletiler, SOURCE tarafından **amqsphac**kuyruğuna yerleştirilir, **amqsmhac**tarafından TARGET 'e aktarılır ve **amqsghac** tarafından TARGET ' den alınır; bkz. Şekil 129 sayfa 1048.



Şekil 129. Yeniden bağlanabilir istemci örnekleri

Örnekleri çalıştırmak için aşağıdaki adımları izleyin.

1. Aşağıdaki komutları içeren bir `hasamples.tst` dosyası yaratın:

```
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. Bir komut istemine aşağıdaki komutları yazın:
   a. `crtmqm QM1`
   b. `strmqm QM1`
   c. `runmqsc QM1 < hasamples.tst`
3. **MQCHLLIB** ortam değişkenini `AMQCLCHL.TAB` istemci kanal tanımlama dosyasının yoluna ayarlayın; örneğin, `SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc`.
4. **MQCHLLIB** kümesiyle üç yeni pencere açın; örneğin, Windowsüzerinde, pencerelerden birinde her programı başlatan önceki komut istemine **start** üç kez yazın. “Kuyruk yöneticisinin ayarlanması ve denetlenmesi” sayfa 1048içindeki “5” sayfa 1049 adımına bakın.)
5. Kuyruk yöneticisini durdurmak için end`mqm -r -p QM1` komutunu yazın ve istemcilerin yeniden bağlanmasına izin verin.
6. Kuyruk yöneticisini yeniden başlatmak için `strmqm QM1` komutunu yazın.

amqsghac, **amqsphac**ve **amqsmhac** örneklerini Windows üzerinde çalıştırmanın sonuçları aşağıdaki örneklerde gösterilmiştir.

Kuyruk yöneticisinin ayarlanması ve denetlenmesi

1. Kuyruk yöneticisini yaratın.

```
C:\> crtmqm QM1
IBM MQ queue manager created.
Directory 'C:\IBM\MQ\MQ7\Data\qmgrs\QM1' created.
Creating or replacing default objects for QM1.
Default objects statistics : 67 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
```

   MQCHLLIB değişkenini daha sonra ayarlamak için veri dizinini hatırlayın.

2. Kuyruk yöneticisini başlatın.

```
C:\> strmqm QM1

IBM MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
IBM MQ queue manager 'QM1' started.
```

3. Kuyrukları ve kanalları oluşturun, dinleyici kapısını değiştirin ve dinleyiciyi ve kanalı başlatın.

```
C:\> runmqsc QM1 < hasamples.tst

5724-H72 (C) Copyright IBM Corp. 1994, 2024.  ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.

     1 : DEFINE QLOCAL(SOURCE) REPLACE
AMQ8006: IBM MQ queue created.
     2 : DEFINE QLOCAL(TARGET) REPLACE
AMQ8006: IBM MQ queue created.
     3 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)   TRPTYPE(TCP) MCAUSER(MUSR_MQADMIN)
REPLACE
AMQ8014: IBM MQ channel created.
     4 : DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) CONNAME('LOCALHOST(2345)')
QMNAME(QM1) REPLACE
AMQ8014: IBM MQ channel created.
     5 : ALTER LISTENER(SYSTEM.DEFAULT.LISTENER.TCP) TRPTYPE(TCP) PORT(2345)
AMQ8623: IBM MQ listener changed.
     6 : START LISTENER(SYSTEM.DEFAULT.LISTENER.TCP)
AMQ8021: Request to start IBM MQ Listener accepted.
     7 : START CHANNEL(CHANNEL1)
AMQ8018: Start IBM MQ channel accepted.
7 MQSC commands read.
No commands have a syntax error.
All valid MQSC commands were processed.
```

4. İstemci kanal çizelgesini istemcilere bildir.

   “1” sayfa 1048. adımda **crtmqm** komutundan döndürülen veri dizinini kullanın ve **MQCHLLIB** değişkenini ayarlamak için `@ipcc` dizinini ekleyin.

```
C:\> SET MQCHLLIB=C:\IBM\MQ\MQ7\Data\qmgrs\QM1\@ipcc
```

5. Diğer pencerelerdeki örnek programları başlat

```
C:\> start amqsphac SOURCE QM1
C:\> start amqsmhac -s SOURCE -t TARGET -m QM1
C:\> start amqsghac TARGET QM1
```

6. Kuyruk yöneticisini sona erdirin ve yeniden başlatın.

```
C:\> endmqm -r -p QM1

Waiting for queue manager 'QM1' to end.
IBM MQ queue manager 'QM1' ending.
IBM MQ queue manager 'QM1' ended.


C:\> strmqm QM1

IBM MQ queue manager 'QM1' starting.
5 log records accessed on queue manager 'QM1' during the log replay phase.
Log replay for queue manager 'QM1' complete.
Transaction manager state recovered for queue manager 'QM1'.
IBM MQ queue manager 'QM1' started.
```

amqsphac

```
Sample AMQSPHAC start
target queue is SOURCE
message Message 1
message Message 2
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnectedmessage
Message 3
message Message 4
message Message 5
```

amqsmhac

```
Sample AMQSMHA0 start
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
No more messages.
Sample AMQSMHA0 end
C:\>
```

amqsghac

```
Sample AMQSGHAC start
message Message 1
message Message 2
16:25:22 : EVENT : Connection Reconnecting (Delay: 0ms)
16:25:45 : EVENT : Connection Reconnecting (Delay: 0ms)
16:26:02 : EVENT : Connection Reconnected
message Message 3
message Message 4
message Message 5
```

İlgili görevler

Paylaşılan dosya sistemi davranışının doğrulanması

İlgili başvurular

amqmfsck (dosya sistemi denetimi)

Inquire örnek programları

Inquire örnek programları, MQINQ çağrısının kullanıldığı bir kuyruğun bazı özniteliklerini sorgularlar.

Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Bu programların tetiklenen programlar olarak çalışması amaçlanmıştır; bu nedenle, bu programların tek girişi IBM MQ for Multiplatformsiçin bir MQTMC2 (tetikleyici ileti) yapısıdır. Bu yapı, sorulacak özniteliklere sahip bir hedef kuyruğun adını içerir. C sürümü kuyruk yöneticisi adını da kullanır. COBOL sürümü, varsayılan kuyruk yöneticisini kullanır.

Tetikleme işleminin çalışması için, kullanmak istediğiniz sorgu örneği programının, SYSTEM.SAMPLE.INQ. Bunu yapmak için, *ApplicId* süreç tanımlamasının alanında SYSTEM.SAMPLE.INQPROCESSkullanmak istediğiniz Sorma örnek programının adını belirtin. IBM i IBM iiçin, bunun için CHGMQMPRC komutunu kullanabilirsiniz; ayrıntılar için bkz. MQ İşleminin Değiştirilmesi (CHGMQMPRC). Örnek kuyruk FIRST tetikleyici tipine sahiptir; istek örneğini çalıştırmadan önce kuyrukta önceden ileti varsa, sorgu örneği gönderdiğiniz iletiler tarafından tetiklenmez.

Tanımlamayı doğru olarak ayarladığınızda:

- ALW AIX, Linux, and Windowsiçin, **runmqtrm** programını bir oturumda başlatın ve `amqsreq` programını başka bir oturumda başlatın.

- IBM i IBM iiçin, AMQSERV4 programını bir oturumda başlatın ve AMQSREQ4 programını başka bir oturumda başlatın. AMQSERV4yerine AMQSTRG4 komutunu kullanabilirsiniz, ancak olası iş gönderimi gecikmeleri, olanların izlenmesini daha az kolaylaştıracaktır.

İstek örnek programlarını, her biri yalnızca bir kuyruk adı içeren istek iletilerini SYSTEM.SAMPLE.INQ. Her istek iletisi için, Inquire örnek programları istek iletisinde belirlenen kuyruğa ilişkin bilgileri içeren bir yanıt iletisi gönderir. Yanıtlar, istek iletisinde belirlenen yanıt kuyruğuna gönderilir.

IBM i IBM iüzerinde, örnek giriş dosyası üyesi QMQMSAMP.AMQSDATA(INQ) kullanıldı, adı geçen son kuyruk yok, bu nedenle örnek, başarısızlığa ilişkin neden kodunu içeren bir rapor iletisi döndürüyor.

Inquire örnek programının tasarımı

Program, başlatılırken iletildiği tetikleyici ileti yapısında adı belirtilen kuyruğu açar. (Daha anlaşılır olması için, buna *istek kuyruğu*diyoruz.) Program, paylaşılan giriş için bu kuyruğu açmak üzere MQOPEN çağrılmasını kullanır.

Program, iletileri bu kuyruktan kaldırmak için MQGET çağrısını kullanır. Bu çağrı, 5 saniyelik bekleme aralığıyla MQGMO_ACCEPT_TRUNCATED_MSG ve MQGMO_WAIT seçeneklerini kullanır. Program, bir istek iletisi olup olmadığını görmek için her iletinin tanımlayıcısını sınar; değilse, program iletiyi atar ve bir uyarı iletisi görüntüler.

İstek kuyruğundan kaldırılan her istek iletisi için, program kuyruğun adını okur (*hedef kuyruk* olarak adlandırılacaktır). verilerde bulunur ve MQOO_INQ seçeneğiyle MQOPEN çağrısıyla o kuyruğu açar. Bu program daha sonra, hedef kuyruğun *InhibitGet*, **CurrentQDepth**ve **OpenInputCount** özniteliklerinin değerlerini sorgulamak için MQINQ çağrılarını kullanır.

MQINQ çağrısı başarılı olursa, program yanıt kuyruğuna bir yanıt iletisi koymak için MQPUT1 çağrıyı kullanır. Bu ileti, üç özniteliğin değerlerini içerir.

MQOPEN ya da MQINQ çağrısı başarısız olursa, program yanıt kuyruğuna bir rapor iletisi koymak için MQPUT1 çağrıyı kullanır. Bu rapor iletisinin ileti tanımlayıcısının *Feedback* alanında, hangisinin başarısız olduğuna bağlı olarak MQOPEN ya da MQINQ çağrısının döndürdüğü neden kodu yer alır.

MQINQ çağrısından sonra, program MQCLOSE çağrıyı kullanarak hedef kuyruğu kapatır.

İstek kuyruğunda kalan ileti yoksa, program o kuyruğu kapatır ve kuyruk yöneticisiyle olan bağlantısını keser.

Message Handle örnek programının Inquire Properties (İleti Tanıtıcısı Özellikleri)

AMQSIQMA, ileti kuyruğundaki bir ileti tanıtıcısının özelliklerini sorgulamak için kullanılan örnek bir C programıdır ve MQINQMP API çağrısının kullanımına bir örnektir.

Bu örnek bir ileti tanıtıcısı yaratır ve bunu MQGMO yapısının MsgHandle alanına koyar. Örnek daha sonra bir ileti alır ve ileti tanıtıcısıyla doldurulan tüm özellikleri sorar ve yazdırır.

```
C:\Program Files\IBM\MQ\tools\c\Samples\Bin >amqsiqm Q QM1
Sample AMQSIQMA start
property name MyProp value MyValue
message text Hello world!
Sample AMQSIQMA end
```

Yayınlama/Abone Olma örnek programları

Yayınlama/abone olma örnek programları, IBM MQiçinde yayınlama ve abone olma özelliklerinin kullanımını gösterir.

IBM MQ yayınlama/abone olma arabirimine nasıl program yapılacağını gösteren üç C dili örnek programı vardır. Eski arabirimleri kullanan bazı C örnekleri ve Java örnekleri vardır. Java örnekleri, com.ibm.mq.jar içindeki IBM MQ yayınlama/abone olma arabirimini ve com.ibm.mqjmsiçindeki JMS yayınlama/abone olma arabirimini kullanır. JMS örnekleri bu konuda ele alınmaz.

C

C samples klasöründe amqspub yayınlayıcı örneğini bulun. İlk parametre olarak istediğiniz herhangi bir konu adıyla ve ardından isteğe bağlı bir kuyruk yöneticisi adıyla çalıştırın. Örneğin, `amqspub mytopic QM3` . `amqspubcad`lı bir istemci sürümü de vardır. İstemci sürümünü çalıştırmayı seçerseniz, ayrıntılar için bkz. “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023 .

Yayınlayıcı varsayılan kuyruk yöneticisine bağlanır ve `target topic is mytopic` çıkışıyla yanıt verir. Şu andan itibaren bu pencereye girdiğiniz her satır `mytopic` ' da yayınlanır.

Aynı dizinde başka bir komut penceresi açın ve aynı konu adını ve isteğe bağlı bir kuyruk yöneticisi adını belirterek abone programını (`amqssub`) çalıştırın. Örneğin, `amqssub mytopic QM3` .

Abone, `Calling MQGET : 30 seconds wait time` çıkışını kullanarak yanıt verir. Şu andan itibaren, yayınlayıcıya yazdığınız satırlar abonenin çıkışında görünür.

Başka bir komut penceresinde başka bir abone başlatın ve her iki abonenin de yayınları almasını izleyin.

Ayar seçenekleri de içinde olmak üzere parametrelere ilişkin eksiksiz bilgi için örnek kaynak koduna bakın. Abone seçenekleri alanının değerleri şu konuda açıklanmıştır: Options (MQLONG).

Komut satırı anahtarları olarak ek abonelik seçenekleri sunan başka bir abone örneği `amqssbx`vardır.

Abone sonlandırıldıktan sonra alıkonan sürekli abonelikleri kullanarak aboneyi çağırmak için `amqssbx -d mysub -t mytopic -k` yazın.

Yayınlayıcıyı kullanarak başka bir öğe yayınlayarak aboneliği sınayın. Abonenin sona ermesi için 30 saniye bekleyin. Aynı konu altında bazı öğeleri yayınlayın. Aboneyi yeniden başlatın. Abone çalışmamışken yayınlanan son öğe, abone tarafından hemen yeniden başlatılır.

C kalıt

Kuyruğa alınan komutları gösteren ek bir C örnekleri kümesi vardır. Bu örneklerden bazıları başlangıçta MQ0C Supportpac 'ın bir parçası olarak gönderildi. Uyumluluk nedeniyle, örneklerin gösterdiği yetenekler tam olarak desteklenir.

Kuyruktaki komut arayüzünü kullanmanızı engelliyoruz. Yayınlama/abone olma API 'sinden çok daha karmaşıktır ve karmaşık kuyruğa alınmış komutları programlamak için ikna edici bir işlevsel neden yoktur. Ancak, kuyruğa alınan yaklaşımı daha uygun bulabilirsiniz; bunun nedeni, arabirimi kullanıyor olmanızdır ya da programlama ortamınızın MQSUB 'ye farklı çağrılar oluşturmak yerine karmaşık bir ileti oluşturup soysal bir MQPUT' yi çağırmayı kolaylaştırmasıdır.

Ek örnekler, `samples` klasöründeki `pubsub` alt dizininde bulunur.

Çizelge 163 sayfa 1052içinde altı tip örnek listelenmiştir.

Çizelge 163. Eski yayınlama/abone olma örnek C programları kategorileri

Kategori	Programlar	Yorumlar
RFH1	`amqssr1a.c` `amqspr1a.c`	RFH1 biçim iletileri kullanılarak oluşturulan basit yayınlama/abone olma örneği.
RFH2	`amqssr2a.c` `amqspr2a.c`	RFH2 biçim iletileri kullanılarak oluşturulan basit yayınlama/abone olma örneği.
MQAI örnekleri	`amqsppca.c` `amqsspca.c`	PCF komutları ve MQAI komut arabirimi kullanılarak oluşturulan basit yayınlama/abone olma örneği.

Çizelge 163. Eski yayınlama/abone olma örnek C programları kategorileri (devamı var)		
Kategori	**Programlar**	**Yorumlar**
MA0C RFH1 kullanan sonuç hizmeti	`amqsgama.c` `amqsresa.c`	RFH1 üstbilgileri kullanılarak oluşturulan sonuç hizmeti 1. `amqsgama.tst` ve `amqsresa.tst` içinde tanımlanan kuyrukları gerektirir 2. `amqsresa` bundan önce başlatılmalıdır `amqsgama`
MA0C RFH2 kullanan sonuç hizmeti	`amqsgr2a.c` `amqsrr2a.c`	RFH2 üstbilgileri kullanılarak oluşturulan sonuç hizmeti 1. `amqsgama.tst` ve `amqsresa.tst` içinde tanımlanan kuyrukları gerektirir 2. `amqsresa` bundan önce başlatılmalıdır `amqsgama`
Yöneltme çıkışı yayınlama /abone olma örneği	`amqspsra.c`	Bir yöneltme çıkışındaki yayınlama/abone olma iletisine ilişkin kuyruk ya da kuyruk yöneticisi hedefinin nasıl değiştirileceğini gösterir.

Java için örnek program

Java Örnek `MQPubSubApiSample.java` , yayınlayıcıyı ve aboneleri tek bir programda birleştirir. Kaynak ve derlenmiş sınıf dosyaları, `wmqjava` örnekler klasöründe bulunur.

İstemci kipinde çalıştırmayı seçerseniz, ayrıntılar için bkz. “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023 .

Yapılandırılmış bir Java ortamınız varsa, Java komutunu kullanarak komut satırından örneği çalıştırın. Örneği, Java programlama çalışma ortamı önceden ayarlanmış olan IBM MQ Explorer Eclipse çalışma alanından da çalıştırabilirsiniz.

Çalıştırmak için örnek programın bazı özelliklerini değiştirmeniz gerekebilir. Bunu, JVM ' ye parametre sağlayarak ya da kaynağı düzenleyerek yaparsınız.

“MQPubSubApiSample Java örneğinin çalıştırılması” sayfa 1053 içindeki yönergeler, örneğin Eclipse çalışma alanından nasıl çalıştırılacağını gösterir.

MQPubSubApiSample Java örneğinin çalıştırılması
MQPubSubApiSample , Eclipse platformundan Java Development Tools kullanılarak nasıl çalıştırılır?

Başlamadan önce

Eclipse çalışma ortamını açın. Yeni bir çalışma alanı dizini yaratın ve seçin. Hoş geldiniz penceresini kapatın.

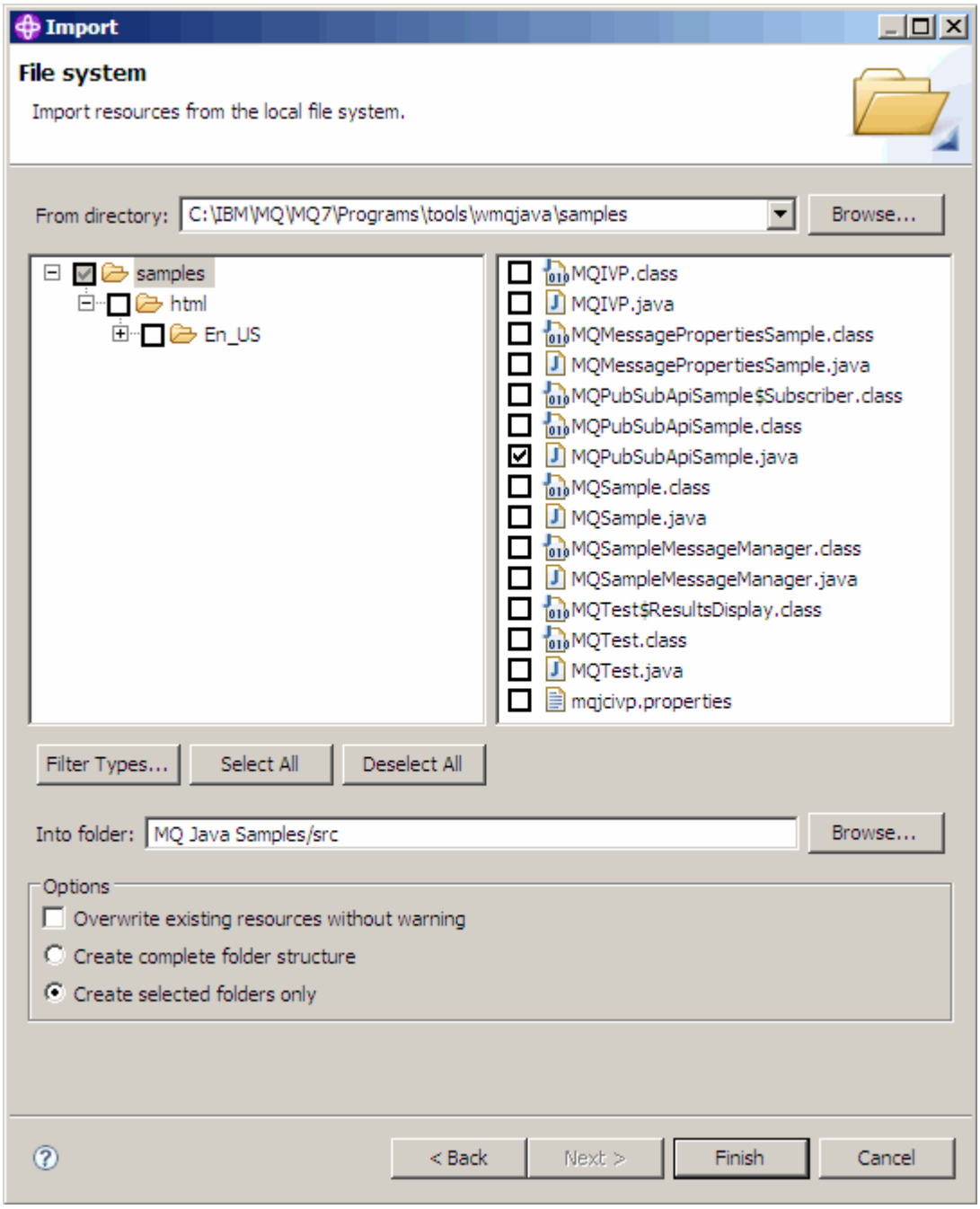
İstemci olarak çalıştırılmadan önce “Çoklu platformlarda istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması” sayfa 1023 başlıklı konudaki adımları izleyin.

Bu görev hakkında

Java yayınlama/abone olma örnek programı bir IBM MQ MQI client Java programıdır. Örnek, 1414 numaralı kapıda dinleyen varsayılan bir kuyruk yöneticisi kullanılarak değişiklik yapılmadan çalışır. Görev, bu basit durumu açıklar ve genel olarak parametrelerin nasıl sağlanacağını ve örneğin farklı IBM MQ yapılandırmalarına uyacak şekilde nasıl değiştirileceğini gösterir. Örnek, Windowsüzerinde çalışırken gösterilir. Dosya yolları diğer platformlarda farklılık gösterir.

Yordam

1. Java örnek programlarını içe aktarma
   a) Çalışma ortamında **Pencere** > **perspektifi aç** > **Diğer** > **Java** seçeneğini tıklatın ve **Tamam**' ı tıklatın.
   b) **Paket Gezgini** görünümüne geçin.
   c) **Paket Gezgini** görünümünde beyaz alanı sağ tıklatın. **Yeni** > **Java proje**öğesini tıklatın.
   d) **`Project name`** alanına `MQ Java Samples`yazın. **İleri**düğmesini tıklatın.
   e) **`Java Settings`** panosunda **Kitaplıklar** sekmesine geçin.
   f) **Dış JAR Ekle**öğesini tıklatın.
   g) *MQ_INSTALLATION_PATH* `\java\lib` ' a göz atın; burada *MQ_INSTALLATION_PATH* , IBM MQ kuruluş klasörüdür ve `com.ibm.mq.jar` ve `com.ibm.mq.jmqi.jar` öğelerini seçin.
   h) **Aç** > **Son** düğmesini tıklatın.
   i) **Paket Gezgini** görünümünde `src` öğesini sağ tıklatın.
   j) Seç **İçe Aktar ...** > **Genel** > **Dosya Sistemi** > **Sonraki** > **Göz At**... ve *MQ_INSTALLATION_PATH* `\tools\wmqjava\samples` yoluna göz atın; burada *MQ_INSTALLATION_PATH* , IBM MQ kuruluş dizinidir.
   k) **Import** (İçe Aktar) panosunda Şekil 130 sayfa 1055, `samples` (onay kutusunu seçmeyin) seçeneğini tıklatın.
   l) `MQPubSubApiSample.java`seçeneğini belirleyin. **`Into folder`** alanı `MQ Java Samples/src`içermelidir. **Bitir**'i tıklatın.



Şekil 130. Dosya sistemi içe aktarma

2. Yayınlama/abone olma örnek programını çalıştırın.

   Varsayılan parametreleri değiştirmeniz gerekip gerekmediğine bağlı olarak programı çalıştırmanın iki yolu vardır.

   - İlk seçim, programı herhangi bir değişiklik yapmadan çalıştırır:
     - Çalışma alanı ana menüsünde `src` klasörünü genişletin. **MQPubSubApiSample.java** öğesini sağ tıklatın. **Şu şekilde çalıştır** > **1. Java Uygulama**
   - İkinci seçenek, programı parametrelerle ya da ortamınız için değiştirilmiş kaynak kodla çalıştırır:
     - `MQPubSubApiSample.java` dosyasını açın ve `MQPubSubApiSample` oluşturucusunu araştırın.
     - Programın özniteliklerini değiştirin.

Bu öznitelikler, -D JVM anahtarı kullanılarak ya da kaynak kodu düzenleyerek System property için varsayılan bir değer sağlanarak değiştirilebilir.

- topicObject
- queueManagerAdı
- subscriberCount

Bu öznitelikler yalnızca oluşturucuda kaynak kod düzenlenerek değiştirilebilir.

- hostname
- kapı
- kanal

System properties değerini ayarlamak için erişimcide varsayılan bir değer kodlayın; örneğin:

```
queueManagerName = System.getProperty("com.ibm.mq.pubSubSample.queueManagerName",
"QM3");
```

Ya da aşağıdaki adımlarda gösterildiği gibi, `-D` seçeneğini kullanarak JVM ' ye parametreyi sağlayın:

a. Ayarlamak istediğiniz System.Property özelliğinin tam adını kopyalayın, örneğin: `com.ibm.mq.pubSubSample.queueManagerName`.

b. Çalışma alanında **Çalıştır** > **Çalıştır İletişim Kutusunu Aç**seçeneğini sağ tıklatın. **Uygulamaları oluştur, Yönet ve Çalıştır** içindeki Java Uygulamayı çift tıklatın ve **(x) = Bağımsız Değişkenler** sekmesini tıklatın.

c. **VM bağımsız değişkenleri:** bölmesine `-D` yazın ve System.property adını `com.ibm.mq.pubSubSample.queueManagerName`, ardından `=QM3`yazın. **Uygula** > **Çalıştır**düğmesini tıklatın.

d. Virgülle ayrılmış bir liste olarak ya da bölmede virgül ayırıcıları olmadan ek satırlar olarak başka bağımsız değişkenler ekleyin.

Örneğin: `-Dcom.ibm.mq.pubSubSample.queueManagerName=QM3`, `-Dcom.ibm.mq.pubSubSample.subscriberCount=6`.

Yayınlama Çıkışı örnek programı

AMQSPSE0 , bir yayın aboneye teslim edilmeden önce yayını kesmek için çıkışın örnek bir C programıdır. Çıkış, örneğin, ileti üstbilgilerini, bilgi yükünü ya da hedefini değiştirebilir ya da iletinin bir aboneye yayınlanmasını önleyebilir.

Örneği çalıştırmak için aşağıdaki görevleri gerçekleştirin:

1. Kuyruk yöneticisini yapılandırın:
   - Linux AIX AIX and Linux sistemlerinde `qm.ini` kütüğüne buna benzer bir kıta ekleyin:

     ```
     PublishSubscribe:
     PublishExitPath=Module
     PublishExitFunction=EntryPoint
     ```

     burada modül *MQ_INSTALLATION_PATH*`/samp/bin/amqspse`' dir. *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.
   - Windows Windows üzerinde, kayıttaki eşdeğer öznitelikleri ayarlayın.
2. Modülün IBM MQtarafından erişilebilir olduğundan emin olun.
3. Yapılandırmayı almak için kuyruk yöneticisini yeniden başlatın.
4. İzlenecek uygulama işleminde, izleme dosyalarının nereye yazılması gerektiğini tanımlayın. Örneğin:

- Linux AIX AIX and Linux sistemlerinde `/var/mqm/trace` dizininin var olduğundan emin olun ve **MQPSE_TRACE_LOGFILE** ortam değişkenini dışa aktarın:

```
export MQPSE_TRACE_LOGFILE=/var/mqm/trace/PubTrace
```

- Windows Windowssistemlerinde `C:\temp` dizininin var olduğundan emin olun ve **MQPSE_TRACE_LOGFILE** ortam değişkenini ayarlayın:

```
set MQPSE_TRACE_LOGFILE=C:\temp\PubTrace
```

Put örnek programları

Put örnek programları, MQPUT çağrılarını kullanarak iletileri kuyruğa koyar.

Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Put örnek programının tasarımı

Program, iletileri koymak üzere hedef kuyruğu açmak için MQOO_OUTPUT seçeneğiyle MQOPEN çağrısıyla birlikte kullanılır.

Kuyruğu açamazsa, program MQOPEN çağrısıyla döndürülen neden kodunu içeren bir hata iletisi verir. Programı basit tutmak için, bu ve sonraki MQI çağrılarında, program birçok seçenek için varsayılan değerleri kullanır.

Her giriş satırı için, program metni bir arabelleğe okur ve o satırın metnini içeren bir veri paketi iletisi yaratmak için MQPUT çağrıyı kullanır. Program, girişin sonuna ulaşıncaya ya da MQPUT çağrısı başarısız oluncaya kadar devam eder. Program girişin sonuna ulaşırsa, MQCLOSE çağrıyı kullanarak kuyruğu kapatır.

Put örnek programlarını çalıştırma

amqsput ve amqsputc örneklerinin çalıştırılması

ALW

amqqsput örneği, iletileri yerel bağ tanımlarını kullanarak yerleştirmek için kullanılan programdır ve **amqsputc** örneği, iletileri istemci bağ tanımlarını kullanarak yerleştirmek için kullanılan programdır. Bu programların her biri aşağıdaki konumsal parametreleri alır:

1. Hedef kuyruğun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)

   Bir kuyruk yöneticisi belirtilmezse, **amqsput** varsayılan kuyruk yöneticisine bağlanır ve **amqsputc** , *MQSERVER* ortam değişkeniyle ya da istemci kanal tanımlama dosyasıyla tanıtılan kuyruk yöneticisine bağlanır.
3. Açma seçenekleri (isteğe bağlı)

   Açma seçenekleri belirtilmezse, örnek şu iki seçeneğin birleşiminden oluşan 8208 değerini kullanır:

   - MQOO_ÇıKışı
   - MQOO_FAIL_IF_QUIESCING
4. Kapatma seçenekleri (isteğe bağlı)

   Kapatma seçenekleri belirtilmezse, örnek 0 değerini kullanır (MQCO_NONE).
5. Hedef kuyruk yöneticisinin adı (isteğe bağlı)

   Hedef kuyruk yöneticisi belirtilmezse, MQOD ' daki `ObjectQMgrName` alanı boş bırakılır.
6. Dinamik kuyruğun adı (isteğe bağlı)

Dinamik bir kuyruk adı belirtilmezse, MQOD içindeki `DynamicQName` alanı boş bırakılır.

Kuyruk yöneticisiyle kimlik doğrulaması yapmak için kullanılan kimlik bilgilerini sağlamak üzere aşağıdaki ortam değişkenlerini kullanın:

MQSAMP_USER_ID
Kuyruk yöneticisiyle kimlik doğrulaması yapmak için bir kullanıcı kimliği ve parola kullanmak istiyorsanız, bağlantı kimlik doğrulaması için kullanılacak kullanıcı kimliğini ayarlayın. Program, kullanıcı kimliğiyle birlikte parolanın da girmesini ister.

Linux AIX V 9.3.4 **MQSAMP_TOKEN**
Kuyruk yöneticisiyle kimlik doğrulaması için bir kimlik doğrulama belirteci sağlamak istiyorsanız, boş olmayan bir değere ayarlayın. Program, kimlik doğrulama belirtecini ister. Kimlik doğrulama belirteçleri yalnızca istemci bağ tanımlarını kullanan **amqsputc** örneği tarafından kullanılabilir.

Bu programları çalıştırmak için aşağıdaki komutlardan birini girin:

- `amqsput` *`myqueue qmanagername`*
- `amqsputc` *`myqueue qmanagername`*

Burada *kuyruğum* , iletilerin konacağı kuyruğun adıdır ve *qmanagername* , *kuyruğum*' un sahibi olan kuyruk yöneticisidir.

amq0put örneğinin çalıştırılması

ALW

COBOL sürümünde parametre yok. Varsayılan kuyruk yöneticisine bağlanır ve çalıştırdığınızda size sorulur:

```
Please enter the name of the target queue
```

StdIn ' den giriş alır ve her giriş satırını hedef kuyruğa ekler. Boş bir satır, başka veri olmadığını gösterir.

AMQSPUT4 C örneğinin çalıştırılması (IBM i)

IBM i

Yalnızca IBM i platformu için kullanılabilen C programı AMQSPUT4, kaynak dosyanın bir üyesinden veri okuyarak ileti yaratır.

Programı başlattığınızda, dosyanın adını parametre olarak belirlemeniz gerekir. Dosyanın yapısı şu olmalıdır:

```
queue name
text of message 1
text of message 2
⋮
text of message n
blank line
```

QMQMSAMP AMQSDATA dosyası PUT kitaplığında, koyma örneklerine ilişkin bir giriş örneği sağlandı.

Not: Kuyruk adlarının büyük ve küçük harfe duyarlı olduğunu unutmayın. AMQSAMP4 örnek dosya yaratma programı tarafından oluşturulan tüm kuyrukların adları büyük harfli karakterlerle oluşturulmuştur.

C programı, dosyanın ilk satırında adı belirtilen kuyruğa ileti koyar; sağlanan SYSTEM.SAMPLE.LOCAL. Program, dosyanın aşağıdaki satırlarından her birinin metnini ayrı veri paketi iletilerine koyar ve dosyanın sonunda boş bir satır okuduğunda durur.

Örnek veri dosyası kullanılarak komut aşağıdaki gibidir:

```
CALL PGM(QMQM/AMQSPUT4) PARM('QMQMSAMP/AMQSDATA(PUT)')
```

AMQ0PUT4 COBOL örneğini çalıştırma (IBM i)

IBM i

Yalnızca IBM i platformunda kullanılabilen COBOL programı AMQ0PUT4, klavyeden veri kabul ederek iletiler oluşturur.

Programı başlatmak için programı çağırın ve program parametresi olarak hedef kuyruğunuzun adını verin. Program, klavyeden arabelleğe giriş kabul eder ve her metin satırı için bir veri paketi iletisi yaratır. Klavyede boş bir satır girdiğinizde program durur.

Reference Message örnek programları

Başvuru İletisi örnekleri, büyük bir nesnenin kaynak ya da hedef düğümlerde IBM MQ kuyruklarında saklanmasına gerek kalmadan bir düğümden diğerine (genellikle farklı sistemlerde) aktarılmasını sağlar.

Başvuru İletilerinin bir kuyruğa nasıl konulabileceğini, ileti çıkışları tarafından nasıl alınabileceğini ve kuyruktan nasıl alınabileceğini göstermek için bir dizi örnek program sağlanır. Örnek programlar, dosyaları taşımak için Başvuru İletilerini kullanır. Veritabanları gibi diğer nesneleri taşımak istiyorsanız ya da güvenlik denetimleri gerçekleştirmek istiyorsanız, örneğe (amqsxrm) dayalı olarak kendi çıkışınızı tanımlayın.

Kullanılacak Reference Message çıkış örnek programının sürümü, kanalın çalıştığı platforma bağlıdır:

- Tüm platformlarda, gönderme ucunda amqsxrma kullanın.
- Alıcı IBM idışında herhangi bir platform altında çalışıyorsa, alma ucunda amqsxrma kullanın.
- IBM i Günlük nesnesi IBM ialtında çalışıyorsa, amqsxrm4komutunu kullanın.

.

IBM i *IBM i kullanıcıları için notlar*

Örnek ileti çıkışını kullanarak bir Başvuru İletisi almak için, IFS ' nin kök dosya sisteminde ya da herhangi bir altdizinde bir dosya belirleyin; böylece, akış dosyası yaratılabilir.

IBM i üzerindeki örnek ileti çıkışı kütüğü yaratır, verileri EBCDIC ' ye dönüştürür ve kod sayfasını sistem kod sayfanıza ayarlar. Daha sonra bu dosyayı QSYS.LIB dosya sistemi CPYFRMSTMF komutunu kullanarak. Örneğin:

```
CPYFRMSTMF FROMSTMF('JANEP/TEST.TXT')
TOMBR('qsys.lib.janep.lib/test.fie/test.mbr') MBROPT(*REPLACE)
CVTDTA(*NONE)
```

CPYFRMSTMF komutu dosyayı oluşturmaz. Bu komutu çalıştırmadan önce yaratmalısınız.

QSYS.LIB, örneklerde değişiklik gerekmez. Diğer dosya sistemlerinde, MQRMH yapısındaki CodedCharSetId alanında belirtilen CCSID değerinin, göndermekte olduğunuz toplu verilerle eşleştiğini doğrulayın.

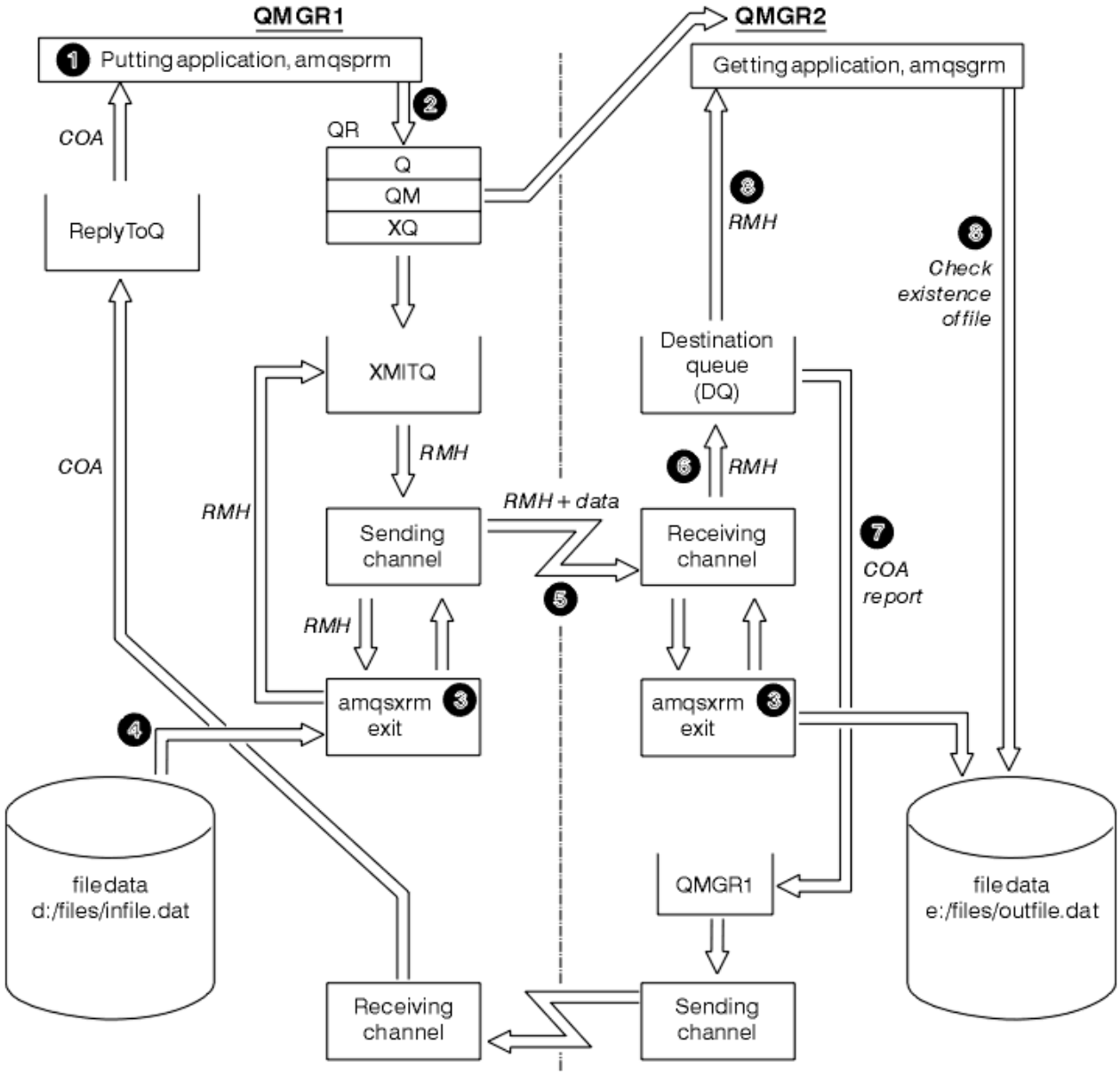
Tümleşik dosya sistemini kullanırken, SYSIFCOPT (*IFSIO) seçenek takımıyla program birimleri yaratın. Veritabanı ya da sabit uzunluklu kayıt dosyalarını taşımak istiyorsanız, sağlanan AMQSXRM4örneğine dayalı olarak kendi çıkışınızı tanımlayın.

Bir veri tabanı dosyasını aktarmak için önerilen yöntem, dosyayı CPYTOSTMF komutunu kullanarak IFS yapısına dönüştürmek ve ardından IFS dosyasını ekleyen Başvuru İletisi 'ni göndermektir. Bir veri tabanı dosyasını IFS içinden aktarmayı seçerseniz, ancak IFS yapısına dönüştürmezseniz, üye adını belirtmeniz gerekir. Bu yöntemi seçerseniz, veri bütünlüğü garanti edilmez.

Multi *Başvuru İletisi örneklerinin çalıştırılması*

AIX, Linux, and Windowsüzerinde Başvuru İletisi örnek uygulaması AMQSPRM 'nin ya da IBM üzerinde AMQSPRMA' nın nasıl çalıştırılacağını öğrenmek için bu örneği kullanın. Bu örnekte, Başvuru İletilerinin bir kuyruğa nasıl konabileceği, ileti çıkışları tarafından nasıl alınabileceği ve kuyruktan nasıl alınabileceği gösterilmektedir.

Başvuru İletisi örnekleri aşağıdaki gibi çalışır:



Şekil 131. Başvuru İletisi örneklerinin çalıştırılması

1. Ortamı, dinleyicileri, kanalları ve tetikleyicileri başlatabileceğiniz ve kanallarınızı ve kuyruklarınızı tanımlayabileceğiniz şekilde ayarlayın.

   Bu örnek, Başvuru İletisinin nasıl ayarlanacağını açıklamak amacıyla, gönderen makineyi QMGR1 adlı kuyruk yöneticisiyle MACHINE1 olarak ve alıcı makineyi QMGR2adlı kuyruk yöneticisiyle MACHINE2 olarak ifade eder.

   Not: Aşağıdaki tanımlar, QMGR1 kuyruk yöneticisinden QMGR2 ' ye nesne tipi FLATFILE olan bir dosyayı göndermek ve dosyayı AMQSPRM (ya da IBM iüzerinde AMQSPRMA)çağrısında tanımlandığı şekilde yeniden yaratmak için bir Başvuru İletisi oluşturulmasını sağlar. Başvuru İletisi (dosya verileri de içinde olmak üzere), CHL1 kanalı ve XMITQ iletim kuyruğu kullanılarak gönderilir ve DQ kuyruğuna yerleştirilir. Kural dışı durum ve COA raporları, QMGR1kanal REPORT ve iletim kuyruğu kullanılarak QMGR1 ' e geri gönderilir.

   Başvuru İletisini (AMQSGRM ya da IBM i üzerinde AMQSGRMA) alan uygulama, başlatma kuyruğu INITQ ve süreç PROC kullanılarak tetiklenir. CONNAME alanlarının doğru ayarlandığından ve MSGEXIT alanının, makine tipine ve IBM MQ ürününün kurulduğu yere bağlı olarak dizin yapınızı yansıttığınızdan emin olun.

IBM i MQSC tanımlamaları çıkışları tanımlamak için AIX biçemini kullandığından IBM iüzerinde MQSC kullanıyorsanız, bunları buna göre değiştirmeniz gerekir. FLATFILE ileti verilerinin büyük ve küçük harfe duyarlı olduğunu ve büyük harfli olmadıkça örneğin çalışmadığını göz önünde bulundurmanız önemlidir.

MACHINE1makinesinde, kuyruk yöneticisi QMGR1

MQSC sözdizimi

```
define chl(chl1) chltype(sdr) trptype(tcp) conname('machine2') xmitq(xmitq)
msgdata(FLATFILE) msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)
')

define ql(xmitq) usage(xmitq)

define chl(report) chltype(rcvr) trptype(tcp) replace

define qr(qr) rname(dq) rqmname(qmgr2) xmitq(xmitq) replace
```

IBM i **IBM i komut sözdizimi**

Not: Bir kuyruk yöneticisi adı belirtmezseniz, sistem varsayılan kuyruk yöneticisini kullanır.

```
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)
```

MACHINE2makinesinde, kuyruk yöneticisi QMGR2

MQSC sözdizimi

```
define chl(chl1) chltype(rcvr) trptype(tcp)
msgexit('/usr/lpp/mqm/samp/bin/amqsxrm(MsgExit)')
        msgdata(flatfile)

define chl(report) chltype(sdr) trptype(tcp) conname('MACHINE1')
        xmitq(qmgr1)

define ql(initq)

define ql(qmgr1) usage(xmitq)

define pro(proc) applicid('/usr/lpp/mqm/samp/bin/amqsgrm')

define ql(dq) initq(initq) process(proc) trigger trigtype(first)
```

IBM i **IBM i komut sözdizimi**

Not: IBM i Açık IBM i: Bir kuyruk yöneticisi adı belirtmezseniz, sistem varsayılan kuyruk yöneticisini kullanır.

```
CRTMQMCHL CHLNAME(CHL1) CHLTYPE(*RCVR) MQMNAME(QMGR2) +
REPLACE(*YES) TRPTYPE(*TCP) +
MSGEXIT(QMQM/AMQSXRM4) MSGUSRDATA(FLATFILE)

CRTMQMCHL CHLNAME(REPORT) CHLTYPE(*SDR) MQMNAME(QMGR2) +
REPLACE(*YES) TRPTYPE(*TCP) +
CONNAME('MACHINE1(60500)') TMQNAME(QMGR1)
```

```
CRTMQMQ  QNAME(INITQ) QTYPE(*LCL) MQMNAME(QMGR2) +
REPLACE(*YES) USAGE(*NORMAL)

CRTMQMQ  QNAME(QMGR1) QTYPE(*LCL) MQMNAME(QMGR2) +
REPLACE(*YES) USAGE(*TMQ)

CRTMQMPRC PRCNAME(PROC) MQMNAME(QMGR2) REPLACE(*YES) +
APPID('QMQM/AMQSGRM4')

CRTMQMQ  QNAME(DQ) QTYPE(*LCL) MQMNAME(QMGR2) +
REPLACE(*YES) PRCNAME(PROC) TRGENBL(*YES) +
INITQNAME(INITQ)
```

2. IBM MQ nesneleri yaratıldıktan sonra:
   a. Platform için geçerliyse, gönderen ve alan kuyruk yöneticileri için dinleyiciyi başlatın
   b. CHL1 ve REPORT kanallarını başlat
   c. Alıcı kuyruk yöneticisinde, INITQ başlatma kuyruğuna ilişkin tetikleyici izleyicisini başlatın.
3. Aşağıdaki parametreleri kullanarak komut satırından koyma Başvuru İletisi örnek programı AMQSPRM (IBM i üzerinde AMQSPRMA) ' yi çağırın:

   -m
   Yerel kuyruk yöneticisinin adı; varsayılan olarak varsayılan kuyruk yöneticisi kullanılır

   -i
   Kaynak dosyanın adı ve yeri

   -o
   Hedef dosyanın adı ve yeri

   -q
   Kuyruğun adı

   -g
   -q parametresinde tanımlanan kuyruğun bulunduğu kuyruk yöneticisinin adı. Varsayılan değer olarak, -m değiştirgesinde belirtilen kuyruk yöneticisi kullanılır.

   -t
   Nesne tipi

   -w
   Bekleme aralığı; kural dışı durum için bekleme süresi ve alan kuyruk yöneticisinden COA raporları

   Örneğin, örneği daha önce tanımlanan nesnelerle kullanmak için aşağıdaki parametreleri kullanabilirsiniz:

```
-mQMGR1 -iInput File -oOutput File -qQR -tFLATFILE -w120
```

   Bekleme süresini artırmak, program iletileri zamanaşımına uğramadan önce büyük bir dosyanın ağ üzerinden gönderilmesine izin verir.

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

   IBM i kullanıcıları: IBM i IBM iişletim sistemlerinde aşağıdaki adımları tamamlayın:

   a. Aşağıdaki komutu kullanın:

```
CALL    PGM(QMQM/AMQSPRM4) PARM('-mQMGR1' +
'-i/refmsgs/rmsg1' +
'-o/refmsgs/rmsgx' '-qQR' +
'-gQMGR1' '-tFLATFILE' '-w15')
```

      Bu işlem, özgün `rmsg1` dosyasının `/refmsgs` IFS dizininde olduğunu ve hedef dosyanın hedef sistemdeki `rmsgx` IFS dizininde `/refmsgs` olmasını istediğinizi varsayar.

   b. Kök dizini kullanmak yerine CRTDIR komutunu kullanarak kendi dizininizi yaratın.

c. Veri yerleştiren programı çağırdığınızda, çıkış dosyası adının IFS adlandırma kuralını yansıtması gerektiğini unutmayın; örneğin, /TEST/FILENAME, TEST dizininde FILENAME adlı bir dosya yaratır.

Not:

IBM i IBM iüzerinde, parametreleri belirtirken eğik çizgi (/) ya da kısa çizgi (-) kullanabilirsiniz. Örneğin:

```
amqsprm /i d:\files\infile.dat /o e:\files\outfile.dat /q QR
/m QMGR1 /w 30 /t FLATFILE
```

Linux AIX AIX and Linux altyapılarında, hedef dosya dizinini belirtmek için biri yerine iki ters eğik çizgi (\\) kullanmalısınız. Bu nedenle, **amqsprm** komutu şöyle görünür:

```
amqsprm -i /files/infile.dat -o e:\\files\\outfile.dat -q QR
-m QMGR1 -w 30 -t FLATFILE
```

Başvuru İletisi koyma programının çalıştırılması aşağıdaki işlemleri gerçekleştirir:

- Başvuru İletisi, QMGR1kuyruk yöneticisinde QR kuyruğuna yerleştirilir.
- Kaynak dosya ve yol `d:\files\infile.dat` ve örnek komutun verildiği sistemde var.
- QR kuyruğu uzak bir kuyruksa, Başvuru İletisi, `e:\files\outfile.dat`adı ve yoluyla bir dosyanın yaratıldığı farklı bir sistemde başka bir kuyruk yöneticisine gönderilir. Bu dosyanın içeriği kaynak dosyayla aynı.
- amqsprm, hedef kuyruk yöneticisinden bir COA raporu için 30 saniye bekler.
- Nesne tipi `flatfile`olduğundan, iletileri QR kuyruğundan taşımak için kullanılan kanal bunu *MsgData* alanında belirtmelidir.

4. Kanallarınızı tanımladığınızda, hem gönderme hem de alma uçlarındaki ileti çıkışını amqsxrm olacak şekilde seçin.

   Windows Bu, Windows üzerinde aşağıdaki gibi tanımlanır:

   ```
   msgexit(' pathname\amqsxrm.dll(MsgExit)')
   ```

   Linux AIX Bu, AIX and Linux üzerinde aşağıdaki gibi tanımlanır:

   ```
   msgexit(' pathname/amqsxrm(MsgExit)')
   ```

   Bir yol adı belirtirseniz, tam adı belirtin. Yol adını atlarsanız, programın `qm.ini` kütüğünde (ya da IBM MQ for Windowsüzerinde, kayıtta belirtilen yolda) belirtilen yolda olduğu varsayılır.
5. Kanal çıkışı, Başvuru İletisi üstbilgisini okur ve başvurduğu dosyayı bulur.
6. Kanal çıkışı, üstbilgiyle birlikte kanalı aşağı göndermeden önce dosyayı bölebilir.

   Linux AIX AIX and Linux'da, örnek ileti çıkışının dosyayı o dizinde yaratabilmesi için hedef dizinin grup sahibini' mqm ' olarak değiştirin. Ayrıca, hedef dizinin izinlerini değiştirerek, mqm grubu üyelerinin bu dizine yazmalarına izin verin. Dosya verileri IBM MQ kuyruklarında saklanmaz.
7. Dosyanın son bölümü alıcı ileti çıkışı tarafından işlendiğinde, Başvuru İletisi amqsprm tarafından belirlenen hedef kuyruğa yerleştirilir. Bu kuyruk tetiklenirse (tanım **Trigger**, **InitQ**ve **Process** kuyruk özniteliklerini belirtiyorsa), hedef kuyruğun PROC parametresi tarafından belirlenen program tetiklenir. Tetiklenecek program, **Process** özniteliğinin `ApplId` alanında tanımlanmalıdır.
8. Başvuru İletisi hedef kuyruğa (DQ) ulaştığında, bir COA raporu koyma uygulamasına (amqsprm) geri gönderilir.
9. Başvuru İletisini Al örneği (amqsgrm), giriş tetikleyicisi iletisinde belirtilen kuyruktan iletileri alır ve dosyanın varlığını denetler.

Put Reference Message Sample (amqsprma.c, AMQSPRM4)
Bu konuda, bir Put Reference Message örneğinin ayrıntılı bir açıklaması verilir.

Bu örnek, bir dosyaya gönderme yapan ve dosyayı belirtilen bir kuyruğa koyan bir Başvuru İletisi yaratır:

1. Örnek, MQCONN kullanarak yerel bir kuyruk yöneticisine bağlanır.
2. Daha sonra rapor iletilerini almak için kullanılan bir model kuyruğunu açar (MQOPEN).
3. Örnek, dosyayı taşımak için gereken değerleri (örneğin, kaynak ve hedef dosya adları ve nesne tipi) içeren bir Başvuru İletisi oluşturur. Örneğin, IBM MQ ile verilen örnek, `d:\x\file.in` dosyasını `QMGR1` 'den `QMGR2` ' e göndermek ve dosyayı `d:\y\file.out` olarak yeniden oluşturmak için aşağıdaki parametreleri kullanarak bir Başvuru İletisi oluşturur:

   ```
   amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
   ```

   Burada `QR` , `QMGR2`üzerindeki bir hedef kuyruğa başvuran uzak kuyruk tanımlamasıdır.

   Not: AIX and Linux altyapılarında, hedef dosya dizinini belirtmek için biri yerine iki ters eğik çizgi (\\) kullanın. Bu nedenle, **amqsprm** komutu şöyle görünür:

   ```
   amqsprm -q QR -m QMGR1 -i /x/file.in -o d:\\y\\file.out -t FLATFILE
   ```
4. Başvuru İletisi, /q parametresiyle belirlenen kuyruğa yerleştirilir (dosya verisi olmadan). Bu bir uzak kuyruksa, ileti ilgili iletim kuyruğuna yerleştirilir.
5. Örnek, kural dışı durum raporlarıyla birlikte yerel kuyruk yöneticisinde (QMGR1) yaratılan dinamik kuyruğa geri gönderilen COA raporları için, /w parametresinde belirtilen süre boyunca (varsayılan değer olarak 15 saniye) bekler.

Başvuru İletisi Çıkışı örneği tasarımı (amqsxrma.c, AMQSXRM4)
Bu örnek, kanal tanımlamasının ileti çıkışı kullanıcı verileri alanındaki nesne tipiyle eşleşen bir nesne tipine sahip Başvuru İletilerini tanır.

Bu iletiler için aşağıdaki durum oluşur:

- Gönderen ya da sunucu kanalında, belirtilen veri uzunluğu, belirtilen dosyanın belirtilen göreli konumundan, Başvuru İletisinden sonra aracı arabelleğinde kalan alana kopyalanır. Dosyanın sonuna ulaşılmazsa, Başvuru İletisi *DataLogicalOffset* alanı güncellendikten sonra iletim kuyruğuna geri yerleştirilir.
- İstekte bulunan ya da alıcı kanalında *DataLogicalOffset* alanı sıfırsa ve belirtilen dosya yoksa, yaratılır. Başvuru İletisini izleyen veriler, belirtilen dosyanın sonuna eklenir. Başvuru İletisi belirtilen dosya için son ileti değilse, atılır. Tersi durumda, hedef kuyruğa eklenecek veriler olmadan kanal çıkışına döndürülür.

Gönderen ve sunucu kanalları için, giriş Başvuru İletisi içindeki *DataLogicalLength* alanı sıfırsa, dosyanın geri kalan kısmı *DataLogicalOffset* ' dan dosyanın sonuna kadar kanal boyunca gönderilir. Sıfır değilse, yalnızca belirtilen uzunluk gönderilir.

Bir hata oluşursa (örneğin, örnek bir dosyayı açamazsa), MQCXP. *ExitResponse* , işlenmekte olan iletinin hedef kuyruğa devam etmek yerine teslim edilmeyen ileti kuyruğuna konması için MQXCC_SUPPRESS_FUNCTION olarak ayarlanır. MQCXP ' de bir geribildirim kodu döndürülür. *Feedback* ve iletiyi bir rapor iletisinin ileti tanımlayıcısının *Feedback* alanına koyan uygulamaya geri döndü. Bunun nedeni, koyma uygulamasının MQMD ' nin *Report* alanında MQRO_EXCEPTION ayarlanarak kural dışı durum raporlarını istemesi olabilir.

Başvuru İletisinin kodlaması ya da *CodedCharacterSetId* (CCSID) değeri kuyruk yöneticisinden farklıysa, Başvuru İletisi yerel kodlamaya ve CCSID ' ye dönüştürülür. Örneğimizde, amqsprm, nesnenin biçimi MQFMT_STRING 'dir, bu nedenle amqsxrm, veriler dosyaya yazılmadan önce nesne verilerini alma ucunda yerel CCSID' ye dönüştürür.

Dosya çok baytlı karakterler (örneğin, DBCS ya da Unicode) içeriyorsa, aktarılmakta olan dosyanın biçimini MQFMT_STRING olarak belirtmeyin. Bunun nedeni, dosya gönderme sonunda bölümlendiğinde

çok baytlık bir karakterin bölünebilmedir. Böyle bir dosyayı aktarmak ve dönüştürmek için, biçimi MQFMT_STRING dışında bir biçim olarak belirtin; böylece, Başvuru İletisi çıkışı bu dosyayı dönüştürmez ve aktarma tamamlandığında dosyayı alma sonunda dönüştürmez.

Başvuru İletisi Çıkışı örneği derleniyor
Reference Message Exit örneğini derlemek için, IBM MQ ' in kurulu olduğu platforma ilişkin komutu kullanın.

MQ_INSTALLATION_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

amqsxrma 'yı derlemek için aşağıdaki komutları kullanın:

AçıkAIX

AIX

```
xlc_r -q64 -e MsgExit -bE:amqsxrm.exp -bM:SRE -o amqsxrm_64_r
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -lmqm_r amqsqrma.c
```

AçıkIBM i

IBM i

```
CRTCMOD MODULE(MYLIB/AMQSXRMA) SRCFILE(QMQMSAMP/QCSRC)
TERASPACE(*YES *TSIFC)
```

Not:

1. IFS dosya sistemini kullanacak şekilde modülünüzü yaratmak için SYSIFCOPT(*IFSIO) seçeneğini ekleyin.
2. Programı, iş parçacığı kullanmayan kanallarla kullanmak üzere yaratmak için aşağıdaki komutu kullanın: CRTPGM PGM(MYLIB/AMQSXRMA) BNDSRVPGM(QMQM/LIBMQM)
3. Programı iş parçacıklı kanallarla kullanmak üzere yaratmak için aşağıdaki komutu kullanın: CRTPGM PGM(MYLIB/AMQSXRMA) BNDSRVPGM(QMQM/LIBMQM_R)

AçıkLinux

Linux

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsxrma amqsqrma.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath= MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64
-lmqm_r
```

AçıkWindows

Windows IBM MQ artık mqm kitaplığını istemci paketlerinin yanı sıra sunucu paketlerini de sağlar; bu nedenle, aşağıdaki örnekte mqmvx.libyerine mqm.lib kullanılmaktadır:

```
cl amqsqrma.c /link /out:amqsxrm.dll /dll mqm.lib mqm.lib /def:amqsxrm.def
```

İlgili kavramlar

“Kanal çıkış programları yazılıyor” sayfa 923
Kanal çıkış programları yazmanıza yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

Get Reference Message örneğinin tasarımı (amqsgrma.c, AMQSGRM4)
Bu konuda, Get Reference Message örneğinin tasarımı açıklanmaktadır.

Program mantığı aşağıdaki gibidir:

1. Örnek tetiklenir ve kuyruk ve kuyruk yöneticisi adlarını giriş tetikleyicisi iletisinden alır.
2. Daha sonra MQCONN kullanarak belirtilen kuyruk yöneticisine bağlanır ve MQOPEN kullanarak belirtilen kuyruğu açar.
3. Örnek, kuyruktan ileti almak için bir döngü içinde 15 saniyelik bir bekleme aralığıyla MQGET ' i verir.
4. Bir ileti bir Başvuru İletisi ise, örnek, aktarılan dosyanın varlığını denetler.
5. Daha sonra kuyruğu kapatır ve kuyruk yöneticisiyle bağlantısını keser.

İstek örnek programları

İstek örnek programları, istemci/sunucu işlemesini gösterir. Örnekler, bir sunucu programı tarafından işlenen hedef sunucu kuyruğuna istek iletileri koyan istemcilerdir. Sunucu programının yanıt kuyruğuna yanıt iletisi koymasını bekler.

İstek örnekleri, MQPUT çağrısının kullanıldığı hedef sunucu kuyruğuna bir dizi istek iletisi yerleştirir. Bu iletiler, SYSTEM.SAMPLE.REPLY yanıtlayın. Programlar yanıt iletilerini bekler ve bunları görüntüler. Yanıtlar, yalnızca hedef sunucu kuyruğu bir sunucu uygulaması tarafından işleniyorsa ya da bu amaçla bir uygulama tetiklendiyse (sorgu, küme ve echo örnek programları tetiklenecek şekilde tasarlanmıştır) gönderilir. C örneği, ilk yanıtın gelmesi için 1 dakika (COBOL örneği 5 dakika bekler), sonraki yanıtlar için 15 saniye bekler, ancak her iki örnek de yanıt almadan sona erebilir. İstek örnek programlarının adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

İstek örnek programlarının çalıştırılması

amqsreq0.c, amqsreq ve amqsreqc örneklerinin çalıştırılması

Programın C sürümü üç parametre alır:

1. Hedef sunucu kuyruğunun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)
3. Yanıt kuyruğu (isteğe bağlı)

Örneğin, aşağıdakilerden birini girin:

- `amqsreq myqueue qmanagername replyqueue`
- `amqsreqc myqueue qmanagername`
- `amq0req0 myqueue`

Burada `myqueue` hedef sunucu kuyruğunun adı, `qmanagername` `myqueue`' un sahibi olan kuyruk yöneticisinin adı ve `replyqueue` yanıt kuyruğunun adıdır.

Kuyruk yöneticisinin adını atlarsanız, kuyruğun varsayılan kuyruk yöneticisinin sahibi olduğu varsayılır. Yanıt kuyruğunun adını atlarsanız, varsayılan yanıt kuyruğu sağlanır.

amq0req0.cbl örneğinin çalıştırılması

COBOL sürümünde parametre yok. Varsayılan kuyruk yöneticisine bağlanır ve çalıştırdığınızda size sorulur:

```
Please enter the name of the target server queue
```

Program girişini StdIn ' den alır ve her satırı hedef sunucu kuyruğuna ekleyerek her bir metin satırını bir istek iletisinin içeriği olarak alır. Boş bir satır okunduğunda program sona erer.

AMQSREQ4 örneğinin çalıştırılması

C programı, girişi sonlandıran boş bir süre ile stdin 'den (klavye) veri alarak iletiler oluşturur. Program en çok üç parametreyi alır: hedef kuyruğun adı (gerekli), kuyruk yöneticisi adı (isteğe bağlı) ve yanıt kuyruğu adı (isteğe bağlı). Kuyruk yöneticisi adı belirtilmezse, varsayılan kuyruk yöneticisi kullanılır. Yanıt kuyruğu belirtilmezse, SYSTEM.SAMPLE.REPLY kuyruğu kullanıldı.

Aşağıda, yanıt kuyruğunu belirten, ancak kuyruk yöneticisinin varsayılan olarak izin veren C örnek programının nasıl çağrılacağına bir örnek verilmiştir:

```
CALL PGM(QMQM/AMQSREQ4) PARM('SYSTEM.SAMPLE.LOCAL' '' 'SYSTEM.SAMPLE.REPLY')
```

Not: Kuyruk adlarının büyük ve küçük harfe duyarlı olduğunu unutmayın. AMQSAMP4 örnek dosya yaratma programı tarafından oluşturulan tüm kuyrukların adları büyük harfli karakterlerle oluşturulmuştur.

AMQ0REQ4 örneğinin çalıştırılması

COBOL programı, verileri klavyeden kabul ederek ileti yaratır. Programı başlatmak için programı çağırın ve hedef kuyruğunuzun adını parametre olarak belirleyin. Program, klavyeden arabelleğe giriş kabul eder ve her metin satırı için bir istek iletisi yaratır. Klavyede boş bir satır girdiğinizde program durur.

Tetikleme kullanılarak İstek örneğinin çalıştırılması

Örnek, tetikleme ve Sorma, Küme ya da Yankı örnek programlarından biriyle kullanılıyorsa, giriş satırı, tetiklenen programın erişmesini istediğiniz kuyruğun adı olmalıdır.

ALW *AIX, Linux, and Windows üzerinde tetikleme kullanılarak İstek örneğinin çalıştırılması*

AIX, Linux, and Windows'ta, bir oturumda tetikleyici izleme programı RUNMQTRM' yi başlatın ve daha sonra, amqsreq programını başka bir oturumda başlatın.

Örnekleri tetikleme kullanarak çalıştırmak için:

1. Tetikleyici izleme programı RUNMQTRM ' yi tek bir oturumda (başlatma kuyruğu SYSTEM.SAMPLE.TRIGGER seçeneğini kullanabilirsiniz).
2. amqsreq programını başka bir oturumda başlatın.
3. Hedef sunucu kuyruğu tanımladığınızı doğrulayın.

   İletileri koymak için istek örneği için hedef sunucu kuyruğu olarak kullanabileceğiniz örnek kuyruklar şunlardır:

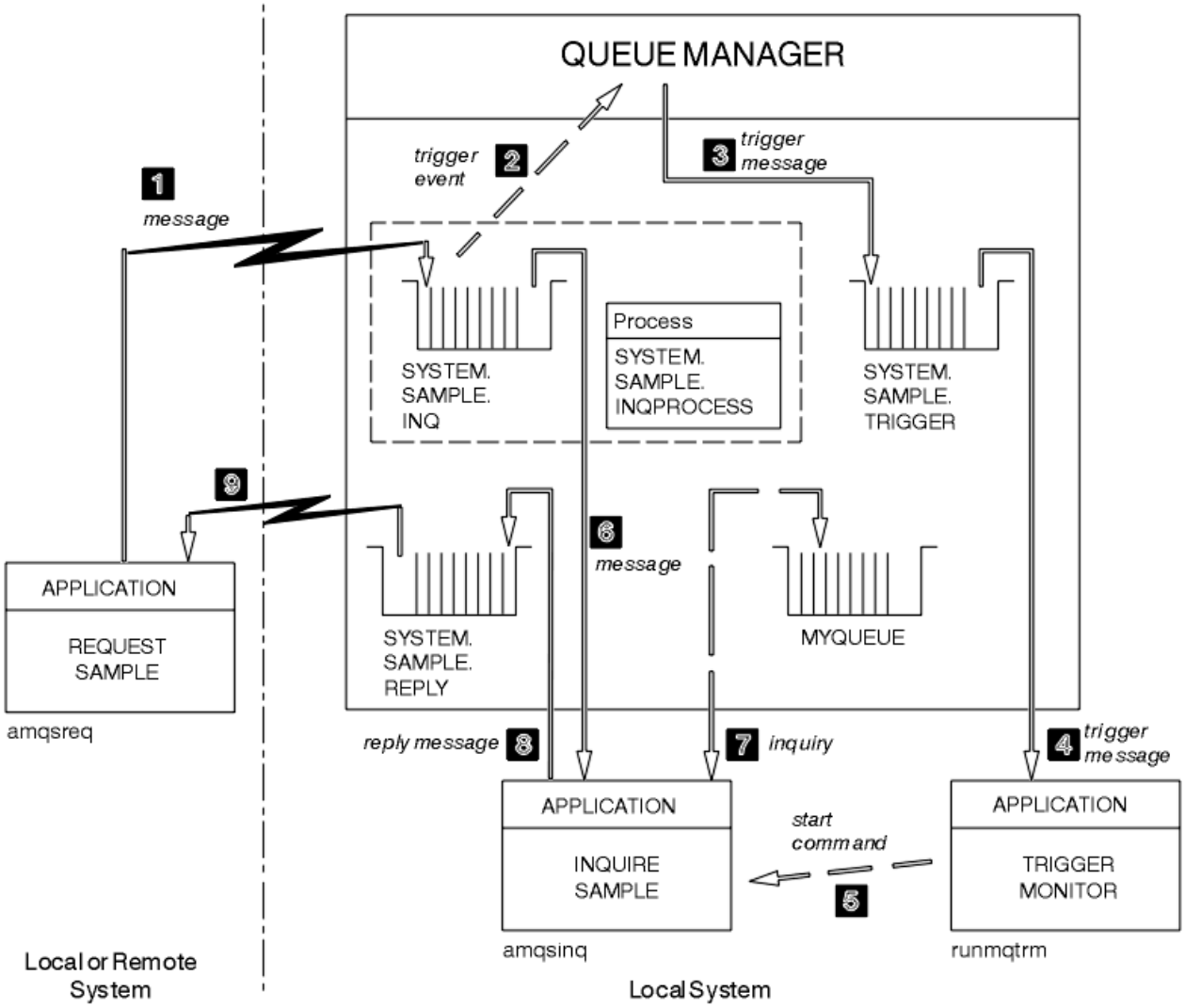
   - SYSTEM.SAMPLE.INQ -Inquire örnek programı için
   - SYSTEM.SAMPLE.SET -Set sample programı için
   - SYSTEM.SAMPLE.ECHO -Echo örnek programı için

   Bu kuyrukların tetikleyici tipi FIRST olduğu için, İstek örneğini çalıştırmadan önce kuyruklarda önceden ileti varsa, sunucu uygulamaları gönderdiğiniz iletiler tarafından tetiklenmez.
4. Inquire, Set ya da Echo örnek programının kullanması için bir kuyruk tanımladığınızdan emin olun.

Bu, istek örneği bir ileti gönderdiğinde tetikleyici izleyicisinin hazır olduğu anlamına gelir.

Not: RUNMQSC ve amqscos0.tst dosyası kullanılarak yaratılan örnek süreç tanımlamaları C örneklerini tetikler. amqscos0.tst içindeki süreç tanımlamalarını değiştirin ve COBOL sürümlerini kullanmak için bu güncellenmiş dosyayla RUNMQSC ' yi kullanın.

Şekil 132 sayfa 1068 içinde İstek ve Sorma örneklerinin birlikte nasıl kullanılacağı gösterilmektedir.



Şekil 132. Tetikleme kullanarak örnek iste ve sor

İstek örneği, Şekil 132 sayfa 1068 içinde iletileri hedef sunucu kuyruğuna yerleştirir, SYSTEM.SAMPLE.INQve Sorgu örneği, MYQUEUE kuyruğunu sorgular. Alternatif olarak, Inquire örneği için amqscos0.tstkomutunu ya da tanımladığınız başka bir kuyruğu çalıştırdığınızda tanımlanan örnek kuyruklardan birini kullanabilirsiniz.

Not: Şekil 132 sayfa 1068 içindeki sayılar olayların sırasını gösterir.

İstek ve Sorma örneklerini çalıştırmak için tetikleme yöntemini kullanarak:

1. Kullanmak istediğiniz kuyrukların tanımlı olup olmadığını denetleyin. Örnek kuyrukları tanımlamak ve bir kuyruk tanımlamak için amqscos0.tstkomutunu çalıştırın.
2. RUNMQTRM tetikleyici izleme komutunu çalıştırın:

```
RUNMQTRM -m qmanagername -q SYSTEM.SAMPLE.TRIGGER
```

3. İstek örneğini çalıştır

```
amqsreq SYSTEM.SAMPLE.INQ
```

Not: Süreç nesnesi, neyin tetikleneceğini tanımlar. İstemci ve sunucu aynı altyapıda çalışmıyorsa, tetikleyici izleme programı tarafından başlatılan her işlem *ApplType*değerini tanımlamalıdır; tersi

durumda, sunucu varsayılan tanımlamalarını (sunucu makinesiyle olağan olarak ilişkilendirilen uygulama tipi) alır ve bir hataya neden olur.

Uygulama tiplerinin listesi için bkz. ApplType.

4. Sorgu örneğinin kullanmasını istediğiniz kuyruğun adını girin:

```
MYQUEUE
```

5. İstek programını sona erdirmek için boş bir satır girin.
6. Daha sonra istek örneği, MYQUEUE ' dan alınan Inquire programının verilerini içeren bir ileti görüntüler.

Birden çok kuyruk kullanabilirsiniz; bu durumda “4” sayfa 1069. adımda diğer kuyrukların adlarını girin.

Tetikleme hakkında daha fazla bilgi için bkz. “Tetikleyiciler kullanılarak IBM MQ uygulamalarının başlatılması” sayfa 829.

IBM i *IBM i üzerinde tetikleme kullanılarak İstek örneğinin çalıştırılması*

IBM iişletim sistemlerinde, örnek tetikleyici sunucusunu (AMQSERV4) bir işte başlatın ve AMQSREQ4 ' ü başka bir işte başlatın. Bu, İstek örnek programı bir ileti gönderdiğinde tetikleyici sunucusunun hazır olduğu anlamına gelir.

Not:

1. AMQSAMP4 tarafından yaratılan örnek tanımlar, örneklerin C sürümlerini tetikler. COBOL sürümlerini tetiklemek istiyorsanız, SYSTEM.SAMPLE.ECHOPROCESS, SYSTEM.SAMPLE.INQPROCESSve SYSTEM.SAMPLE.SETPROCESS. CHGMQMPRC komutunu kullanabilirsiniz (ayrıntılar için bkz. Change MQ Process (CHGMQMPRC)) bunu yapmak için ya da kendi AMQSAMP4sürümünüzü düzenlemek ve çalıştırmak için.
2. AMQSERV4 kaynak kodu yalnızca C dili için sağlanır. Ancak, QMQM kitaplığında derlenmiş bir sürüm (COBOL örnekleriyle kullanabileceğiniz) sağlanır.

İstek iletilerinizi bu örnek sunucu kuyruklarına koyabilirdiniz:

- SYSTEM.SAMPLE.ECHO (Echo örnek programları için)
- SYSTEM.SAMPLE.INQ (Inquire örnek programları için)
- SYSTEM.SAMPLE.SET (Set örnek programları için)

SYSTEM.SAMPLE.ECHO programı Şekil 133 sayfa 1071içinde gösterilir. Örnek veri dosyası kullanıldığında, bu sunucuya C programı isteğini yayınlamak için aşağıdaki komutu kullanın:

```
CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(ECHO)')
```

Not: Bu örnek kuyruk FIRST tetikleyici tipine sahiptir; bu nedenle, İstek örneğini çalıştırmadan önce kuyrukta önceden ileti varsa, sunucu uygulamaları gönderdiğiniz iletiler tarafından tetiklenmez.

Başka örnekler denemek isterseniz, aşağıdaki çeşitlemeleri deneyebilirsiniz:

- AMQSTRG4 komutunu kullanın (ya da komut satırı eşdeğeri STRMQMTRM, ayrıntılar için Start MQ Trigger Monitor (STRMQMTRM) konusuna bakın). Bunun yerine işi göndermek için AMQSERV4 yerine, olası iş gönderimi gecikmeleri, olanların izlenmesini daha az kolaylaştıracaktır.
- SYSTEM.SAMPLE.INQUIRE ve SYSTEM.SAMPLE.SET örnek programları. Örnek veri dosyası kullanıldığında, bu sunuculara C programı isteklerini yayınlamak için aşağıdaki komutlar kullanılır:

```
CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(INQ)')
CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(SET)')
```

Bu örnek kuyrukların tetikleyici tipi de FIRST ' dir.

İstek örnek programının tasarımı
Program, iletileri koyabilmesi için hedef sunucu kuyruğunu açar. MQOO_OUTPUT seçeneğiyle MQOPEN çağrıyı kullanır. Kuyruğu açamazsa, program MQOPEN çağrısıyla döndürülen neden kodunu içeren bir hata iletisi görüntüler.

Program daha sonra SYSTEM.SAMPLE.REPLY , böylece yanıt iletileri alabilir. Bunun için, program MQOPEN çağrısıyla MQOO_INPUT_EXCLUSIVE seçeneğini kullanır. Kuyruğu açamazsa, program MQOPEN çağrısıyla döndürülen neden kodunu içeren bir hata iletisi görüntüler.

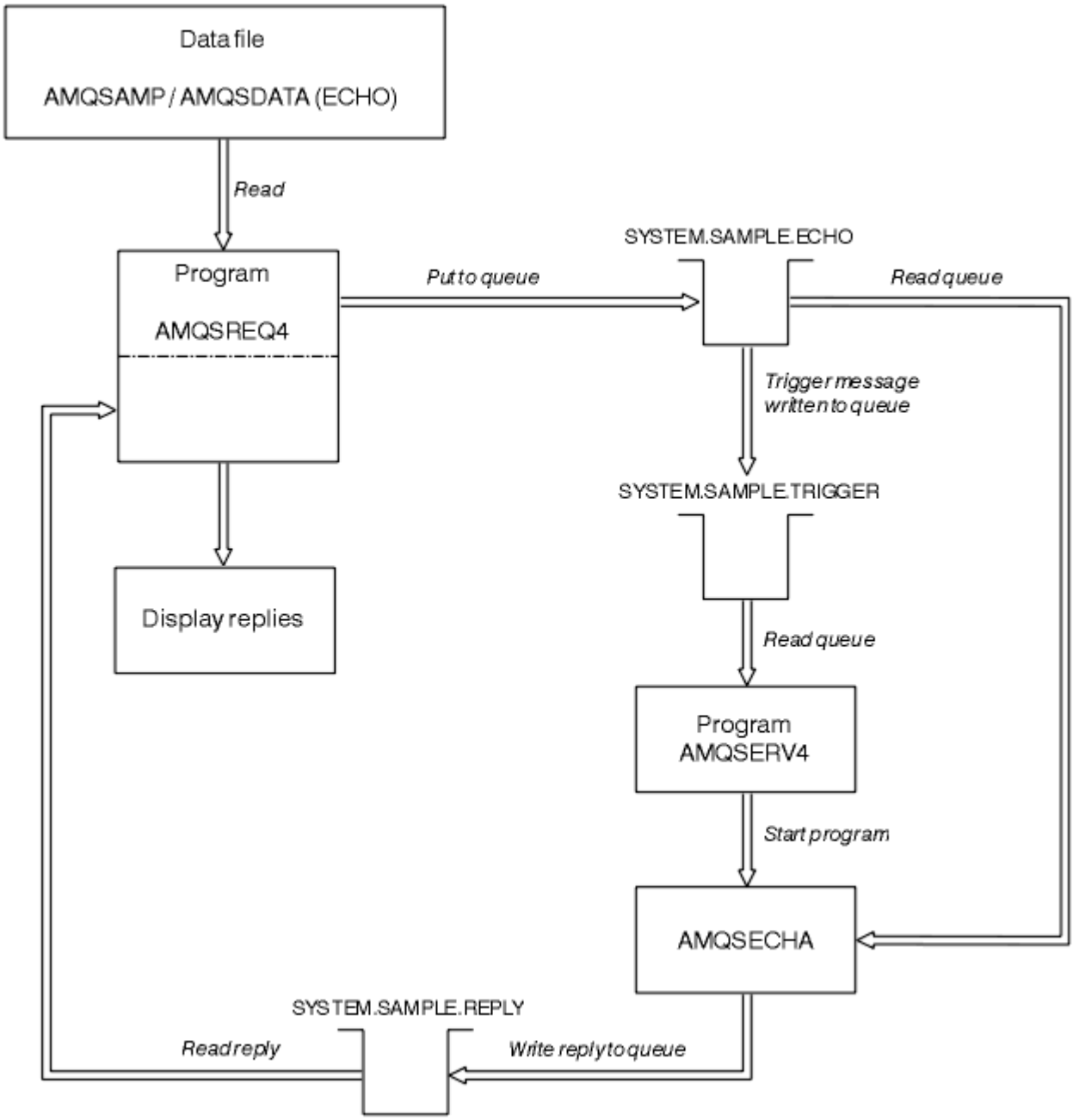
Her giriş satırı için, program metni bir arabelleğe okur ve o satırın metnini içeren bir istek iletisi yaratmak için MQPUT çağrıyı kullanır. Bu çağrıda program, istek iletisiyle ilgili olarak gönderilen rapor iletilerinin ileti verilerinin ilk 100 baytını içermesini istemek için MQRO_EXCEPTION_WITH_DATA rapor seçeneğini kullanır. Program, girişin sonuna ulaşıncaya ya da MQPUT çağrısı başarısız oluncaya kadar devam eder.

Daha sonra, program yanıt iletilerini kuyruktan kaldırmak için MQGET çağrısını kullanır ve yanıtlarda bulunan verileri görüntüler. MQGET çağrısı MQGMO_WAIT, MQGMO_CONVERT ve MQGMO_ACCEPT_TRUNCATED seçeneklerini kullanıyor. *WaitInterval* , COBOL sürümünde 5 dakika, C sürümünde 1 dakika, ilk yanıt (bir sunucu uygulamasının tetiklenmesine izin vermek için) ve sonraki yanıtlar için 15 saniyedir. Kuyrukta ileti yoksa, program bu dönemleri bekler. Bu aralığın süresi dolmadan hiçbir ileti gelmezse, çağrı başarısız olur ve MQRC_NO_MSG_AVAILABLE neden kodunu döndürür. Çağrı ayrıca MQGMO_ACCEPT_TRUNCATED_MSG seçeneğini de kullanır; böylece, bildirilen arabellek büyüklüğünden daha uzun iletiler kesilir.

Çağrı bu alanları aldığı iletide bulunan değerlere ayarladığından, bu program her MQGET çağrısından sonra MQMD yapısının *MsgId* ve *CorrelId* alanlarının nasıl temizleneceğini gösterir. Bu alanların temizlenmesi, ardışık MQGET çağrıları iletilerin kuyrukta tutulduğu sırada iletileri alır.

MQGET çağrısı MQRC_NO_MSG_AVAILABLE neden kodunu döndürünceye ya da MQGET çağrısı başarısız oluncaya kadar program devam eder. Çağrı başarısız olursa, program neden kodunu içeren bir hata iletisi görüntüler.

Daha sonra program, MQCLOSE çağrıyı kullanarak hem hedef sunucu kuyruğunu hem de yanıt kuyruğunu kapatır.



Şekil 133. Örnek IBM i Client/Server (Echo) programı akış şeması

Set örnek programları

Örnek programları ayarla seçeneği, kuyruğun **InhibitPut** özniteliğini değiştirmek için MQSET çağrısıyla bir kuyruğa koyma işlemlerini engeller. Ayrıca, Set örnek programlarının tasarımı hakkında bilgi edinin.

Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Programların tetiklenen programlar olarak çalışması amaçlanmıştır; bu nedenle, bu programların tek girişi, istenen öznitelikleri olan bir hedef kuyruğun adını içeren bir MQTMC2 (tetikleyici iletisi) yapısıdır. C sürümü kuyruk yöneticisi adını da kullanır. COBOL sürümü, varsayılan kuyruk yöneticisini kullanır.

Tetikleme işleminin çalışması için, kullanmak istediğiniz Set örnek programının SYSTEM.SAMPLE.SETkuyruğuna gelen iletiler tarafından tetiklendiğinden emin olun. Bunu yapmak için, SYSTEM.SAMPLE.SETPROCESSsüreç tanımlamasının *ApplicId* alanında kullanmak istediğiniz

Set sample programının adını belirtin. Örnek kuyruk, FIRST tetikleyici tipine sahiptir; İstek örneğini çalıştırmadan önce kuyrukta önceden ileti varsa, Gönderdiğiniz iletiler Set örneğini tetiklemez.

Tanımlamayı doğru olarak ayarladığınızda:

- ALW AIX, Linux, and Windows sistemleri için, bir oturumda **runmqtrm** programını başlatın ve ardından amqsreq programını başka bir oturumda başlatın.
- IBM i IBM iiçin, AMQSERV4 programını bir oturumda başlatın ve AMQSREQ4 programını başka bir oturumda başlatın. AMQSERV4yerine AMQSTRG4 komutunu kullanabilirsiniz, ancak olası iş gönderimi gecikmeleri, olanların izlenmesini daha az kolaylaştıracaktır.

Her biri yalnızca bir kuyruk adı içeren istek iletilerini SYSTEM.SAMPLE.SETkuyruğuna göndermek için İstek örnek programlarını kullanın. Her istek iletisi için, Set örnek programları, belirlenen kuyrukta koyma işlemlerinin engellendiğini belirten bir doğrulama içeren bir yanıt iletisi gönderir. Yanıtlar, istek iletisinde belirlenen yanıt kuyruğuna gönderilir.

Set örnek programının tasarımı

Program, başlatılırken iletildiği tetikleyici ileti yapısında adı belirtilen kuyruğu açar. (Daha anlaşılır olması için, buna *istek kuyruğu*diyoruz.) Program, paylaşılan giriş için bu kuyruğu açmak üzere MQOPEN çağrılmasını kullanır.

Program, iletileri bu kuyruktan kaldırmak için MQGET çağrısını kullanır. Bu çağrı, 5 saniyelik bekleme aralığıyla MQGMO_ACCEPT_TRUNCATED_MSG ve MQGMO_WAIT seçeneklerini kullanır. Program, bir istek iletisi olup olmadığını görmek için her iletinin tanımlayıcısını sınar; değilse, program iletiyi atar ve bir uyarı iletisi görüntüler.

İstek kuyruğundan kaldırılan her istek iletisi için, program kuyruğun adını okur (*hedef kuyruk* olarak adlandırılacaktır). verilerde yer alır ve MQOO_SET seçeneğiyle MQOPEN çağrısıyla o kuyruğu açar. Daha sonra program, hedef kuyruğun **InhibitPut** özniteliğinin değerini MQQA_PUT_INENGELLENDI olarak ayarlamak için MQSET çağrısına kullanır.

MQSET çağrısı başarılı olursa, program yanıt kuyruğuna bir yanıt iletisi koymak için MQPUT1 çağrıyı kullanır. Bu ileti PUT `inhibited`dizgisini içerir.

MQOPEN ya da MQSET çağrısı başarısız olursa, program yanıt kuyruğuna bir `report` iletisi koymak için MQPUT1 çağrıyı kullanır. Bu rapor iletisinin ileti tanımlayıcısının *Feedback* alanında, hangisinin başarısız olduğuna bağlı olarak MQOPEN ya da MQSET çağrısının döndürdüğü neden kodu bulunur.

MQSET çağrısından sonra, program MQCLOSE çağrıyı kullanarak hedef kuyruğu kapatır.

İstek kuyruğunda kalan ileti yoksa, program o kuyruğu kapatır ve kuyruk yöneticisiyle olan bağlantısını keser.

TLS örnek programı

AMQSSSLC, MQCONNX çağrısında TLS istemci bağlantısı bilgilerini sağlamak için MQCNO ve MQSCO yapılarının nasıl kullanılacağını gösteren örnek bir C programıdır. Bu, bir istemci MQI uygulamasının, istemci kanal tanımlama çizelgesi (CCDT) olmadan çalıştırma zamanında istemci bağlantı kanalının ve TLS ayarlarının tanımlamasını sağlamasına olanak sağlar.

Bir bağlantı adı sağlanırsa, program MQCD yapısında bir istemci bağlantı kanalı tanımlaması oluşturur.

Anahtar havuzu dosyasının kök adı sağlanırsa, program bir MQSCO yapısı oluşturur; bir OCSP yanıtlayıcısı URL de sağlanırsa, program bir kimlik doğrulama bilgileri kaydı MQAIR yapısı oluşturur.

Program daha sonra MQCONNX kullanarak kuyruk yöneticisine bağlanır. Bağlandığı kuyruk yöneticisinin adını sorar ve yazdırır.

Bu programın MQI istemci uygulaması olarak bağlanması amaçlanmıştır. Ancak, olağan bir MQI uygulaması olarak bağlanabilir; bu durumda, yalnızca yerel bir kuyruk yöneticisine bağlanır ve istemci bağlantısı bilgilerini yoksayar.

V 9.3.0 V 9.3.0 Anahtar havuzuna erişmek için kullanılan parola bir dosyaya saklanmamışsa, uygulama çalıştığında **amqssslc** ' e geçiş tümceciğini sağlamanız gerekir. Geçiş tümceciğini aşağıdaki yollarla da sağlayabilirsiniz:

- Geçiş tümceciği için **amqssslc** bilgi isteminde bulunma ya da
- *MQKEYRPWD* ortam değişkeninin kullanılması ya da
- İstemci yapılandırma dosyasında **SSLKeyRepositoryPassword** özniteliğinin kullanılması

IBM MQ MQI client uygulamalarına anahtar havuzu parolası sağlanmasıyla ilgili daha fazla bilgi için bkz. AIX, Linux, and Windowsüzerinde IBM MQ MQI client için anahtar havuzu parolası sağlanması.

amqssslc , tümü isteğe bağlı olan aşağıdaki parametreleri kabul eder:

-m QmgrName
Bağlanılacak kuyruk yöneticisinin adı

-c ChannelName
Kullanılacak kanalın adı

-x ConnName
Sunucu bağlantısı adı

TLS parametreleri:

V 9.3.0 V 9.3.0 **-k KeyReposFileName**
Anahtar havuzu dosyasının adı. Dosya uzantısı sağlanmazsa, dosya uzantısının `.kdb`olduğu varsayılır. Örneğin:

```
/home/user/client.kdb
C:\User\client.p12
```

-s CipherSpec
Kuyruk yöneticisindeki SVRCONN kanal tanımlamasında **SSLCIPH** ile ilgili TLS kanal CipherSpec dizgisi.

-f
Yalnızca FIPS 140-2 sertifikalı algoritmaların kullanılması gerektiğini belirtir.

-b VALUE1[,VALUE2...]
Yalnızca Suite B uyumlu algoritmaların kullanılması gerektiğini belirtir. Bu değiştirge, şu değerlerden birinin ya da daha fazlasının virgülle ayrılmış listesidir: NONE,128_BIT,192_BIT. Bu değerler, **MQSUITEB** ortam değişkenine ilişkin değerlerle ve istemci yapılanış kütüğü SSL kısmına ilişkin eşdeğer **EncryptionPolicySuiteB** ayarıyla aynıdır.

-p İlkesi
Kullanılacak sertifika doğrulama ilkesini belirtir. Bu, aşağıdaki değerlerden biri olabilir:

Fark Etmez
Güvenli yuva kitaplığı tarafından desteklenen sertifika doğrulama ilkelerinin her birini uygulayın ve ilkelerden herhangi biri sertifika zincirini geçerli kabul ediyorsa sertifika zincirini kabul edin. Bu ayar, modern sertifika standartlarına uymayan eski dijital sertifikalarla en üst düzeyde geriye dönük uyumluluk için kullanılabilir.

RFC5280
Yalnızca RFC 5280 uyumlu sertifika doğrulama ilkesini uygulayın. Bu ayar, HERHANGİ BİRİ ayarından daha sıkı doğrulama sağlar, ancak bazı eski dijital sertifikaları reddeder.

Varsayılan değer ANY 'dir.

-l CertLabel
Güvenli bağlantı için kullanılacak sertifika etiketi.

Not: Değeri küçük harfli karakterler kullanarak belirtmelisiniz.

V 9.3.0 V 9.3.0 **-w**

amqssslc ' in sağlanacak anahtar havuzu parolası için bilgi isteminde bulunacağını belirtir.

V 9.3.0 V 9.3.0 **-i**

Sağlanacak anahtar havuzu parolasını şifrelemek için kullanılan ilk anahtar için **amqssslc** bilgi isteminde bulunacağını belirtir.

Anahtar havuzu parolası **runmqicred** yardımcı programı kullanılarak şifrelendiğinde ilk anahtar dosyası belirtildiyse bu seçeneği belirleyin.

OCSP sertifikası iptal parametresi:

-o URL

OCSP Yanıtlayıcısı URL

Kuyruk yöneticisiyle kimlik doğrulaması yapmak için kullanılan kimlik bilgilerini sağlamak üzere aşağıdaki ortam değişkenlerinden birini de ayarlayabilirsiniz:

MQSAMP_USER_ID

Kuyruk yöneticisiyle kimlik doğrulaması yapmak için bir kullanıcı kimliği ve parola kullanmak istiyorsanız, bağlantı kimlik doğrulaması için kullanılacak kullanıcı kimliğini ayarlayın. Program, kullanıcı kimliğiyle birlikte parolanın da girmesini ister.

Linux AIX V 9.3.4 **MQSAMP_TOKEN**

Kuyruk yöneticisiyle kimlik doğrulaması için bir kimlik doğrulama belirteci sağlamak istiyorsanız, boş olmayan bir değere ayarlayın. Program, kimlik doğrulama belirtecini ister.

TLS örnek programının çalıştırılması

TLS örnek programını çalıştırmak için öncelikle TLS ortamınızı ayarlamanız gerekir. Daha sonra, komut satırından bir dizi parametre sağlayarak örneği çalıştırabilirsiniz.

Bu görev hakkında

Aşağıdaki yönergeler, örnek programı kişisel sertifikaları kullanarak çalıştırıyor. Komutu değiştirerek, örneğin CA sertifikalarını kullanabilir ve OCSP yanıtlayıcılarını kullanarak durumlarını denetleyebilirsiniz. Örnek içindeki yönergelere bakın.

Yordam

1. QM1adıyla bir kuyruk yöneticisi yaratın. Daha fazla bilgi için bkz. crtmqm.
2. Kuyruk yöneticisi için bir anahtar havuzu yaratın. Daha fazla bilgi için bkz. AIX, Linux, and Windows' da anahtar havuzu ayarlama.
3. İstemci için bir anahtar havuzu yaratın. *clientkey.kdb*adını verin.

   Anahtar havuzunu yaratırken anahtar havuzu parolasını bir dosyada saklayabilirsiniz.
4. Kuyruk yöneticisi için bir kişisel sertifika yaratın. Daha fazla bilgi için bkz. AIX, Linux, and Windowsüzerinde kendinden imzalı kişisel sertifika oluşturma.
5. İstemci için kişisel sertifika yaratın.
6. Kişisel sertifikayı sunucu anahtarı havuzundan çıkarın ve istemci havuzuna ekleyin. Daha fazla bilgi için bkz. AIX, Linux, and Windows üzerindeki bir anahtar havuzundan kendinden onaylı sertifikanın genel kısmının çıkarılmasıve AIX, Linux, and Windows sistemlerinde bir anahtar havuzuna CA sertifikası (ya da kendinden onaylı bir sertifikanın genel kısmı) eklenmesi.
7. Kişisel sertifikayı istemci anahtar havuzundan çıkarın ve sunucu anahtarı havuzuna ekleyin.
8. MQSC komutunu kullanarak bir sunucu bağlantı kanalı yaratın:

   ```
   DEFINE CHANNEL(QM1SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
   SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
   ```

   Daha fazla bilgi için bkz. Sunucu-bağlantı kanalı

9. Kuyruk yöneticisinde bir kanal dinleyicisi tanımlayın ve başlatın. Daha fazla bilgi için bkz. DEFINE LISTENER ve START LISTENER.
10. Aşağıdaki komutu kullanarak örnek programı çalıştırın:

V 9.3.0 V 9.3.0

```
AMQSSSLC -m QM1 -c QM1SVRCONN -x localhost
-k "C:\Program Files\IBM\MQ\clientkey.kdb" -s TLS_RSA_WITH_AES_128_CBC_SHA256
-o http://dummy.OCSP.responder
```

Sonuçlar

Örnek program aşağıdaki işlemleri gerçekleştirir:

1. Belirtilen herhangi bir kuyruk yöneticisine ya da varsayılan kuyruk yöneticisine, belirtilen seçenekleri kullanarak bağlanır.
2. Kuyruk yöneticisini açar ve adını sorar.
3. Kuyruk yöneticisini kapatır.
4. Kuyruk yöneticisiyle bağlantıyı keser.

Örnek program başarılı bir şekilde çalışırsa, aşağıdaki örneğe benzer bir çıkış görüntüler:

```
Sample AMQSSSLC start
Connecting to queue manager QM1
Using the server connection channel QM1SVRCONN
on connection name localhost.
Using TLS CipherSpec TLS_RSA_WITH_AES_128_CBC_SHA256
Using TLS key repository stem C:\Program Files\IBM\MQ\clientkey
Using OCSP responder URL http://dummy.OCSP.responder
Connection established to queue manager QM1

Sample AMQSSSLC end
```

Örnek program bir sorunla karşılaşırsa, uygun bir hata iletisi görüntüler; örneğin, geçersiz bir OCSP yanıtlayıcısı URLbelirtirseniz aşağıdaki iletiyi alırsınız:

```
MQCONNX ended with reason code 2553
```

Neden kodlarının bir listesi için API tamamlama ve neden kodlarıbaşlıklı konuya bakın.

Triggering örnek programları

Tetikleme örneğinde sağlanan işlev, **runmqtrm** programındaki tetikleyici izleyicisinde sağlanan bir altkümedir.

Bu programların adları için bkz. “Multiplatforms üzerinde örnek programlarda gösterilen özellikler” sayfa 1015 .

Tetikleyici örneğinin tasarımı

Tetikleyici örnek program, MQOPEN çağrısıyla MQOO_INPUT_AS_Q_DEF seçeneğini kullanarak başlatma kuyruğunu açar. Sınırsız bekleme aralığı belirterek, MQGMO_ACCEPT_TRUNCATED_MSG ve MQGMO_WAIT seçenekleriyle MQGET çağrısıyla başlatma kuyruğundan iletileri alır. Program, sırasıyla ileti almak için her MQGET çağrısının öncesinde *MsgId* ve *CorrelId* alanlarını temizler.

Başlatma kuyruğundan bir ileti aldığında, program iletinin boyutunun bir MQTM yapısıyla aynı olduğundan emin olmak için iletinin boyutunu denetleyerek iletiyi sınar. Bu sınama başarısız olursa, program bir uyarı görüntüler.

Geçerli tetikleyici iletiler için, tetikleyici örnek şu alanlardaki verileri kopyalar: *ApplicId*, *EnvrData*, *Version*ve *ApplType*. Bu alanların son ikisi sayısaldır, bu nedenle program IBM i, AIX, Linux, and Windows sistemleri için MQTMC2 yapısında kullanılacak karakter değişimleri yaratır.

Tetikleyici örnek, tetikleyici iletisinin *ApplicId* alanında belirtilen uygulamaya bir başlatma komutu verir ve MQTMC2 ya da MQTMC (tetikleyici iletisinin karakter sürümü) yapısını geçirir.

- ALW AIX, Linux, and Windows sistemlerinde *EnvData* alanı, çağıran komut dizgisinin uzantısı olarak kullanılır.
- IBM i IBM i' da iş sunma parametreleri olarak kullanılır; örneğin, iş önceliği ya da iş tanımlaması.

Son olarak, program başlatma kuyruğunu kapatır.

IBM i üzerinde tetikleyici örnek programların sona erdirilmesi

IBM i

Tetikleyici izleme programı, sistem isteği seçeneği 2 (ENDRQS) ya da tetikleyici kuyruğundan alma işlemini engelleyerek sona erdirilebilir.

Örnek tetikleyici kuyruğu kullanılırsa, komut aşağıdaki gibidir:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') MQMNAME GETENBL(*NO)
```

Önemli: Bu kuyrukta yeniden tetiklemeye başlamadan önce aşağıdaki komutu girmeniz gerekir:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*YES)
```

Triggering örnek programlarının çalıştırılması
Bu konuda, Triggering örnek programlarının çalıştırılmasıyla ilgili bilgiler yer alır.

amqstrg0.c, amqstrg ve amqstrgc örneklerinin çalıştırılması

Program 2 parametre alır:

1. Başlatma kuyruğunun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)

Bir kuyruk yöneticisi belirtilmezse, varsayılan yöneticiye bağlanır. amqscos0.tst; komutunu çalıştırdığınızda örnek bir başlatma kuyruğu tanımlanacaktır; bu kuyruğun adı SYSTEM.SAMPLE.TRIGGERve bu programı çalıştırırken kullanabilirsiniz.

Not: Bu örnekteki işlev, runmqtrm programında sağlanan tam tetikleme işlevinin bir altkümesidir.

AMQSTRG4 örneğinin çalıştırılması

IBM i

Bu, IBM i ortamı için bir tetikleyici izleme programıdır. Başlatılacak her uygulama için bir IBM i işi sunar. Bu, her tetikleyici iletisiyle ilişkili ek işleme olduğu anlamına gelir.

AMQSTRG4 (QCSRC içinde) iki parametre alır: hizmet vereceği başlatma kuyruğunun adı ve kuyruk yöneticisinin adı (isteğe bağlı). AMQSAMP4 (QCLSRC içinde), örnek bir başlatma kuyruğunu (SYSTEM.SAMPLE.TRIGGERtetikleyicisi, örnek programları denerken kullanabileceğiniz bir tetikleyicidir.

Örnek tetikleyici kuyruğu kullanıldığında, verilecek komut şudur:

```
CALL PGM(QMQM/AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER')
```

Diğer bir seçenek olarak, CL eşdeğeri STRMQMTRM ' yi kullanabilirsiniz; ayrıntılar için Start MQ Trigger Monitor (STRMQMTRM)başlıklı konuya bakın.

AMQSERV4 örneğinin çalıştırılması

IBM i

Bu, IBM i ortamı için bir tetikleyici sunucusudur. Bu sunucu, her tetikleme iletisi için, belirlenen uygulamayı başlatmak üzere kendi işinde başlatma komutunu çalıştırır. Tetikleyici sunucu CICS hareketlerini çağırabilir.

AMQSERV4 iki parametreyi alır: hizmet vereceği başlatma kuyruğunun adı ve kuyruk yöneticisinin adı (isteğe bağlı). AMQSAMP4 , örnek bir başlatma kuyruğunu (SYSTEM.SAMPLE.TRIGGERtetikleyicisi, örnek programları denerken kullanabileceğiniz bir tetikleyicidir.

Örnek tetikleyici kuyruğunun kullanılması, verilecek komutun aşağıdaki gibidir:

```
CALL PGM(QMQM/AMQSERV4) PARM('SYSTEM.SAMPLE.TRIGGER')
```

Tetikleyici sunucusunun tasarımı
Tetikleyici sunucusunun tasarımı, birkaç kural dışı durum dışında, tetikleyici izleme programının tasarımına benzer.

Tetikleyici sunucusunun tasarımı, tetikleyici sunucusu dışında, tetikleyici izleme programının tasarımına benzer:

- MQAT_CICS ve MQAT_OS400 uygulamalarına izin verir.
- IBM i Bir IBM i işi göndermek yerine, IBM i uygulamalarını kendi işiyle çağırır (ya da CICS uygulamalarını başlatmak için STRCICSUSR komutunu kullanır).
- CICS uygulamalarında, örneğin, STRCICSUSR komutundaki tetikleme iletisinden CICS bölgesini belirlemek için *EnvData*yerine kullanılır.
- Paylaşılan giriş için başlatma kuyruğunu açar; böylece birçok tetikleyici sunucusu aynı anda çalışabilir.

Not: AMQSERV4 tarafından başlatılan programlar, tetikleyici sunucusunu durdurduğundan MQDISC çağrılarını kullanmamalıdır. AMQSERV4 tarafından başlatılan programlar MQCONN çağrıyı kullanıyorsa, MQRC_ALREADY_CONNECTED neden kodunu alır.

ALW *AIX, Linux, and Windows üzerinde TUXEDO örneklerinin kullanılması*

TUXEDO için Put ve Get örnek programları hakkında bilgi edinin ve TUXEDO ' da sunucu ortamı oluşturma.

Başlamadan önce

Bu örnekleri çalıştırmadan önce sunucu ortamını oluşturmanız gerekir.

Bu görev hakkında

Not: Bu bölüm boyunca, uzun komutları birden çok satıra bölmek için ters eğik çizgi (\) karakteri kullanılır. Bu karakteri girmeyin. Her komutu tek bir satır olarak girin.

ALW *Sunucu ortamı oluşturuluyor*
Farklı platformlar için IBM MQ için sunucu ortamı oluşturulmasına ilişkin bilgiler.

Başlamadan önce

Çalışan bir TUXEDO ortamınız olduğu varsayılır.

AIX *AIX (32 bit) için sunucu ortamı oluşturuluyor*
IBM MQ for AIX (32 bit) için sunucu ortamı oluşturma.

Yordam

1. Sunucu ortamının oluşturulduğu bir dizin (örneğin, APPDIR) oluşturun ve bu dizindeki tüm komutları yürütün.
2. Aşağıdaki ortam değişkenlerini dışa aktarın; burada TUXDIR , TUXEDO 'un kök dizinidir ve *MQ_INSTALLATION_PATH* , IBM MQ ' un kurulu olduğu üst düzey dizini gösterir:

```
$ 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. udataobj/RMTUXEDO dosyasına aşağıdaki satırı ekleyin:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa -lmqm
```

4. Komutları çalıştırın:

```
$ 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. ubbstxcx.cfg dosyasını düzenleyin ve makine adının, çalışma dizinlerinin ve kuyruk yöneticisinin ayrıntılarını gerektiği şekilde ekleyin:

```
$ tmloadcf  -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. TLOGDEVICE yaratın:

```
$tmadmin -c
```

   Bir bilgi istemi görüntülenir. Bu komut isteminde şunu girin:

```
> crdl -z /APPDIR/TLOG1
```

7. Kuyruk yöneticisini başlat:

```
$ strmqm
```

8. Smokin 'i Başlat:

```
$ tmboot -y
```

Sonraki adım

Artık iletileri kuyruğa koymak ve kuyruktan almak için doputs ve doalmalarını kullanabilirsiniz.

AIX *AIX (64 bit) için sunucu ortamı oluşturuluyor*

IBM MQ for AIX (64 bit) için sunucu ortamı oluşturma.

Yordam

1. Sunucu ortamının oluşturulduğu bir dizin (örneğin, APPDIR) oluşturun ve bu dizindeki tüm komutları yürütün.
2. Aşağıdaki ortam değişkenlerini dışa aktarın; burada TUXDIR TUXEDO 'un kök dizinini, *MQ_INSTALLATION_PATH* ise IBM MQ ' un 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. udataobj/RMTUXEDO dosyasına aşağıdaki satırı ekleyin:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmxa64 -lmqm
```

4. Komutları çalıştırın:

```
$ 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. ubbstxcx.cfg dosyasını düzenleyin ve makine adının, çalışma dizinlerinin ve kuyruk yöneticisinin ayrıntılarını gerektiği şekilde ekleyin:

```
$ tmloadcf  -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. TLOGDEVICE yaratın:

```
$tmadmin -c
```

   Bir bilgi istemi görüntülenir. Bu komut isteminde şunu girin:

```
> crdl -z /APPDIR/TLOG1
```

7. Kuyruk yöneticisini başlat:

```
$ strmqm
```

8. Smokin 'i Başlat:

```
$ tmboot -y
```

Sonraki adım

Artık iletileri kuyruğa koymak ve kuyruktan almak için doputs ve doalmalarını kullanabilirsiniz.

Windows *Windows (32 bit) için sunucu ortamı oluşturuluyor*
IBM MQ for Windows (32 bit) için sunucu ortamı oluşturuluyor.

Bu görev hakkında

Not: Aşağıda *VARIABLE* olarak tanımlanan alanları dizin yollarıyla değiştirin:

Çizelge 164. Dizin yollarına değiştirilecek alanlar

Alan	Dizin yolu
MQMDIR	IBM MQ kurulduğunda belirtilen dizin yolu; örneğin, `g:\Program Files\IBM\MQ`.
TUXDIR	TUXEDO kurulduğunda belirtilen dizin yolu; örneğin, `f:\tuxedo`.
APPDIR	Örnek uygulama için kullanılacak dizin yolu; örneğin, `f:\tuxedo\apps\mqapp`.

```
*RESOURCES
IPCKEY       99999
UID          0
GID          0
MAXACCESSERS 20
MAXSERVERS   20
MAXSERVICES  50
MASTER       SITE1
MODEL        SHM
LDBAL        N

*MACHINES
MachineName LMID=SITE1
            TUXDIR="f:\tuxedo"
            APPDIR="f:\tuxedo\apps\mqapp;g:\Program Files\IBM\WebSphere MQ\bin"
            ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
            TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
            ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
            TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
            TLOGNAME=TLOG
            TYPE="i386NT"
            UID=0
            GID=0

*GROUPS
GROUP1
        LMID=SITE1 GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSERIES_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1   SRVGRP=GROUP1 SRVID=1
MQSERV2   SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Şekil 134. IBM MQ for Windows için ubbstxcn.cfg dosyası örneği

Not: *MachineName*makine adını ve dizin yollarını kuruluşunuzla eşleşecek şekilde değiştirin. Ayrıca, *MYQUEUEMANAGER* kuyruk yöneticisi adını, bağlanmak istediğiniz kuyruk yöneticisinin adıyla değiştirin.

IBM MQ for Windows için örnek ubbconfig dosyası Şekil 134 sayfa 1080içinde listelenmiştir. IBM MQ örnekleri dizininde ubbstxcn.cfg olarak sağlanır.

IBM MQ for Windows için sağlanan örnek makefile (bkz. Şekil 135 sayfa 1081) ubbstxmn.makolarak adlandırılır ve IBM MQ samples dizininde tutulur.

```
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
```

Şekil 135. IBM MQ for Windows için örnek TUXEDO makefile

Sunucu ortamını ve örneklerini oluşturmak için aşağıdaki adımları tamamlayın.

Yordam

1. Örnek uygulamanın oluşturulacağı bir uygulama dizini yaratın; örneğin:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Aşağıdaki örnek dosyaları IBM MQ örnek dizininden uygulama dizinine kopyalayın:
   - amqstxmn.mak
   - amqstxen.env
   - ubbstxcn.cfg
3. Kuruluşunuzda kullanılan dizin adlarını ve dizin yollarını ayarlamak için bu dosyaların her birini düzenleyin.
4. Bağlanmak istediğiniz makine adının ve kuyruk yöneticisinin ayrıntılarını eklemek için ubbstxcn.cfg dosyasını düzenleyin (bkz. Şekil 134 sayfa 1080).
5. *TUXDIR*udataobj\rmTUXEDO dosyasına aşağıdaki satırı ekleyin:

   ```
   MQSERIES_XA_RMI;MQRMIXASwitchDynamic;MQMDIR\tools\lib\mqmxa.lib MQMDIR\tools\lib\mqm.lib
   ```

   Yeni giriş, dosyada bir satır olmalıdır.
6. Aşağıdaki ortam değişkenlerini ayarlayın:

   ```
   TUXDIR=TUXDIR
   TUXCONFIG=APPDIR\tuxconfig
   FIELDTBLS=MQMDIR\tools\c\samples\amqstxvx.fld
   LANG=C
   ```

7. TUXEDO için bir TLOG aygıtı oluşturun.

Bunu yapmak için `tmadmin -c`komutunu çağırın ve şu komutu girin:

```
crdl -z APPDIR\TLOG
```

8. Yürürlükteki dizini *APPDIR*olarak ayarlayın ve örnek makefile `amqstxmn.mak` dosyasını dış proje makefile olarak çağırın. Örneğin, Microsoft Visual C++ ile aşağıdaki komutu verin:

```
msvc amqstxmn.mak
```

Tüm örnek programları oluşturmak için **build** (oluştur) seçeneğini belirleyin.

Windows *Windows (64 bit) için sunucu ortamı oluşturuluyor*

IBM MQ for Windows (64 bit) için sunucu ortamı oluşturma.

Bu görev hakkında

Not: Aşağıda *VARIABLE* olarak tanımlanan alanları dizin yollarıyla değiştirin:

Çizelge 165. Dizin yollarına değiştirilecek alanlar

Alan	**Dizin yolu**
MQMDIR	IBM MQ kurulduğunda belirtilen dizin yolu; örneğin, `g:\Program Files\IBM\MQ`.
TUXDIR	TUXEDO kurulduğunda belirtilen dizin yolu; örneğin, `f:\tuxedo`.
APPDIR	Örnek uygulama için kullanılacak dizin yolu; örneğin, `f:\tuxedo\apps\mqapp`.

```
*RESOURCES
IPCKEY       99999
UID          0
GID          0
MAXACCESSERS 20
MAXSERVERS   20
MAXSERVICES  50
MASTER       SITE1
MODEL        SHM
LDBAL        N

*MACHINES
MachineName LMID=SITE1
            TUXDIR="f:\tuxedo"
            APPDIR="f:\tuxedo\apps\mqapp;g:\Programi¿½Files\IBM\WebSphere MQ\bin"
            ENVFILE="f:\tuxedo\apps\mqapp\amqstxen.env"
            TUXCONFIG="f:\tuxedo\apps\mqapp\tuxconfig"
            ULOGPFX="f:\tuxedo\apps\mqapp\ULOG"
            TLOGDEVICE="f:\tuxedo\apps\mqapp\TLOG"
            TLOGNAME=TLOG
            TYPE="i386NT"
            UID=0
            GID=0

*GROUPS
GROUP1
        LMID=SITE1 GRPNO=1
        TMSNAME=MQXA
        OPENINFO="MQSERIES_XA_RMI:MYQUEUEMANAGER"

*SERVERS
DEFAULT: CLOPT="-A -- -m MYQUEUEMANAGER"

MQSERV1   SRVGRP=GROUP1 SRVID=1
MQSERV2   SRVGRP=GROUP1 SRVID=2

*SERVICES
MPUT1
MGET1
MPUT2
MGET2
```

Şekil 136. IBM MQ for Windows için ubbstxcn.cfg dosyası örneği

Not: *MachineName*makine adını ve dizin yollarını kuruluşunuzla eşleşecek şekilde değiştirin. Ayrıca, *MYQUEUEMANAGER* kuyruk yöneticisi adını, bağlanmak istediğiniz kuyruk yöneticisinin adıyla değiştirin.

IBM MQ for Windows için örnek ubbconfig dosyası Şekil 136 sayfa 1083içinde listelenmiştir. IBM MQ örnekleri dizininde `ubbstxcn.cfg` olarak sağlanır.

Örnek makefile (bkz. Şekil 137 sayfa 1084) IBM MQ for Windows için sağlanan `ubbstxmn.mak`olarak adlandırılır ve IBM MQ samples dizininde tutulur.

```
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
```

Şekil 137. IBM MQ for Windows için örnek TUXEDO makefile

Sunucu ortamını ve örneklerini oluşturmak için aşağıdaki adımları tamamlayın.

Yordam

1. Örnek uygulamanın oluşturulacağı bir uygulama dizini yaratın; örneğin:

   ```
   f:\tuxedo\apps\mqapp
   ```

2. Aşağıdaki örnek dosyaları IBM MQ örnek dizininden uygulama dizinine kopyalayın:
   - `amqstxmn.mak`
   - `amqstxen.env`
   - `ubbstxcn.cfg`
3. Kuruluşunuzda kullanılan dizin adlarını ve dizin yollarını ayarlamak için bu dosyaların her birini düzenleyin.
4. Düzenle `ubbstxcn.cfg` (bkz. Şekil 136 sayfa 1083) Bağlanmak istediğiniz makine adı ve kuyruk yöneticisine ilişkin ayrıntıları eklemek için.
5. *TUXDIR*`udataobj\rm` TUXEDO dosyasına aşağıdaki satırı ekleyin

   ```
   MQSERIES_XA_RMI;MQRMIXASwitchDynamic;MQMDIR\tools\lib64\mqmxa64.lib
   MQMDIR\tools\lib64\mqm.lib
   ```

   Yeni giriş, dosyada bir satır olmalıdır.
6. Aşağıdaki ortam değişkenlerini ayarlayın:

   ```
   TUXDIR=TUXDIR
   TUXCONFIG=APPDIR\tuxconfig
   FIELDTBLS=MQMDIR\tools\c\samples\amqstxvx.fld
   LANG=C
   ```

7. TUXEDO için bir TLOG aygıtı oluşturun. Bunu yapmak için `tmadmin -c`komutunu çağırın ve şu komutu girin:

   ```
   crdl -z APPDIR\TLOG
   ```

8. Yürürlükteki dizini *APPDIR*olarak ayarlayın ve örnek makefile `amqstxmn.mak` dosyasını dış proje makefile olarak çağırın. Örneğin, Microsoft Visual C++ ile aşağıdaki komutu verin:

```
msvc amqstxmn.mak
```

Tüm örnek programları oluşturmak için **build** (oluştur) seçeneğini belirleyin.

ALW *TUXEDO için örnek sunucu programı*

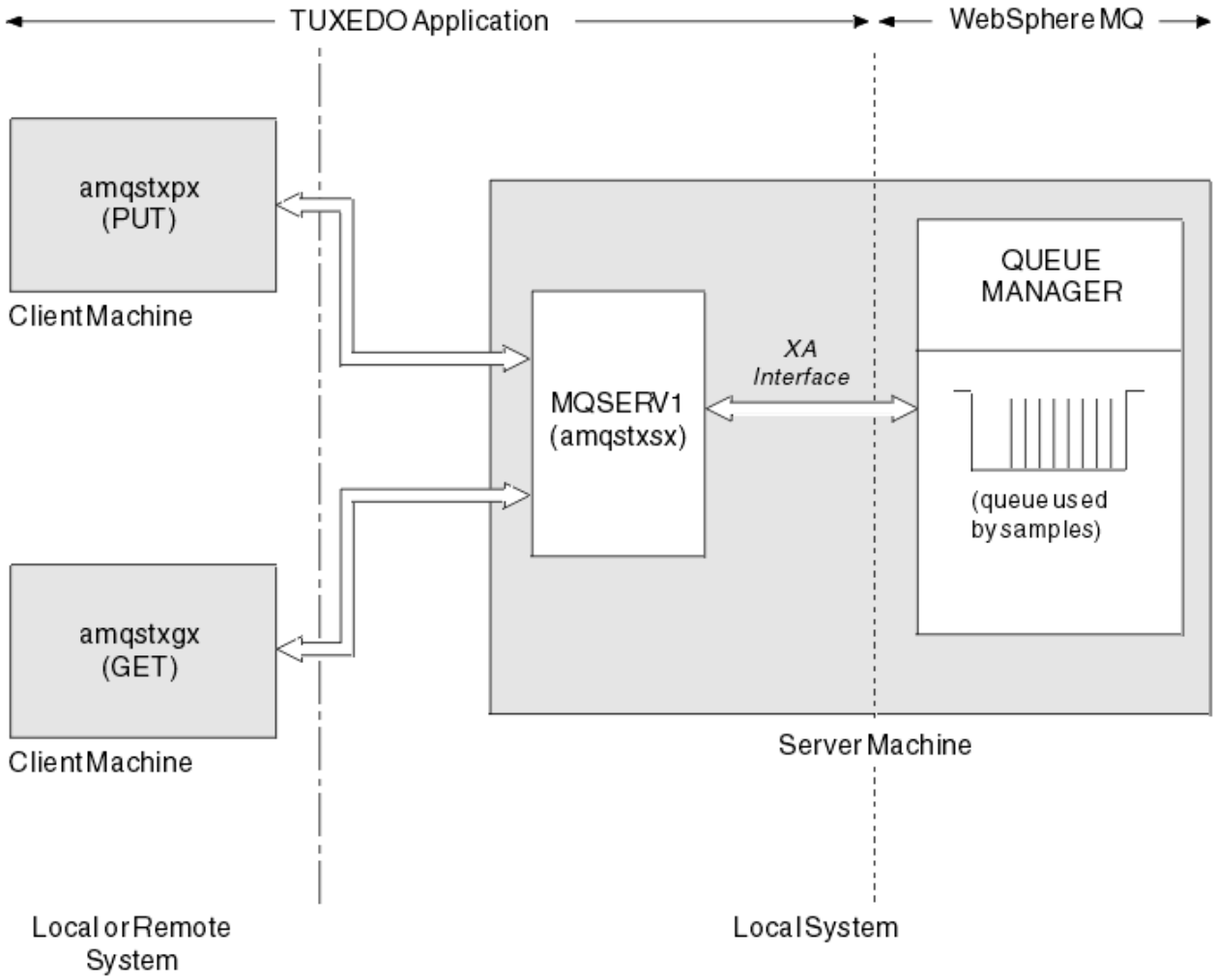
Örnek sunucu programı (amqstxsx), Put (amqstxpx.c) ve Get (amqstxgx.c) örnek programlarıyla çalışacak şekilde tasarlanmıştır. Örnek sunucu programı, TUXEDO başlatıldığında otomatik olarak çalışır.

Not: TUXEDO ' u başlatmadan önce kuyruk yöneticinizi başlatmanız gerekir.

Örnek sunucu iki TUXEDO hizmeti sağlar: MPUT1 ve MGET1:

- MPUT1 hizmeti PUT örneği tarafından yönlendirilir ve bir iletiyi TUXEDO tarafından denetlenen bir iş birimine yerleştirmek için eşitleme noktasında MQPUT1 ' i kullanır. PUT örneği tarafından sağlanan QName ve Message Text parametrelerini alır.
- MGET1 hizmeti, her ileti aldığında kuyruğu açar ve kapatır. GET örneği tarafından sağlanan QName ve Message Text parametrelerini alır.

Tüm hata iletileri, neden kodları ve durum iletileri TUXEDO günlük dosyasına yazılır.



Şekil 138. TUXEDO örnekleri birlikte nasıl çalışır?

ALW *TUXEDO için örnek program koy*

Bu örnek, kaynak yöneticisi olarak TUXEDO kullanarak eşitleme gösterip bir iletiyi toplu olarak bir kuyruğa birden çok kez koymanızı sağlar.

Koyma örneğinin başarılı olması için örnek sunucu programı `amqstxsx` çalışıyor olmalıdır; sunucu örnek programı kuyruk yöneticisine bağlanır ve XA arabirimini kullanır. Örneği çalıştırmak için şunu girin:

- `doputs -n queuename -b batchsize -c trancount -t message`

Örneğin:

- `doputs -n myqueue -b 5 -c 6 -t "Hello World"`

Bu, her biri beş ileti içeren altı toplu iş halinde `myqueue`adlı kuyruğa 30 ileti koyar. Herhangi bir sorun varsa, bir ileti kümesini geri alır, tersi durumda bunları kesinleştirir.

Tüm hata iletileri TUXEDO günlük dosyasına ve stderr dosyasına yazılır. Herhangi bir neden kodu stderr 'e yazılır.

ALW *TUXEDO örneğini al*

Bu örnek, toplu işler halinde bir kuyruktan ileti almanızı sağlar.

Örnek sunucu programının başarılı olması için `amqstxsx` örnek sunucu programı çalışıyor olmalıdır; örnek sunucu programı kuyruk yöneticisine bağlanır ve XA arabirimini kullanır. Örneği çalıştırmak için aşağıdaki komutu girin:

- `dogets -n queuename -b batchsize -c trancount`

Örneğin:

- `dogets -n myqueue -b 6 -c 4`

Bu, her biri dört ileti içeren, altı toplu iş halinde `myqueue`adlı kuyruktan 24 iletiyi alır. Bunu, `myqueue`üzerine 30 ileti koyan koyma örneğinden sonra çalıştırırsanız, `myqueue`üzerinde yalnızca altı ileti vardır. Toplu iş sayısı ve toplu iş boyutu, iletileri koyma ve alma arasında farklılık gösterebilir.

Tüm hata iletileri TUXEDO günlük dosyasına ve stderr dosyasına yazılır. Herhangi bir neden kodu stderr 'e yazılır.

Windows *Windows üzerinde SSPI güvenlik çıkışının kullanılması*

Bu konuda, Windows sistemlerinde SSPI kanal çıkışı programlarının nasıl kullanılacağı açıklanmaktadır. Sağlanan çıkış kodu iki biçimdedir: nesne ve kaynak.

Nesne kodu

Nesne kod dosyası `amqrspin.dll`olarak adlandırılır. Hem istemci hem de sunucu için, *MQ_INSTALLATION_PATH*`/exits/`*INSTALLATION_NAME* klasöründe IBM MQ for Windows ' un standart bir parçası olarak kurulur. Örneğin, `C:\Program Files\IBM\MQ\exits\installation2`. Standart bir kullanıcı çıkışı olarak yüklenir. Sağlanan güvenlik kanalı çıkışını çalıştırabilir ve kanal tanımınızda kimlik doğrulama hizmetlerini kullanabilirsiniz.

Bunu yapmak için aşağıdakilerden birini belirtin:

```
SCYEXIT('amqrspin(SCY_KERBEROS)')
SCYEXIT('amqrspin(SCY_NTLM)')
```

Sınırlı bir kanal için destek sağlamak üzere SVRCONN kanalında aşağıdakileri belirtin:

```
SCYDATA('remote_principal_name')
```

Burada *remote_principal_name* , *DOMAIN**user*biçimindedir. Güvenli kanal, uzak birincil kullanıcının adı *remote_principal_name*ile eşleşiyorsa oluşturulur.

Kerberos güvenlik etki alanında çalışan sistemler arasında sağlanan kanal çıkış programlarını kullanmak için, kuyruk yöneticisi için bir **servicePrincipalName** yaratın.

Kaynak kodu

Çıkış kaynak kod dosyası `amqsspin.c`olarak adlandırılır. `C:\Program Files\IBM\MQ\Tools\c\Samples`içinde.

Kaynak kodu değiştirirseniz, değiştirilen kaynağı yeniden derlemeniz gerekir.

SSPI üstbilgilerine derleme sırasında ve SSPI güvenlik kitaplıklarına önerilen ilişkili kitaplıklarla birlikte bağlantı sırasında erişilmesi gerekmesi dışında, ilgili platforma ilişkin diğer kanal çıkışlarıyla aynı şekilde derleyip bağlantı kurarsınız.

Aşağıdaki komutu yürütmeden önce, yolunuzda `cl.exe`ve Visual C++ kitaplığının ve `include` klasörünün bulunduğundan emin olun. Örneğin:

```
cl /VERBOSE /LD /MT /I path_to_Microsoft_platform_SDK\include
/I path_to_IBM_MQ\tools\c\include amqsspin.c /DSECURITY_WIN32
-link /DLL /EXPORT:SCY_KERBEROS /EXPORT:SCY_NTLM STACK:8192
```

Not: Kaynak kod, izleme ya da hata işleme için herhangi bir sağlama içermiyor. Kaynak kodu değiştirir ve kullanırsanız, kendi izleme ve hata işleme yordamlarınızı ekleyin.

Örneklerin uzak kuyruklar kullanılarak çalıştırılması

Bağlı kuyruk yöneticilerindeki örnekleri çalıştırarak uzaktan kuyruğa alma gösterebilirsiniz.

amqscos0.tst programı uzak kuyruğun yerel bir tanımını sağlar (SYSTEM.SAMPLE.REMOTE). Bu örnek tanımlamayı kullanmak için, DİĞER değerini, kullanmak istediğiniz ikinci kuyruk yöneticisinin adıyla değiştirin. Ayrıca, iki kuyruk yöneticiniz arasında bir ileti kanalı oluşturmanız gerekir; bunun nasıl gerçekleştirileceğine ilişkin bilgi için Kanalları tanımlamabaşlıklı konuya bakın.

İstek örnek programları, gönderdikleri iletilerin *ReplyToQMgr* alanına kendi yerel kuyruk yöneticisi adlarını koyar. Sorma ve Örnekleri Ayarla seçeneği, işledikleri istek iletilerinin *ReplyToQ* ve *ReplyToQMgr* alanlarında adı belirtilen kuyruk ve ileti kuyruğu yöneticisine yanıt iletileri gönderir.

Küme Kuyruğu İzleme örnek programı (AMQSCLM)

Bu örnek, iletileri, bağlı uygulamaları olan kuyrukların eşgörünümlerine yönlendirmek için yerleşik IBM MQ küme iş yükü dengeleme özelliklerini kullanır. Bu otomatik yön, hiçbir tüketen uygulamanın bağlı olmadığı bir küme kuyruğu eşgörünümünde iletilerin oluşturulmasını önler.

Genel Bakış

Farklı kuyruk yöneticilerindeki aynı kuyruk için birden çok tanımlaması olan bir küme ayarlayabilirsiniz. Bu yapılandırma, daha fazla kullanılabilirlik ve iş yükü dengeleme avantajı sağlar. Ancak, IBM MQ ' da iletilerin bağlı uygulamaların durumuna dayalı olarak bir kümede dağıtımını dinamik olarak değiştirme yeteneği yoktur. Bu nedenle, iletilerin işlenmesini sağlamak için her zaman kuyruğun her eşgörünümüne bir uygulama eklenmelidir.

Küme kuyruğu izleme örnek programı, bağlı uygulamaların durumunu izler. Program, yerleşik iş yükü dengeleme yapılandırmasını, iletileri, tüketen uygulamalar eklenmiş olarak kümelenmiş bir kuyruk eşgörünümlerine yönlendirecek şekilde dinamik olarak ayarlar. Bazı durumlarda bu program, her zaman bir kuyruğun her bir örneğine bağlı olan, tüketen bir uygulama gereksinmesini rahatlatmak için kullanılabilir. Ayrıca, kuyruğun bir eşgörünümünde kuyruğa alınan ve hiçbir uygulama eklenmeyen iletileri de yeniden gönderir. İletilerin yeniden gönderilmesiyle, iletilerin geçici olarak kapatılan bir uygulama etrafında yönlendirilebilmesine olanak sağlanır.

Bu program, uygulamaları sık sık bağlamak ve ayırmak yerine, uzun süre çalışan uygulamaların kullanıldığı yerlerde kullanılmak üzere tasarlanmıştır.

Küme kuyruğu izleme örnek programı, `amqsclma.c`C örnek dosyasının derlenmiş yürütülebilir programıdır.

Kümeler ve iş yükü hakkında daha fazla bilgi için İş yükü yönetimi için kümeleri kullanma konusuna bakın.

AMQSCLM: Örneği kullanmak için tasarım ve planlama
Küme kuyruğu izleme örnek programının nasıl çalıştığına, örnek programın çalıştırılacağı bir sistem ayarlarken dikkate alınması gereken noktalara ve örnek kaynak kodda yapılabilecek değişikliklere ilişkin bilgiler.

Tasarım

Küme kuyruğu izleme örnek programı, bağlı uygulamaları olan yerel kümelenmiş kuyrukları izler. Program, kullanıcı tarafından belirlenen kuyrukları izler. Kuyruğun adı belirli olabilir; örneğin, APP.TEST01ya da soysal. Soysal adlar, PCF ' ye (Programlanabilir Komut Biçimi) uygun biçimde olmalıdır. Soysal ad örnekleri: APP.TEST*ya da APP*.

İzlenecek bir yerel kuyruk eşgörünümüne sahip olan bir kümedeki her kuyruk yöneticisi, küme kuyruğu izleme örnek programının bir eşgörünümünün ona bağlanmasını gerektirir.

Dinamik ileti yöneltmesi

Küme kuyruğu izleme örnek programı, o kuyrukta herhangi bir tüketici olup olmadığını belirlemek için bir kuyruğun **IPPROCS** (giriş işlemi sayısı için aç) değerini kullanır. 0 'dan büyük bir değer, kuyrukta en az bir tüketen uygulama eklendiğini gösterir. Bu kuyruklar etkindir. 0 değeri, kuyruğun ekli bir tüketen programı olmadığını gösterir. Bu kuyruklar etkin değil.

Bir kümede birden çok eşgörünümü olan kümelenmiş bir kuyruk için IBM MQ , iletilerin hangi eşgörünümlere gönderileceğini saptamak üzere her kuyruk eşgörünümünün küme iş yükü önceliği özelliğini **CLWLPRTY** kullanır. IBM MQ , iletileri en yüksek **CLWLPRTY** değerine sahip bir kuyruğun kullanılabilir eşgörünümlerine gönderir.

Küme kuyruğu izleme örnek programı, yerel **CLWLPRTY** değerini 1 olarak ayarlayarak bir küme kuyruğunu etkinleştirir. Program, **CLWLPRTY** değerini 0 olarak ayarlayarak bir küme kuyruğunu devre dışı bırakır.

IBM MQ kümeleme teknolojisi, kümelenmiş bir kuyruğun güncellenen **CLWLPRTY** özelliğini kümedeki tüm ilgili kuyruk yöneticilerine geçirir. Örneğin,

- İletileri kuyruğa koyan bağlı bir uygulamaya sahip bir kuyruk yöneticisi.
- Aynı kümede aynı adı taşıyan yerel bir kuyruğa sahip olan bir kuyruk yöneticisi.

Yayma, kümenin tam havuz kuyruğu yöneticileri kullanılarak yapılır. Küme kuyruğuna ilişkin yeni iletiler, küme içindeki en yüksek **CLWLPRTY** değerine sahip yönetim ortamlarına yönlendirilir.

Kuyruğa alınan ileti aktarımı

CLWLPRTY değerinin dinamik olarak değiştirilmesi, yeni iletilerin yönlendirilmesini etkiler. Bu dinamik değişiklik, bağlı tüketicileri olmayan bir kuyruk eşgörünümünde kuyruğa alınmış iletileri ya da değiştirilen bir **CLWLPRTY** değeri kümeye yayılmadan önce iş yükü dengeleme mekanizmasından geçmiş iletileri etkilemez. Sonuç olarak, iletiler etkin olmayan herhangi bir kuyrukta kalır ve tüketen bir uygulama tarafından işlenmez. Bu sorunu çözmek için, küme kuyruğu izleme örnek programı, tüketicisi olmayan yerel bir kuyruktan ileti alabilir ve bu iletileri, tüketicilerin bağlı olduğu aynı kuyruğun uzak örneklerine gönderebilir.

Küme kuyruğu izleme örnek programı, iletileri alarak (**MQGET** kullanarak) etkin olmayan bir yerel kuyruktan bir ya da daha çok etkin uzak kuyruğa ileti aktarır. ve ileti koyma (**MQPUT** kullanarak) aynı kümelenmiş kuyruğa. Bu aktarma, IBM MQ küme iş yükü yönetiminin yerel kuyruk örneğininkinden daha yüksek bir CLWLPRTY değerine dayalı olarak farklı bir hedef eşgörünüm seçmesine neden olur. İleti aktarımı sırasında ileti kalıcılığı ve bağlamı korunur. İleti sırası ve bağlama seçenekleri korunmaz.

Planlama

Küme kuyruğu izleme örnek programı, uygulamaların tüketilmesinde bir değişiklik olduğunda küme yapılandırmasını değiştirir. Değişiklikler, küme kuyruğu izleme örnek programının izleme kuyrukları

olduğu kuyruk yöneticilerinden kümedeki tam havuz kuyruğu yöneticilerine iletilir. Tam havuz kuyruğu yöneticileri, yapılandırma güncellemelerini işler ve bunları kümedeki tüm ilgili kuyruk yöneticilerine yeniden gönderir. İlgili kuyruk yöneticileri, aynı ada sahip kümelenmiş kuyruklara sahip olan (küme kuyruğu izleme örnek programının bir eşgörünümünün çalıştığı) kuyruk yöneticilerini ve bir uygulamanın son 30 gün içinde kendisine ileti yerleştirmek için küme kuyruğunu açtığı herhangi bir kuyruk yöneticisini içerir.

Değişiklikler kümede zamanuyumsuz olarak işlenir. Bu nedenle, her değişiklikten sonra, kümedeki farklı kuyruk yöneticileri belirli bir süre için yapılandırmanın farklı görünümlerine sahip olabilir.

Küme kuyruğu izleme örnek programı yalnızca, tüketen uygulamaların nadiren bağlandığı ya da ayrıldığı sistemler için uygundur; örneğin, uzun süre çalışan tüketen uygulamalar. Uygulamaları kullanan uygulamaların yalnızca kısa dönemler için eklendiği sistemleri izlemek için kullanıldığında, yapılandırma güncellemeleri dağıtılırken oluşan gecikme, kümedeki kuyruk yöneticilerinin tüketicilerin bağlı olduğu kuyrukların yanlış bir görünümüne sahip olmasına neden olabilir. Bu gecikme, yanlış yönlendirilmiş iletilere neden olabilir.

Birçok kuyruk izlenirken, tüm kuyruklar boyunca bağlı tüketicilerin göreceli olarak düşük bir değişiklik oranı, küme genelinde küme yapılandırma trafiğini artırabilir. Küme yapılandırması trafiğinin artması, aşağıdaki kuyruk yöneticilerinden birinde ya da daha fazlasında aşırı yüke neden olabilir.

- Küme kuyruğu izleme örnek programının çalıştığı kuyruk yöneticileri
- Tam havuz kuyruğu yöneticileri
- İletileri kuyruğa koyan bağlı bir uygulamaya sahip bir kuyruk yöneticisi
- Aynı kümede aynı adı taşıyan bir yerel kuyruğa sahip olan bir kuyruk yöneticisi

Tam havuz kuyruğu yöneticilerindeki işlemci kullanımı değerlendirilmelidir. Ek işlemci kullanımı, SYSTEM.CLUSTER.COMMAND.QUEUE. Bu kuyrukta iletiler oluşursa, tüm havuz kuyruğu yöneticilerinin sistemdeki küme yapılandırma değişikliği hızına ayak uyduramadığını gösterir.

Küme kuyruğu izleme örnek programı tarafından çok sayıda kuyruk izlenirken, örnek program ve kuyruk yöneticisi tarafından gerçekleştirilen bir iş miktarı vardır. Bu iş, bağlı tüketiciler üzerinde herhangi bir değişiklik olmadığında bile gerçekleştirilir. **-i** bağımsız değişkeni, izleme döngüsünün sıklığı azaltılarak, yerel sistemdeki örnek programın işlemci kullanımını azaltmak için değiştirilebilir.

Çok fazla etkinliğin saptanmasına yardımcı olmak için, küme kuyruğu izleme örnek programı yoklama aralığı, geçen işleme süresi ve yapılandırma değişiklikleri sayısı başına ortalama işleme süresini bildirir. Raporlar, hangisi daha önceyse, her 30 dakikada bir **CLM0045I**bilgi iletisiyle ya da her 600 anket aralığına teslim edilir.

Küme kuyruğu izleme kullanım gereksinimleri

Küme kuyruğu izleme örnek programının gereksinimleri ve kısıtlamaları vardır. Bu kısıtlamaların bazılarını nasıl kullanılacağına ilişkin olarak değiştirmek için sağlanan örnek kaynak kodunu değiştirebilirsiniz. Bu bölümde listelenen örnekler, yapılabilecek değişiklikleri ayrıntılı olarak gösterir.

- Küme kuyruğu izleme örnek programı, kullanan uygulamaların bağlı olduğu ya da bağlı olmadığı kuyrukları izlemek için kullanılmak üzere tasarlanmıştır. Sistemde sık sık bağlanan ve ayrılan uygulamalar varsa, örnek program kümenin tamamında aşırı küme yapılandırması etkinliği oluşturabilir. Bu, kümedeki kuyruk yöneticilerinin performansını etkileyebilir.
- Küme kuyruğu izleme örnek programı, temel IBM MQ sistemine ve küme teknolojisine bağlıdır. İzlenmekte olan kuyrukların sayısı, izleme sıklığı ve her bir kuyruktaki durum değişikliğinin sıklığı, genel sistemdeki yükü etkiler. İzlenecek kuyruklar ve izlemenin yoklama aralığı seçilirken bu etkenler dikkate alınmalıdır.
- Küme kuyruğu izleme örnek programının bir eşgörünümü, izlenecek bir kuyruk eşgörünümüne sahip olan kümedeki her kuyruk yöneticisine bağlanmalıdır. Örnek programın, kuyrukların sahibi olmayan kümedeki kuyruk yöneticilerine bağlanması gerekmez.
- Küme kuyruğu izleme örnek programı, gerekli tüm IBM MQ kaynaklarına erişmek için uygun yetkiyle çalıştırılmalıdır. Örneğin,

  - Bağlanılacak kuyruk yöneticisi
  - SYSTEM.ADMIN.COMMAND.QUEUE
  - İleti aktarımı gerçekleştirildiğinde izlenecek tüm kuyruklar
- Küme kuyruğu izleme örnek programı bağlı olan her kuyruk yöneticisi için komut sunucusu çalışıyor olmalıdır.
- Küme kuyruğu izleme örnek programının her bir eşgörünümü, bağlı olduğu kuyruk yöneticisinde yerel (kümelenmemiş) bir kuyruğun özel olarak kullanılmasını gerektirir. Bu yerel kuyruk, örnek programı denetlemek ve kuyruk yöneticisinin komut sunucusuna yapılan yanıt iletilerini almak için kullanılır.
- Küme kuyruğu izleme örnek programının tek bir eşgörünümü tarafından izlenecek tüm kuyruklar aynı kümede olmalıdır. Bir kuyruk yöneticisinin izleme gerektiren birden çok kümede kuyrukları varsa, örnek programın birden çok eşgörünümü gerekir. Her eşgörünüm, denetim ve yanıt iletileri için yerel bir kuyruğa ihtiyaç duyar.
- İzlenecek tüm kuyruklar tek bir kümede olmalıdır. Küme ad listesi kullanacak şekilde yapılandırılan kuyruklar izlenmez.
- Etkin olmayan kuyruklardan ileti aktarımının etkinleştirilmesi isteğe bağlıdır. Küme kuyruğu izleme örnek programının yönetim ortamı tarafından izlenmekte olan tüm kuyruklar için geçerlidir. İzlenmekte olan kuyrukların yalnızca bir altkümesi için ileti aktarımının etkinleştirilmesi gerekiyorsa, küme kuyruğu izleme örnek programının iki eşgörünümü gerekir. Örnek programlardan birinde ileti aktarımı etkin, diğerinde ileti aktarımı devre dışı bırakıldı. Örnek programın her bir eşgörünümü, denetim ve yanıt iletileri için yerel bir kuyruğa ihtiyaç duyar.
- IBM MQ küme iş yükü dengelemesi, varsayılan olarak, bir koyma uygulamasının bağlı olduğu aynı kuyruk yöneticisinde bulunan kümelenmiş kuyrukların eşgörünümlerine ileti gönderir. Yerel kuyruk aşağıdaki durumlarda etkin değilken bu işlem geçersiz kılınmalıdır:
  - Uygulamaların, izlenmekte olan etkin olmayan bir kuyruk eşgörünümlerinin sahibi olan kuyruk yöneticilerine bağlanması
  - Kuyruğa alınan iletiler, etkin olmayan kuyruklardan etkin kuyruklara aktarılıyor.

  Kuyruktaki yerel iş yükü dengeleme tercihi, `CLWLUSEQ` değeri `ANY`olarak ayarlanarak statik olarak geçersiz kılınabilir. Yerel kuyruklara konan bu yapılanış iletilerinde, yerel tüketen uygulamalar olsa bile, iş yükünü dengelemek için yerel ve uzak kuyruk eşgörünümlerine dağıtılır. Diğer bir seçenek olarak, kuyruk etkin durumdayken yalnızca yerel iletilerin bir kuyruğun yerel eşgörünümlerine gitmesiyle sonuçlanan bağlı tüketiciler yoksa, küme kuyruğu izleme örnek programı **CLWLUSEQ** değerini geçici olarak `ANY` olarak ayarlayabilir.
- IBM MQ sistemi ve uygulamaları, izlenecek kuyruklar ya da kullanılmakta olan kanallar için **CLWLPRTY** kullanmamalıdır. Tersi durumda, **CLWLPRTY** kuyruk özniteliklerinde küme kuyruğu izleme örnek programının işlemleri istenmeyen etkiler olabilir.
- Küme kuyruğu izleme örnek programı, çalıştırma zamanı bilgilerini bir rapor dosyaları kümesine kaydeder. Bu raporları saklamak için bir dizin gereklidir ve küme kuyruğu izleme örnek programının bu dizine yazma yetkisi olmalıdır.

AMQSCLM: Örneği hazırlama ve çalıştırma
Küme kuyruğu izleme örneği, yerel olarak bir kuyruk yöneticisine bağlı ya da bir kanal üzerinden bağlı bir istemci olarak çalıştırılabilir. Örneğin, kuyruk yöneticisi her çalıştığında çalışması gerekir; yerel olarak çalıştırıldığında, kuyruk yöneticisi hizmeti olarak, örneği kuyruk yöneticisiyle birlikte otomatik olarak başlatıp durduracak şekilde yapılandırılabilir.

Başlamadan önce

Küme kuyruğu izleme örneği çalıştırılmadan önce aşağıdaki adımlar tamamlanmalıdır.

1. Örneğin iç kullanımı için her kuyruk yöneticisinde bir çalışma kuyruğu yaratın.

   Örneğin her bir eşgörünümü, dışlayıcı iç kullanım için yerel bir kümelenmemiş kuyruk gerektirir. Kuyruğun adını seçebilirsiniz. Örnek, `AMQSCLM.CONTROL.QUEUE`adını kullanır. Örneğin, Windows' da aşağıdaki **MQSC** komutunu kullanarak bu kuyruğu oluşturabilirsiniz:

```
DEFINE QLOCAL(AMQSCLM.CONTROL.QUEUE)
```

MAXDEPTH ve **MAXMSGL** değerlerini varsayılan olarak bırakabilirsiniz.

2. Hata ve bilgi iletisi günlükleri için bir dizin oluşturun.

   Örnek, tanılama iletilerini rapor dosyalarına yazar. Dosyaların saklanacağı dizini seçmelisiniz. Örneğin, Windowsüzerinde aşağıdaki komutu kullanarak bir dizin oluşturabilirsiniz:

   ```
   mkdir C:\AMQSCLM\rpts
   ```

   Örnek tarafından oluşturulan rapor dosyaları aşağıdaki adlandırma kurallarına sahip:

   ```
   QmgrName.ClusterName.RPT0n.LOG
   ```

3. (İsteğe bağlı) Küme kuyruğu izleme örneğini IBM MQ hizmeti olarak tanımlayın.

   Kuyrukları izlemek için, örneğin her zaman çalışıyor olması gerekir. Küme kuyruğu izleme örneğinin her zaman çalıştığından emin olmak için örneği bir kuyruk yöneticisi hizmeti olarak tanımlayabilirsiniz. Örneğin hizmet olarak tanımlanması, kuyruk yöneticisi başlatıldığında AMQSCLM ' nin başlatıldığı anlamına gelir. Küme kuyruğu izleme örneğini IBM MQ hizmeti olarak tanımlamak için aşağıdaki örneği kullanabilirsiniz.

   ```
   define service(AMQSCLM) +
         descr('Active Cluster Queue Message Distribution Monitor - AMQSCLM') +
         control(qmgr) +
         servtype(server) +
         startcmd('MQ_INSTALLATION_PATH\tools\c\samples\Bin\AMQSCLM.exe') +
         startarg('-m +QMNAME+ -c CLUSTER1 -q ABC* -r AMQSCLM.CONTROL.QUEUE -l
   c:\AMQSCLM\rpts') +
         stdout('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stdout.log') +
         stderr('C:\AMQSCLM\rpts\+QMNAME+.TSTCLUS.stderr.log')
   ```

Tanım	Açıklama
service	Hizmet adını belirtir. Hizmet adını seçebilirsiniz.
descr	Hizmetin metin tanımlamasını belirtir.
control	Hizmetin kuyruk yöneticisiyle aynı anda başlatıldığını ve durduğunu gösterir.
servtype	Bu kuyruk yöneticisi için bir kerede tek bir yönetim ortamı yürütülebileceği anlamına gelen bir sunucu hizmeti nesnesini gösterir.
startcmd	Programın yerini ve adını belirler.
startarg	Örneğin bağımsız değişkenlerini belirtir. *+ QMNAME +*kullanımına dikkat edin. Kuyruk yöneticisinin adı otomatik olarak değiştirilir.
stdout	Standart çıkışın yeniden yönlendirileceği tam olarak nitelenmiş dosya adı. Örnek yalnızca bu dosyaya, örneğin sonlandırıldığını doğrulayan iletiler yazar. Standart hata dosyası, örnek sonlandırma işleminin daha önceki bir aşamasında zaten kapatıldığından örnek bunu yapar.
stderr	Standart hata çıkışının yeniden yönlendirildiği tam olarak nitelenmiş dosya adı. Örnek, örnek sonlandırılmadan önce hata iletilerini standart hata dosyasına yazar.

Bu görev hakkında

Bu görev, küme kuyruğu izleme örneğini farklı şekillerde başlatmanızı ve durdurmanızı sağlar. Ayrıca, örneği, izlenmekte olan kuyruklar hakkında istatistiksel bilgiler içeren rapor dosyaları oluşturan bir kipte çalıştırmanızı da sağlar.

Örnek program aşağıdaki komut kullanılarak çalıştırılabilir.

```
AMQSCLM -m QMgrName -c ClusterName (-q QNameMask| -f QListFile) -r MonitorQName
[-l ReportDir] [-t] [-u ActiveVal] [-i Interval] [-d] [-s] [-v]
```

Çizelge, küme kuyruğu izleme örneğiyle kullanılabilecek bağımsız değişkenleri ve her biriyle ilgili ek bilgileri listeler.

Bağımsız Değişken	Değişken	Daha Fazla Bilgi
-m	**QMgrName**	İzlenecek kuyruk yöneticisi.
-c	**ClusterName**	İzlenecek kuyrukları içeren küme.
-q	**QNameMask**	İzlenecek kuyruk ya da kuyruklar. Sondaki * , adları sıfır ya da daha çok sondaki karakterle eşleşen tüm kuyrukları izler.
-f	**QListFile**	İzlenecek kuyruk adları ya da kuyruk adı maskeleri listesini içeren dosyanın tam yolu ve dosya adı. Dosya, satır başına bir kuyruk adı/maskesi içermelidir. **-q** ya da **-f**belirtebilirsiniz, ancak her ikisini birden belirleyemezsiniz.
-r	**MonitorQName**	Örnek tarafından özel olarak kullanılan yerel kuyruk.
-l	**ReportDir**	Günlüğe kaydedilen bilgi iletilerinin bir kaydırma kümesinde saklanacağı dizin yolu[9]rapor dosyaları.
-t		(İsteğe bağlı) Kuyruktaki iletilerin etkin olmayan yerel kuyruklardan etkin kuyruklara aktarılmasını sağlar. Etkinleştirilmezse, yalnızca kümeye giren yeni iletiler dinamik olarak bir kuyruğun etkin örneklerine yönlendirilir.
-u	**ActiveVal**	(İsteğe bağlı) İzlenen bir kuyruk eşgörünümünün **CLWLUSEQ** özelliğini etkin olmadığında otomatik olarak ANY değerine ve etkin olduğunda **ActiveVal** değerine değiştirir. **ActiveVal** , LOCAL ya da QMGRolabilir. Bu bağımsız değişken, koyma uygulamalarının aynı kuyruk yöneticisine bağlandığı ya da ileti aktarımının etkinleştirildiği bir sistemde ayarlanmazsa, izlenen kuyrukların **CLWLUSEQ** değeri ANYya da kuyruk yöneticisi ANYdeğerine sahip QMGR olmalıdır.
-i	**Interval**	(İsteğe bağlı) İzleme programının kuyrukları denetleyeceği saniye cinsinden zaman aralığı. Varsayılan değer 300 saniyedir (5 dakika).
-d		(İsteğe bağlı) Ek tanılama çıkışını etkinleştirir. Hata ayıklama çıkışı, sistemi ilk kez yapılandırırken ya da örnek kodla çalışırken yararlı olabilir.
-s		(İsteğe bağlı) Aralık başına minimum istatistiksel çıktı sağlar.
-v		(İsteğe bağlı) Rapor bilgilerini rapor dosyalarına ek olarak standard out' e de günlüğe kaydet.

Bağımsız değişken listesi örnekleri:

```
-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
```

Örnek kuyruk listesi dosyası:

[9] Her kuyruk yöneticisi ve kuyruk birleşimi için, dolu olduğunda üzerine yazılacak, değişmez büyüklüklü bir günlük dosyası üretilir. Kaydedici her zaman aynı dosyaya yazar ve dosyanın önceki iki sürümünü de tutar.

```
Q1
QUEUE.*
ABC
ABD
```

Yordam

1. Küme kuyruğu izleme örneğini başlatın. Örneği aşağıdaki yollardan biriyle başlatabilirsiniz:
   - Uygun kullanıcı yetkileriyle bir komut istemi kullanın.
   - Örnek bir IBM MQ hizmeti olarak yapılandırıldıysa, MQSC **START SERVICE** komutunu kullanın.

   Bağımsız değişken listesi her iki durumda da aynıdır.

   Örnek, program başlatıldıktan sonra 10 saniye boyunca kuyrukları izlemeye başlamaz. Bu gecikme, uygulamaların önce izlenen kuyruklara bağlanmasını sağlayarak kuyruğun etkin durumunda gereksiz değişiklikler yapılmasını önler.
2. Küme kuyruğu izleme örneğini durdurun. Örnek, kuyruk yöneticisi durdurulduğunda, durdurulduğunda, susturulduğunda ya da kuyruk yöneticisiyle bağlantı kesildiğinde otomatik olarak durdurulur. Kuyruk yöneticisini sona erdirmeden örneği durdurmanın yolları vardır:
   - Alma işlevini geçersiz kılmak için, örnek tarafından özel olarak kullanılan yerel kuyruğu yapılandırın.
   - Örnek tarafından özel olarak kullanılan yerel kuyruğa **CorrelId** / `"STOP CLUSTER MONITOR\0\0\0\0"`içeren bir ileti gönderin.
   - Örnek işlemi sonlandırın. Bu, kalıcı olmayan iletilerin etkin kuyruklara aktarılmasına neden olabilir. Ayrıca, örnek tarafından kullanılan yerel kuyruğun, sonlandırmadan sonra birkaç saniye boyunca açık tutulmasıyla da sonuçlanabilir. Bu durum, küme kuyruğu izleme örneğinin yeni bir eşgörünümünün hemen başlatılmasını önler.

   Örnek bir IBM MQ hizmeti olarak başlatıldıysa, **STOP SERVICE** ' in bir etkisi yoktur.
   Kuyruk yöneticisinde yapılandırılmış bir **STOP SERVICE** düzeneği olarak açıklanan sonlandırma yöntemlerinden biri kullanılabilir.

Sonraki adım

Örneğin durumunu denetleyin.

Raporlama etkinleştirildiyse, durum için rapor dosyalarını gözden geçirebilirsiniz. En güncel rapor dosyasını gözden geçirmek için aşağıdaki komutu kullanın:

```
QMgrName.ClusterName.RPT01.LOG
```

Eski rapor dosyalarını gözden geçirmek için aşağıdaki komutları kullanın:

```
QMgrName.ClusterName.RPT02.LOG
QMgrName.ClusterName.RPT03.LOG
```

Rapor dosyaları yaklaşık 1 MB boyutuna büyür. `RPT01` dosyası dolduğunda yeni bir `RPT01` dosyası oluşturulur. Eski `RPT01` dosyası `RPT02`olarak yeniden adlandırılır. `RPT02` , `RPT03`olarak yeniden adlandırıldı. Eski `RPT03` atılır.

Örnek, aşağıdaki durumlarda bilgi iletileri yaratır:

- başlangıçta
- sonlandırmada
- bir kuyruğu işaretlediğinde **ACTIVE** ya da **INACTIVE**
- etkin olmayan bir kuyruktaki iletileri etkin bir eşgörünüme ya da eşgörünümlere yeniden gönderdiğinde

Örnek, dikkat edilmesi gereken bir sorunu bildirmek için bir hata iletisi *CLMnnnnE* oluşturur.

Her 30 dakikada bir, örnek raporlar yoklama aralığı başına ortalama işleme süresi ve geçen işleme süresi. Bu bilgiler CLM0045Iiletisinde tutulur.

İstatistiksel iletiler etkinleştirildiğinde **-s**, örnek her kuyruk denetimiyle ilgili aşağıdaki istatistiksel bilgileri raporlar:

- Kuyrukların işlenmesi için geçen süre (milisaniye)
- Denetlenen kuyruk sayısı
- Yapılan etkin/etkin olmayan değişikliklerin sayısı
- Aktarılan ileti sayısı

Bu bilgiler CLM0048Iiletisinde raporlanır.

Rapor dosyaları hata ayıklama kipinde hızla büyüyebilir ve hızla kaydırılabilir. Bu durumda, tek tek dosyalar için 1 MB boyut sınırı aşılabilir.

AMQSCLM: Sorun giderme
Aşağıdaki bölümlerde, örnek kullanılırken karşılaşılabilecek senaryolarla ilgili bilgiler bulunur. Bir senaryoya ilişkin olası açıklamalar ve senaryonun nasıl çözümleneceğine ilişkin seçenekler hakkında bilgiler sağlanır.

Senaryo: AMQSCLM başlatılmıyor

Olası açıklama: Yanlış sözdizimi.

İşlem: Doğru sözdizimi için standart hata çıkışını denetleyin

Olası açıklama: Kuyruk yöneticisi kullanılamıyor.

İşlem: İleti tanıtıcısı için rapor dosyasını denetleyin CLM0010E.

Olası açıklama: Rapor kütüğü ya da kütükleri açılamıyor ya da yaratılamıyor.

İşlem: Başlatma sırasında hata iletileri için standart hata çıkışını denetleyin.

Senaryo: AMQSCLM, bir kuyruğu ACTIVE ya da INACTIVE olarak değiştirmiyor

Olası açıklama: Kuyruk, izlenecek kuyruklar listesinde yok.

İşlem: **-q** ve **-f** parametre değerlerini denetleyin.

Olası açıklama: Kuyruk, doğru kümedeki bir yerel kuyruk değil.

İşlem: Kuyruğun yerel ve doğru kümede olup olmadığını denetleyin.

Olası açıklama: AMQSCLM, bu kuyruk yöneticisi ve küme için çalışmıyor.

İşlem: İlgili kuyruk yöneticisi ve küme için AMQSCLM ' yi başlatın.

Olası açıklama: Kuyruk, tüketicisi olmadığı için INACTIVE, **CLWLPRTY** = 0 olarak bırakıldı. Diğer bir seçenek olarak, en az 1 tüketici olduğu için ETKİN **CLWLPRTY** > =1 kalır.

İşlem: Kullanan uygulamaların kuyruğa eklenip eklenmediğini denetleyin.

Olası açıklama: Kuyruk yöneticisinin komut sunucusu çalışmıyor.

Eylem: Rapor dosyalarında hata olup olmadığını denetleyin.

Senaryo: İletiler INACTIVE kuyrukları çevresinde yöneltilmiyor

Olası açıklama: İletiler doğrudan etkin olmayan kuyruğun sahibi olan kuyruk yöneticisine yerleştirilir ve kuyruğun **CLWLUSEQ** değeri ANYdeğildir ve **-u** bağımsız değişkeni AMQSCLM için kullanılmaz.

İşlem: İlgili kuyruk yöneticisinin **CLWLUSEQ** değerini denetleyin ya da AMQSCLM için **-u** bağımsız değişkeninin kullanıldığından emin olun.

Olası açıklama: Herhangi bir kuyruk yöneticinde etkin kuyruk yok. İletiler, bir kuyruk etkin oluncaya kadar tüm etkin olmayan kuyruklarda eşit olarak dengelenir.

Yapılması gereken: Tüm kuyruk yöneticilerindeki kuyrukların durumunu denetleyin.

Olası açıklama: İletiler, kümedeki farklı bir kuyruk yöneticisine, etkin olmayan kuyruğun sahibine yerleştirilir ve güncellenen 0 **CLWLPRTY** değeri, koyma uygulamasının kuyruk yöneticisine yayılmaz.

Yapılması gereken: İzlenen kuyruk yöneticisi ile tam havuz kuyruk yöneticisi arasındaki küme kanallarının çalışıp çalışmadığını denetleyin. Koyma kuyruğu yöneticisi ile tam havuz kuyruk yöneticisi arasındaki kanalların çalışıp çalışmadığını denetleyin. İzlenen, yerleştirme ve tam havuz kuyruğu yöneticilerinin hata günlüklerini denetleyin.

Olası açıklama: Uzak kuyruk eşgörünümleri etkin (CLWLPRTY=1), ancak yerel kuyruk yöneticisindeki küme gönderen kanalı çalışmadığından, iletiler bu kuyruk eşgörünümlerine yöneltilemiyor.

Yapılması gereken: Kuyruğun etkin bir eşgörünümü ile, küme gönderen kanallarının yerel kuyruk yöneticisinden uzak kuyruk yöneticisine ya da yöneticilere durumunu denetleyin.

Senaryo: AMQSCLM, etkin olmayan bir kuyruktan ileti aktarmıyor

Olası açıklama: İleti aktarımı etkinleştirilmedi (**-t**).

İşlem: İleti aktarımının etkinleştirildiğinden emin olun (**-t**).

Olası açıklama: Kuyruk, izlenecek kuyruklar listesinde yok.

İşlem: -q ve **-f** parametre değerlerini denetleyin.

Olası açıklama: AMQSCLM, aynı kuyruğun eşgörünümlerine sahip olan bu işlem ya da kümedeki diğer kuyruk yöneticileri için çalışmıyor.

İşlem: AMQSCLM ' yi başlatın.

Olası açıklama: Kuyrukta **CLWLUSEQ** = LOCAL ya da **CLWLUSEQ** = QMGRvar ve **-u** bağımsız değişkeni ayarlanmadı.

Yapılması gereken: -u değiştirgesini ayarlayın ya da kuyruğu ya da kuyruk yöneticisi yapılanışını ANYolarak değiştirin.

Olası açıklama: Kümede kuyruğun etkin eşgörünümü yok.

İşlem: CLWLPRTY değeri 1 ya da daha büyük olan kuyruk eşgörünümlerini denetleyin.

Olası açıklama: AMQSCLM bu uzak eşgörünümleri izlemediği için, uzak kuyruk eşgörünümlerinin tüketicileri var (**IPPROCS** > = 1), ancak bu kuyruk yöneticilerine ilişkin etkin değil (**CLWLPRTY** = 0).

İşlem: AMQSCLM ' nin bu kuyruk yöneticisinde çalıştığından ve/ya da kuyruğun, **-q** ve **-f** parametre değerlerini denetleyerek izlenecek kuyruklar listesinde olduğundan emin olun.

Olası açıklama: Uzak kuyruk eşgörünümleri etkin (**CLWLPRTY** = 1), ancak yerel kuyruk yöneticisinde (**CLWLPRTY** = 0) etkin değil olarak görülüyor. Bu durum, güncellenen **CLWLPRTY** değerinin bu kuyruk yöneticisine yayılmaması nedeniyle ortaya çıktı.

Yapılması gereken: Uzak kuyruk yöneticilerinin kümedeki tam havuz kuyruğu yöneticilerinden en az birine bağlandığından emin olun. Tam havuz kuyruğu yöneticilerinin doğru çalıştığından emin olun. Tam havuz kuyruğu yöneticileri ile izlenen kuyruk yöneticileri arasındaki kanalların çalışıp çalışmadığını denetleyin.

Olası açıklama: İletiler kesinleştirilmediği için alınamaz.

Yapılması gereken: Gönderme uygulamasının doğru çalışıp çalışmadığını denetleyin.

Olası açıklama: AMQSCLM, iletilerin kuyruğa alındığı yerel kuyruğa erişemez.

İşlem: AMQSCLM ' nin kuyruğa erişmek için yeterli yetkiye sahip bir kullanıcı olarak çalışıp çalışmadığını denetleyin.

Olası açıklama: Kuyruk yöneticisinin komut sunucusu çalışmıyor.

Yapılması gereken: Kuyruk yöneticisinin komut sunucusunu başlatın.

Olası açıklama: AMQSCLM bir hatayla karşılaştı.

Eylem: Rapor dosyalarında hata olup olmadığını denetleyin.

Olası açıklama: Uzak kuyruk eşgörünümleri etkin (CLWLPRTY=1), ancak yerel kuyruk yöneticisinden küme gönderen kanalı çalışmadığından, iletiler bu kuyruk eşgörünümlerine aktarılamıyor. Bu genellikle amqsclm rapor günlüğünde bir CLM0030W uyarısıyla birlikte kullanılır.

Yapılması gereken: Kuyruğun etkin bir eşgörünümü ile, küme gönderen kanallarının yerel kuyruk yöneticisinden uzak kuyruk yöneticisine ya da yöneticilere durumunu denetleyin.

ALW *Bağlantı Uç Noktası Arama (CEPL) için örnek program*

IBM MQ Connection Endpoint Lookup örneği, IBM MQ kullanıcılarına Tivoli Directory Servergibi bir LDAP havuzundan bağlantı tanımlamalarını almanın bir yolunu sunan basit ancak güçlü bir çıkış modülü sağlar.

Tivoli Directory Server v6.3 Client 'ın CEPL kullanabilmesi için kurulması gerekir.

Bu örneği kullanmak için desteklenen altyapılarda IBM MQ yönetimi ile ilgili çalışma bilgisi gereklidir.

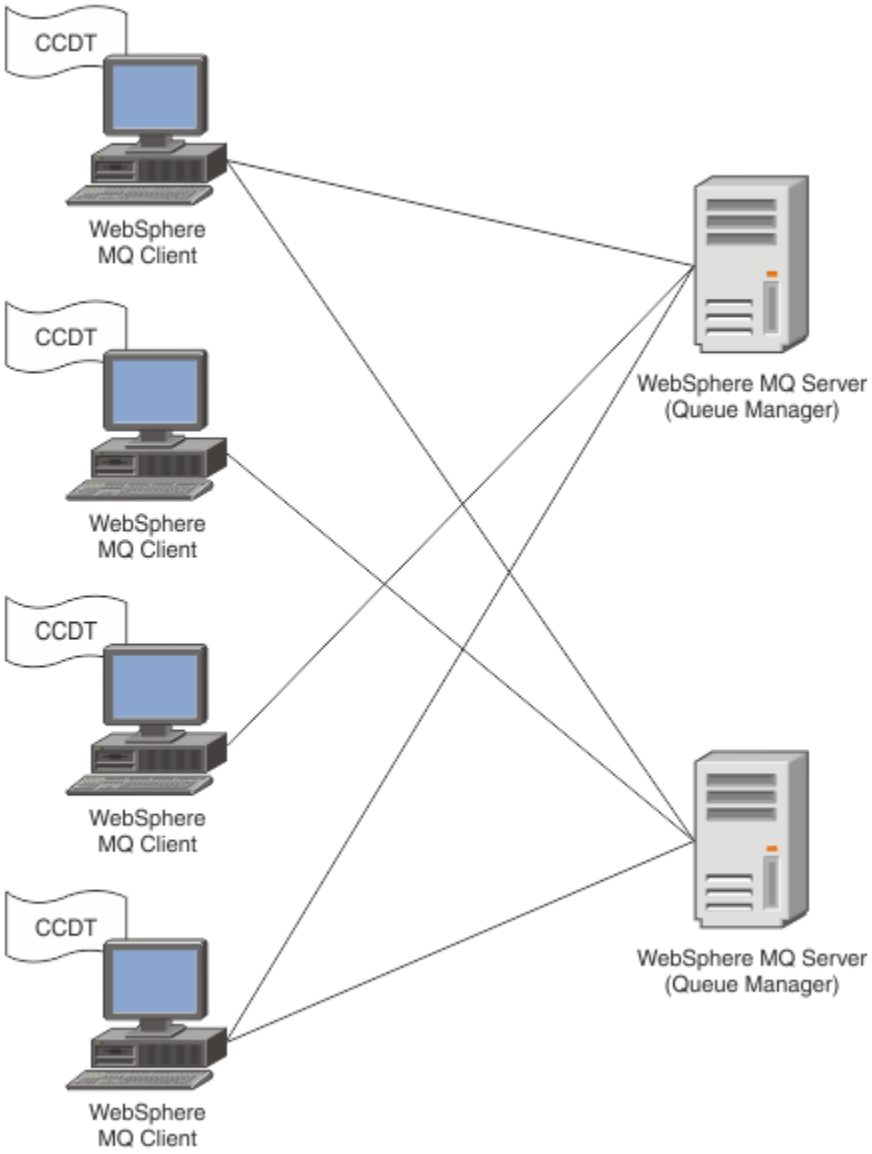
Linux Windows AIX *Giriş*

Bakım ve denetime yardımcı olmak üzere istemci bağlantısı tanımlarını saklamak için bir genel havuz (örneğin, LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) dizini) yapılandırın.

CCDT (Client Connection Definition Table; İstemci Bağlantısı Tanımlama Çizelgesi) aracılığıyla bir kuyruk yöneticisiyle bağlantı kurmak için IBM MQ Client uygulamasını kullanma.

CCDT, standart IBM MQ MQSC Administration arabirimi aracılığıyla yaratılır. Tanımlamada yer alan veriler Kuyruk Yöneticisi ile sınırlı olmasa da, istemci bağlantısı tanımlamaları yaratmak için kullanıcının bir kuyruk yöneticisine bağlanması gerekir. Oluşturulan

CCDT dosyası, istemci makineler ve uygulamalar arasında el ile dağıtılmalıdır.

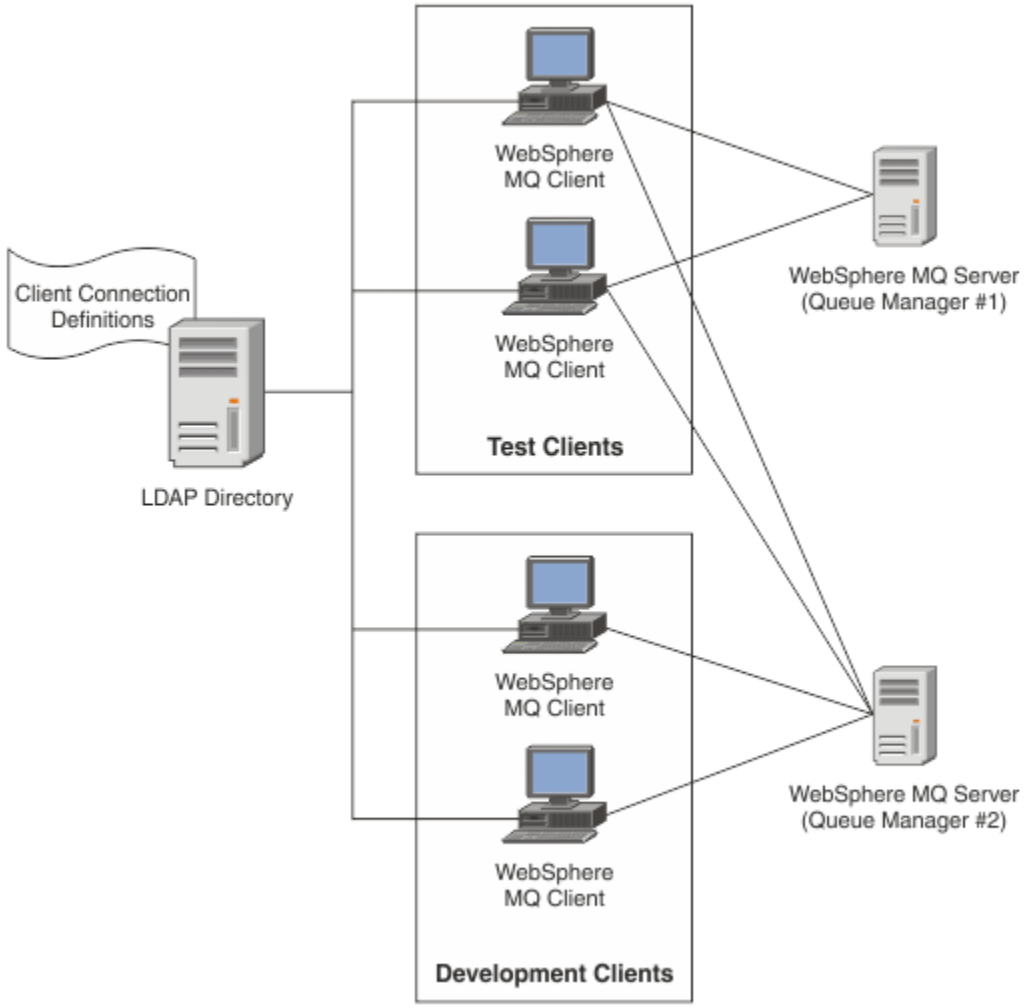


CCDT dosyası her IBM MQ istemcisine dağıtılmalıdır. Binlerce müşterinin yerel ya da küresel olarak var olabildiği yerlerde, kısa süre içinde bakımı ve yönetimi zorlaşacak. Her müşterinin doğru müşteri tanımlarına sahip olmasını sağlamaya yardımcı olmak için daha esnek bir yaklaşım gerekir.

Bu yaklaşımlardan biri, istemci bağlantısı tanımlarının LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) dizini gibi bir genel havuzda saklanmasıdır. LDAP dizini, ek güvenlik, dizin oluşturma ve arama olanakları da sağlayabilir; böylece, her istemcinin yalnızca ilgili bağlantı tanımlarına erişmesine izin verilir.

LDAP dizini, belirli kullanıcı grupları için yalnızca belirli tanımlamaların kullanılabileceği şekilde yapılandırılabilir. Örneğin, Test İstemcileri hem Kuyruk Yöneticisine #1 hem

de #2erişebilirken, Geliştirme İstemcileri yalnızca Kuyruk Yöneticisine #2 erişebilir.



Çıkış modülü, kanal tanımlamalarını almak için bir LDAP havuzu (örneğin, IBM Tivoli Directory Server) arayabilir. Bir IBM MQ istemci uygulaması, bu bağlantı tanımlamalarını kullanarak bir kuyruk yöneticisiyle bağlantı kurabilir.

Çıkış modülü, bir LDAP havuzundan MQCONN/MQCONNX çağrısı sırasında kanal tanımlamasının elde edilmesine olanak sağlayan bir bağlantı öncesi çıkış modülüdür.

Çıkış modülü ve şema şu şekilde gerçekleştirilebilir:

- Mevcut CCDT dosya tabanlı teknolojiyi kullanarak bir beceri tabanı oluşturup yönetim ve dağıtım maliyetlerini azaltmak isteyen müşteriler.
- Müşteri bağlantısı tanımlarının dağıtılması için kendi özel teknolojisini kullanan mevcut müşteriler.
- Şu anda herhangi bir istemci bağlantısı çözümü kullanmayan ve IBM MQtarafından sunulan özellikleri kullanmak isteyen yeni ya da var olan müşteriler.
- İleti sistemi modellerini herhangi bir LDAP iş mimarisiyle doğrudan kullanmak ya da ayarlamak isteyen yeni ya da var olan müşteriler.

ALW *Desteklenen ortamlar*

Connection Endpoint Lookup örneğini çalıştırmadan önce desteklenen bir işletim sisteminiz ve ilgili yazılımınız olduğunu doğrulayın.

IBM MQ Connection Endpoint Lookup için örnek program aşağıdaki yazılımı gerektirir:

- IBM WebSphere MQ 7.0ya da daha sonra

- Tivoli Directory Server V6.3 Client ya da üstü

Desteklenen İşletim Sistemleri:

1. Windows (8/7/2008/2012)
2. AIX
3. Linux
   - System p üzerinde RHEL v4 ve v5
   - System p üzerinde SUSE v9 ve v10
   - RHEL v4 ve v5 x86-64 32 bit ve 64 bit
   - SUSE v9 ve v10 x86-64 32 bit ve 64 bit

Not: Örnek aşağıdaki altyapılar için kullanılamaz:

- z/OS
- IBM i

ALW *Kuruluş ve yapılandırma*
Çıkış modülü ve bağlantı uç noktası şeması kuruluyor ve yapılandırılıyor.

Çıkış modülü kuruluyor

IBM MQkuruluşu sırasında, çıkış modülü `tools/samples/c/preconnexit/bin`altına kurulur. 32 bit altyapılarda, çıkış modülünün kullanılabilmesi için `exit/`*`installation_name`*`/` ' e kopyalanması gerekir. 64 bit altyapılar için, çıkış modülünün kullanılmadan önce exit64/*kuruluş_adı*/dizinine kopyalanması gerekir.

Bağlantı Uç Noktası şeması kuruluyor

Çıkış, `ibm-amq.schema`Bağlantı Uç Noktası şemasını kullanır. Çıkışın kullanılabilmesi için şema dosyasının herhangi bir LDAP sunucusuna aktarılması gerekir. Şema içe aktarıldıktan sonra, özniteliklere ilişkin değerler eklenmelidir.

Aşağıda, Bağlantı Uç Noktası şemasının içe aktarılmasına ilişkin bir örnek verilmiştir. Örnek, IBM Tivoli Directory Server (ITDS) ürününün kullanıldığını varsayar.

- IBM Tivoli Directory Server ürününün çalıştığından emin olun, ardından `ibm-amq.schema` dosyasını ITDS sunucusuna kopyalayın ya da FTP ile kopyalayın.
- ITDS sunucusunda, şemayı ITDS deposuna kurmak için aşağıdaki komutu girin; burada *LDAP kimliği* ve *LDAP parolası* , LDAP sunucusunun kök DN 'si ve parolasıdır:

  `ldapadd -D "`*`LDAP ID`*`" -w "`*`LDAP password`*`" -f ibm-amq.schema`
- Bir komut penceresinde aşağıdaki komutu girin ya da şemaya doğrulamak üzere göz atmak için bir üçüncü taraf aracı kullanın:

```
ldapsearch objectclass=ibm-amqClientConnection
```

Şema dosyasının içe aktarılmasına ilişkin daha fazla ayrıntı için LDAP Sunucusu belgelerinize bakın.

Yapılandırma

İstemci yapılanış kütüğüne PreConnect adlı yeni bir kısım eklenmelidir; örneğin, `mqclient.ini`. PreConnect bölümü aşağıdaki anahtar sözcükleri içerir:

Modül
API çıkış kodunu içeren modülün adı. Bu alan modülün tam yolunu içeriyorsa, olduğu gibi kullanılır. Tersi durumda, IBM MQ kuruluşundaki `exit` ya da `exit64` klasörü aranır.

İşlev
`LdapPreConnect` çıkış kodunu içeren kitaplığa ilişkin işlevsel giriş noktasının adı. İşlev tanımlaması, işletmenizin işlev prototipine uymaktadır.

Uyarı: Gerçek çıkış giriş noktasını belirtirken, işlev deyimindeki tırnak işaretlerini kaldırmanız gerekir.

Veriler
Kanal tanımlamalarını içeren LDAP havuzunun URI 'si.

Aşağıdaki kod parçacığı, `mqclient.ini` dosyasında gerekli olan değişikliklerin bir örneğidir.

```
PreConnect:
Module=amqlcelp
Function="LdapPreconnectExit"
Data=ldap:dap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
Sequence=1
```

ALW *Çıkışa ve şemaya genel bakış*

Sözdizimi ve bir kuyruk yöneticisiyle bağlantı kurmak için kullanılan parametreler.

IBM MQ 9.3 , bir çıkış modülündeki bir giriş noktası için aşağıdaki sözdizimini tanımlar.

```
void MQENTRY MQ_PRECONNECT_EXIT ( PMQNXP pExitParms
                                , PMQCHAR pQMgrName
                                , PPMQCNO ppConnectOpts
                                , PMQLONG pCompCode
                                , PMQLONG pReason)
```

MQCONN/X çağrısının yürütülmesi sırasında IBM MQ C İstemcisi, işlev sözdiziminin somutlamasını içeren çıkış modülünü yükler. Daha sonra kanal tanımlamalarını almak için bir çıkış işlevi çağırır. Alınan kanal tanımlamaları daha sonra bir kuyruk yöneticisiyle bağlantı kurmak için kullanılır.

Parametreler

pExitParms
Tip: PMQNXP giriş/çıkış

PreConnection çıkış parametresi yapısı. Yapı, çıkışı çağıran tarafından ayrılır ve korunur.

```
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 */
};
```

pQMgrAdı
Tip: PMQCHAR giriş/çıkış

Kuyruk yöneticisinin adı. Girişte bu değiştirge, **QMgrName** değiştirgesiyle MQCONN API çağrısına sağlanan süzgeç dizgisidir. Bu alan boş, açık ya da belirli genel arama karakterleri içeriyor olabilir. Alan çıkış tarafından değiştirilir. Çıkış MQXR_TERM ile çağrıldığında değiştirge NULL olur.

ppConnectSeçmeler
Tip: ppConnectOpts giriş/çıkış

MQCONNX işlemini denetleyen seçenekler. Bu, MQCONN API çağrısının işlemini denetleyen bir MQCNO bağlantı seçenekleri yapısına ilişkin bir işaretçidir. Çıkış MQXR_TERM ile çağrıldığında değiştirge NULL olur. MQI istemcisi, başlangıçta uygulama tarafından sağlanmamış olsa da, çıkışa her zaman bir MQCNO yapısı sağlar. Bir uygulama MQCNO yapısı sağlarsa, istemci bunu değiştirildiği çıkışa geçirmek için bir yineleme yapar. İstemci MQCNO ' nun sahipliğini korur. MQCNO ile gönderme yapılan bir MQCD, dizi aracılığıyla sağlanan bağlantı tanımlamasından önceliklidir. İstemci kuyruk yöneticisine bağlanmak için MQCNO yapısını kullanır ve diğerleri yoksayılır.

pCompKodu
Tip: PMQLONG giriş/çıkış

Tamamlanma kodu. Çıkış tamamlama kodunu alan bir MQLONG göstergesi. Aşağıdaki değerlerden biri olmalıdır:

- MQCC_OK -Başarılı tamamlama
- MQCC_WARNING -Uyarı (kısmi tamamlama)
- MQCC_FAILED -Çağrı başarısız oldu

pReason
Tip: PMQLONG giriş/çıkış

Neden niteleyen pCompKodu. Çıkış neden kodunu alan bir MQLONG göstergesi. Tamamlama kodu MQCC_OK ise, geçerli tek değer: MQRC_NONE -(0, x '000') Raporlamak için bir neden yok.

Tamamlanma kodu MQCC_FAILED ya da MQCC_WARNING ise, çıkış işlevi neden kodu alanını geçerli bir MQRC_ * değerine ayarlayabilir.

ALW *MQ LDAP Bağlam Bilgileri*
Çıkış, bağlam bilgileri için aşağıdaki veri yapısını kullanır.

MQNLDAPCTX
MQNLDAPCTX yapısı aşağıdaki C prototipini içeriyor.

```
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 */
};
```

Linux Windows AIX *Bağlantı uç noktası arama çıkışını oluşturmak için örnek kod*
Kaynağı AIX, Linuxya da Windowsüzerinde derlemek için örnek kod parçacıklarını kullanabilirsiniz.

Kaynak derleniyor

Kaynağı herhangi bir LDAP istemcisi kitaplıkları ile derleyebilirsiniz; örneğin, IBM Tivoli Directory Server V6.3 Client kitaplıkları. Bu belge, Tivoli Directory Server V6.3 istemci kitaplıklarını kullandığınızı varsayar.

Not: Bağlantı öncesi çıkış kitaplığı aşağıdaki LDAP sunucularında desteklenir:

- IBM Tivoli Directory Server V6.3

- Novell eDirectory V8.2

Aşağıdaki kod parçacıkları çıkışların nasıl derleneceğini açıklar:

Windows **Windows platformunda çıkışın derlenmesi**

Çıkış kaynağını derlemek için aşağıdaki parçacığı kullanabilirsiniz:

```
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
```

Not: Microsoft Visual Studio 2003 derleyicisiyle derlenen IBM Tivoli Directory Server V6.3 Client kitaplıklarını kullanıyorsanız, IBM Tivoli Directory Server V6.3 Client kitaplıklarını Microsoft Visual Studio 2012ya da daha sonraki bir derleyiciyle derlerken uyarılar alabilir.

Linux AIX **AIX, Linux üzerindeki çıkışın derlenmesi**

Aşağıdaki kod parçacığı, Linuxüzerindeki çıkış kaynağını derlemek içindir. Bazı derleyici seçenekleri AIXüzerinde farklılık gösterebilir.

```
#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 , hem statik hem de dinamik bağlantı kitaplıklarını içerir, ancak tek bir kitaplık tipi kullanabilirsiniz. Bu komut dosyası, statik kitaplıkları kullandığınızı varsayar.

```
#Use static libraries.
LDLIBS=-L$(TDSL) -libibmldapstatic

CFLAGS=-I. -I$(MQMI) -I$(TDSI)

all:amqlcepl

amqlcepl: amqlcel0.c
$(CC) -o cepl amqlcel0.c -shared -fPIC $(XFLAG) $(CFLAGS) $(LDLIBS)
```

ALW *PreConnect çıkış modülünün çağrılması*

PreConnect çıkış modülü üç farklı neden koduyla çağrılabilir: Bir LDAP sunucusuna bağlantı başlatmak ve kurmak için MQXR_INIT neden kodu, bir LDAP sunucusundan kanal tanımlamalarını almak için MQXR_PRECONNECT neden kodu ya da çıkış temizlenirken MQXR_TERM neden kodu.

MQXR_INIT

Çıkış, bir LDAP sunucusuna bağlantı başlatmak ve kurmak için MQXR_INIT neden koduyla çağrılır.

MQXR_INIT çağrısından önce, MQNXP yapısının pExitDataPtr alanı, `mqclient.ini` dosyası (LDAP) içindeki PreConnect kısmı içindeki Veri özniteliğiyle doldurulur.

LDAP URL , en azından arama için protokol, anasistem adı, kapı numarası ve temel DN ' den oluşur. Çıkış, pExitDataPtr alanında bulunan LDAP URL ' yi ayrıştırır, bir MQNLDAPCTX LDAP Arama Bağlamı yapısını ayırır ve buna göre veri yerleştirir. Bu yapının adresi pExitUserAreaPtr alanında saklanır. LDAP URL ' nin doğru şekilde ayrıştırılamaması MQCC_FAILED hatasıyla sonuçlanır.

Bu noktada, çıkış **MQNLDAPCTX** parametrelerini kullanarak LDAP sunucusuna bağlanır ve LDAP sunucusuna bağlanır. Sonuçta ortaya çıkan LDAP API tanıtıcıları da bu yapı içinde saklanır.

MQXR_PRECONNECT
Çıkış modülü, bir LDAP sunucusundan kanal tanımlamalarını almak için MQXR_PRECONNECT neden koduyla çağrılır.

Çıkış, belirtilen süzgeçle eşleşen kanal tanımlamaları için LDAP sunucusunda arama yapar. **QMgrNameparameter** belirli bir kuyruk yöneticisi adını içeriyorsa, arama, **ibm-amqQueueManagerName** LDAP öznitelik değerinin belirtilen kuyruk yöneticisi adıyla eşleştiği tüm kanal tanımlamalarını döndürür.

QMgrName parametresi '*' ya da ' ise ' (boşluk) sonra arama, **ibm-amqIsClientDefault Connection** uç noktası özniteliğinin TRUEolarak ayarlandığı tüm kanal tanımlamalarını döndürür.

Başarılı bir aramadan sonra, çıkış bir ya da daha çok MQCD tanımlaması hazırlar ve çağırana geri döner.

MQXR_TERM
Çıkış, temizlenmek üzere olduğunda bu neden koduyla çağrılır. Bu temizleme sırasında, çıkış LDAP sunucusuyla bağlantısını keser ve MQNLDAPCTX yapısı, gösterge dizisi ve başvurduğu her MQCD de içinde olmak üzere çıkış tarafından ayrılan ve bakımı yapılan tüm belleği serbest bırakır. Diğer alanlar varsayılan değerlere ayarlanır. **pQMgrName** ve **ppConnectOpts** çıkış değiştirgeleri MQXR_TERM neden koduyla bir çıkış sırasında kullanılmaz ve NULLolabilir.

İlgili başvurular
İstemci yapılanış kütüğünün PreConnect kısmı

ALW *LDAP şemaları*
İstemci bağlantı verileri, LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) dizini adı verilen bir genel havuzda saklanır. IBM MQ istemcisi, bağlantı tanımlarını almak için bir LDAP dizini kullanır. LDAP dizini içindeki IBM MQ istemci bağlantısı tanımlarının yapısı LDAP şeması olarak bilinir. LDAP şeması, bir süzgeç ya da öznitelik değeri değerlendirmesinin bir girişin öznitelikleriyle eşleşip eşleşmediğini ve işlemlere izin verilip verilmeyeceğini, işlemlerin eklenip eklenmeyeceğini ve değiştirilip değiştirilmeyeceğini belirlemek için sunucunun kullandığı öznitelik tipi tanımlamaları, nesne sınıfı tanımlamaları ve diğer bilgiler toplamalarıdır.

Verilerin LDAP dizininde saklanması

İstemci bağlantısı tanımlamaları, bağlantı noktası olarak bilinen dizin ağacındaki belirli bir dal altında bulunur. Bir LDAP dizinindeki diğer tüm düğümler gibi, bağlantı noktasının da ilişkili bir Ayırt Edici Adı (DN) vardır. Bu düğümü, dizinde oluşturacağınız sorgular için başlangıç noktası olarak kullanabilirsiniz. İstemci bağlantısı tanımlamalarının bir altkümesini döndürmek için LDAP dizinini sorgularken süzgeç uygulamayı kullanın. Dizin ağacının diğer bölümlerinde verilen izinlere (örneğin, kullanıcılara, bölümlere ya da gruplara) dayalı olarak alt ağaçlara erişimi kısıtlayabilirsiniz.

Kendi özniteliklerinizi ve sınıflarınızı tanımlama
LDAP şemasını değiştirerek istemci kanalı tanımlamasını saklayın. Tüm LDAP veri tanımlamaları için nesneler ve öznitelikler gerekir. Nesneler ve öznitelikler, nesneyi ya da özniteliği benzersiz olarak tanımlayan bir nesne tanıtıcısı (OID) numarasıyla tanımlanır. Bir LDAP şemasındaki tüm sınıflar, doğrudan ya da dolaylı olarak üst nesneden devralır. İstemci kanal tanımlaması nesnesi, en üstteki nesnenin özniteliklerini içerir. Tüm LDAP veri tanımlamaları için nesneler ve öznitelikler gerekir:

- Nesne tanımlamaları, LDAP özniteliklerinin toplamalarıdır.
- Öznitelikler, LDAP veri tipleridir.

Her özniteliğin tanımı ve bunların normal IBM MQ özellikleriyle nasıl eşlendikleri LDAP özniteliklerііçinde açıklanır.

ALW *LDAP öznitelikleri*

Tanımlanan LDAP öznitelikleri IBM MQ ' e özgüdür ve doğrudan istemci bağlantısı özellikleriyle eşlenir.

IBM MQ İstemci Kanal Dizini Dizgi Öznitelikleri

IBM MQ özellikleriyle eşlenen karakter dizgisi öznitelikleri aşağıdaki çizelgede listelenir. Öznitelikler, directoryString (UTF-8 kodlanmış Unicode; yani, altküme olarak IA5/ASCII içeren değişken byte kodlama sistemi) sözdizimini içerebilir. Sözdizimi, nesne tanıtıcı numarasıyla (OID) belirtilir.

Çizelge 166. IBM MQ istemci kanal dizini dizgi öznitelikleri

LDAP Özniteliği	Açıklama	IBM MQ Özellik
CN	Kanal adı ve tanımlayan kuyruk yöneticisi adından oluşan ortak ad.	
ibm-amqChannelAdı	Kanal tanımının adı.	Kanal
ibm-amqConnectionAdı	İletişim bağlantısı tanıtıcısı.	KONADı
ibm-amqDescription	Kanal açıklaması.	TANIMLAMA
ibm-amqLocalAdresi	Kanalın yerel iletişim adresi.	LOCLADDR
ibm-amqModeAdı	LU 6.2 kip adı.	MODENAME (KIPI)
ibm-amqPassword	Kullanılabilecek parola.	Parola
ibm-amqQueueManagerName	Bir IBM MQ istemci uygulamasının bağlantı isteyebileceği kuyruk yöneticisinin ya da kuyruk yöneticisi grubunun adı.	QMNAME
ibm-amqSecurityExitUserData	Güvenlik çıkışına geçirilen kullanıcı verileri.	SCYDATA
ibm-amqSecurityExitName	Kanal güvenlik çıkışı tarafından çalıştırılacak çıkış programının adı.	SCYEXIT
ibm-amqSslCipherSpec	TLS bağlantısı için tek bir CipherSpec .	SSLCIPH
ibm-amqSslPeerName	IBM MQ kanalının diğer ucunda eşdüzey kuyruk yöneticisinden ya da istemciden alınan sertifikanın Ayırt Edici Adını (DN) denetler.	SSLPEER
ibm-amqTransactionProgramName	Hareket işleme programı adı.	TPNAME
ibm-amqUserkimliği	Uzak bir MCA ile güvenli SNA oturumu başlatma girişimi sırasında MCA tarafından kullanılacak kullanıcı kimliği.	USERID

IBM MQ istemci bağlantısı tamsayı öznitelikleri

Önceden tanımlanmış değerleri olan öznitelikler (örneğin, sıralı değer tipi) standart tamsayılar olarak saklanır. Bu değerler, ilişkili değişmez adı kullanılarak değil, LDAP dizininde tamsayı değerleri olarak saklanır.

Çizelge 167. IBM MQ istemci kanal dizini tamsayı öznitelikleri

LDAP Özniteliği	Açıklama	IBM MQ Özellik
ibm-amqConnectionYakınlığı	Aynı kuyruk yöneticisi adıyla birden çok kez bağlanan istemci uygulamalarının aynı istemci kanalını kullanıp kullanmayacağını belirler.	SONRAKALıK

Çizelge 167. IBM MQ istemci kanal dizini tamsayı öznitelikleri (devamı var)		
LDAP Özniteliği	**Açıklama**	**IBM MQ Özellik**
ibm-amqClientChannelWeight	Hangi istemci bağlantı kanalı tanımının kullanıldığını etkileyen bir ağırlıklandırma.	KLNTWGHT
ibm-amqHeartBeatInterval	İletim kuyruğunda ileti olmadığında gönderen MCA ' dan geçirilecek sağlıklı işletim bildirimi akışları arasındaki yaklaşık süre.	HBINT (HBINT
ibm-amqKeepAliveInterval	Bir kanalın zaman çıkış değeri.	KAINT
ibm-amqMaximumMessageLength	Kanalda iletilebilecek ileti uzunluğu üst sınırı.	MAXMSGL
ibm-amqSharingEtkileşimler	Her bir TCP/IP kanal örneğini paylaşan etkileşim sayısı üst sınırı.	SHARECNV
ibm-amqTransportTipi	Kullanılacak iletim tipi.	İZLEMA TIPI

IBM MQ istemci kanalı Boole özniteliği

Bu Boole özniteliği herhangi bir IBM MQ özelliğiyle eşlenmedi. Bu özniteliğin sözdizimi bir Boole değerini gösterir.

Çizelge 168. IBM MQ istemci kanalı Boole özniteliği	
LDAP Özniteliği	**Açıklama**
ibm-amqIsClientDefault	Bu Boole özniteliği, ibm-amqQueueManagerName özniteliği tanımlanmamış olan girişleri arama sorununu çözmek için tanımlanır.

IBM MQ istemci kanalı listesi öznitelikleri

IBM MQ özellikleri, LDAP dizini içinde tek değerli, virgülle ayrılmış liste özniteliği olarak saklanır. Öznitelikler, diğer dizin dizgisi öznitelikleriyle aynı şekilde tanımlanır. IBM MQ özellikleriyle eşlemeleriyle birlikte liste öznitelikleri aşağıdaki çizelgede açıklanmıştır.

Çizelge 169. IBM MQ istemci kanalı listesi öznitelikleri		
LDAP Özniteliği	**Açıklama**	**IBM MQ Özellik**
ibm-amqHeaderSıkıştırma	Kanal tarafından desteklenen üstbilgi veri sıkıştırma tekniklerinin bir listesi.	TAMAMLANDI
ibm-amqMessageSıkıştırma	Kanal tarafından desteklenen ileti veri sıkıştırma tekniklerinin listesi.	ŞIRKET
ibm-amqSendExitUserData	Gönderme çıkışına geçirilen kullanıcı verileri.	SENDDATA
ibm-amqSendExitUserName	Kanal gönderme çıkışı tarafından çalıştırılacak çıkış programının adı.	SENDEXIT
ibm-amqReceiveExitUserVerileri	Alma çıkışına geçirilen kullanıcı verileri.	RCVDATA
ibm-amqReceiveExitName	Kanal alma kullanıcı çıkışı tarafından çalıştırılacak kullanıcı çıkış programının adı.	RCVEXIT

ALW *Ortak Ad*

Ortak ad (CN), kanal adı ve tanımlayan kuyruk yöneticisi adından oluşur.

Önceden var olan bir özniteliktir.

CN biçimi şöyledir:

```
CN=CHANNEL_NAME(DEFINING_Q_MGR_NAME)
```

Örneğin:

```
CN=TC1(QM_T1)
```

Bu öznitelik için tek bir değer belirtebilirsiniz.

Bu öznitelik bir dizgi özniteliğidir ve değerler büyük ve küçük harfe duyarlı değildir. Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten alt şemada kullanılan ve bir alt dizgiyi kullanan (örneğin, CN=jim * burada CN bir özniteliktir) ve bir ya da daha çok joker karakter içeren bir kuraldır.

ALW *ibm-amqChannelAdı*
Bu öznitelik, kanal tanımlamasının adını belirtir.

Bu öznitelik, büyük/küçük harfe duyarlı olmayan en çok 20 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecinde özniteliğin davranışını belirten ve bir ya da daha fazla joker karakter içeren alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqDescription*
Bu LDAP özniteliği kanal açıklamasını sağlar.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 64 byte içeren tek bir dizgi değerine sahiptir. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqConnectionAdı*
Bu LDAP özniteliği, iletişim bağlantısı tanıtıcısıdır. Bu kanal tarafından kullanılacak iletişim bağlantılarını belirler.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 264 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqLocalAdresi*
Bu öznitelik, kanalın yerel iletişim adresini belirtir.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 48 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqModeAdı*
Bu öznitelik, LU 6.2 bağlantılarıyla kullanılmak içindir. Bir iletişim oturumu ayırma işlemi gerçekleştirildiğinde, bağlantının oturum özellikleri için ek tanımlama sağlar.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan tam olarak 8 karakterden oluşan tek bir dizgi değerine sahiptir. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqPassword*
Bu LDAP özniteliği, uzak bir MCA ile güvenli bir LU 6.2 oturumu başlatma girişimi sırasında MCA tarafından kullanılabilecek bir parolayı belirtir.

Bu özniteliğin en çok 12 basamağı olan tek bir tamsayı değeri vardır. Önceden var olan bir öznitelik değil.

ALW *ibm-amqQueueManagerName*
Bu öznitelik, bir IBM MQ istemci uygulamasının bağlantı isteyebileceği kuyruk yöneticisinin ya da kuyruk yöneticisi grubunun adını belirtir.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 48 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

İlgili başvurular
“ibm-amqIsClientDefault” sayfa 1109
Bu Boole özniteliği, ibm-amqQueueManagerName özniteliğinin tanımlanmadığı girdilerin aranması sorununu çözer.

ALW *ibm-amqSecurityExitUserVerileri*
Bu LDAP özniteliği, güvenlik çıkışına geçirilen kullanıcı verilerini belirtir.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 999 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqSecurityExitName*
Bu LDAP özniteliği, kanal güvenlik çıkışı tarafından çalıştırılacak çıkış programının adını belirtir.

Geçerli bir kanal güvenlik çıkışı yoksa boş bırakın.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 999 karakter içeren tek bir dizgi değeri içeriyor. Bu öznitelik, çıkış öncesi bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqSslCipherSpec*
Bu LDAP özniteliği, TLS bağlantısı için tek bir CipherSpec belirtir.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 32 karakterden oluşan tek bir dizgi değerine sahiptir. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqSslPeerName*
Bu LDAP özniteliği, IBM MQ kanalının diğer ucunda eşdüzey kuyruk yöneticisinden ya da istemciden sertifikanın Ayırt Edici Adını (DN) denetlemek için kullanılır.

Bu LDAP özniteliği, büyük ve küçük harfe duyarlı olmayan en çok 1024 bayta sahip tek bir dizgi değerine sahiptir. Önceden var olan bir şey değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqTransactionProgramName*
Bu LDAP özniteliği, hareket programı adını belirtir. Bu, LU 6.2 bağlantılarıyla kullanılmak içindir.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 64 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir şey değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqUserkimliği*
Bu LDAP özniteliği, uzak bir MCA ile güvenli SNA oturumu başlatma girişimi sırasında MCA tarafından kullanılacak kullanıcı kimliğini belirtir.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan tam olarak 12 karakterden oluşan tek bir dizgi değerine sahiptir. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

ALW *ibm-amqConnectionYakınlığı*
Bu LDAP özniteliği, aynı kuyruk yöneticisi adını kullanarak birden çok kez bağlanan istemci uygulamalarının aynı istemci kanalını kullanıp kullanmayacağını belirtir.

Bu özniteliğin tek bir tamsayı değeri var. Önceden var olan bir öznitelik değil.

ALW *ibm-amqClientChannelWeight*
Bu LDAP özniteliği, hangi istemci bağlantı kanalı tanımlamasının kullanıldığını etkileyen bir ağırlıklandırmayı belirtir.

İstemci kanalı ağırlıklandırma özniteliği, birden çok uygun tanımlama varsa, istemci kanalı tanımlarının seçimini sapmak için kullanılır.

Bu özniteliğin tek bir tamsayı değeri var. Önceden var olan bir öznitelik değil.

ALW *ibm-amqHeartBeatInterval (BeatInterval)*
Bu LDAP özniteliği, iletim kuyruğunda ileti olmadığında gönderen MCA ' dan geçirilecek sağlıklı işletim bildirimi akışları arasındaki yaklaşık süreyi belirtir.

Bu özniteliğin tek bir tamsayı değeri var. Önceden var olan bir öznitelik değil. Varsayılan değer 1'dir. Varsayılan değer, yürürlükteki MQSERVER ortam değişkeni işleminde ayarlanır.

ALW *ibm-amqKeepAliveInterval*
Bu LDAP özniteliği, bir kanal için zaman çıkış değeri belirtmek için kullanılır.

Bu özniteliğin değeri, kanala ilişkin canlı tutma zamanlamasını belirten iletişim yığınına geçirilir. Her kanal için farklı bir canlı tutma değeri belirtmek için bu değeri kullanabilirsiniz.

Bu özniteliğin tek bir tamsayı değeri var. Önceden var olan bir öznitelik değil.

ALW *ibm-amqMaximumMessageLength*
Bu LDAP özniteliği, kanalda iletilebilecek bir iletinin uzunluk üst sınırını belirtir.

Bu özniteliğin varsayılan değeri, yürürlükteki MQSERVER ortam değişkeni işlemine göre 104857600 'dür. Bu özniteliğin tek bir tamsayı değeri var ve önceden var olan bir öznitelik değil.

ALW *ibm-amqSharingEtkileşimler*
Bu LDAP özniteliği, her TCP/IP kanal örneğini paylaşan etkileşim sayısı üst sınırını belirtir.

Bu özniteliğin tek bir tamsayı değeri var. Bu öznitelik önceden var olan bir öznitelik değil.

ALW *ibm-amqTransportTipi*
Bu LDAP özniteliği, kullanılacak iletim tipini belirtir.

Bu özniteliğin tek bir tamsayı değeri var. Önceden var olan bir öznitelik değil.

ALW *ibm-amqIsClientDefault*

Bu Boole özniteliği, ibm-amqQueueManagerName özniteliğinin tanımlanmadığı girdilerin aranması sorununu çözer.

Ön bağlantı çıkış modülleri genellikle, arama ölçütü olarak ibm-amqQueueManagerName özniteliğinin değeriyle LDAP sunucularında arama yapın. Böyle bir sorgu, ibm-amqQueueManagerName öznitelik değerinin MQCONN/X çağrısında belirtilen kuyruk yöneticisinin adıyla eşleştiği tüm girişleri döndürür. Ancak, istemci kanal tanımlama çizelgelerini (CCDT) kullanırken, MQCONN/X çağrısındaki kuyruk yöneticisi adını boşluk olarak ayarlayabilir ya da adın başına yıldız işareti (*) koyabilirsiniz. Kuyruk yöneticisinin adı boşsa, istemci varsayılan kuyruk yöneticisine bağlanır. Kuyruk yöneticisine adın başına yıldız işareti (*) eklenirse, istemci herhangi bir kuyruk yöneticisini bağlar.

Benzer şekilde, bir girişteki ibm-amqQueueManagerName özniteliği de tanımsız bırakılabilir. Bu durumda, bu uç nokta bilgilerini kullanan istemcinin herhangi bir kuyruk yöneticisine bağlanması beklenir. Örneğin, bir girdi aşağıdaki satırları içerir:

```
ibm-amqChannelName = "CHANNEL1"
ibm-amqConnectionName = myhost(1414)
```

Bu örnekte, istemci `myhost`üzerinde çalışan belirtilen kuyruk yöneticisine bağlanmayı dener.

Ancak LDAP Sunucularında, tanımlanmamış bir öznitelik değerinde arama yapılmaz. Örneğin, bir girdi ibm-amqQueueManagerNamedışındaki bağlantı bilgilerini içeriyorsa, arama sonuçları bu girdiyi içermez. Bu sorunu çözmek için ibm-amqIsClientDefaulttayarını yapabilirsiniz. Bu bir Boole özniteliğidir ve tanımlanmamışsa FALSE değerine sahip olduğu varsayılır.

ibm-amqQueueManagerName ' in tanımlanmadığı ve aramanın bir parçası olması beklendiği girdiler için ibm-amqIsClientDefault değerini TRUE olarak ayarlayın. MQCONN/X çağrısında kuyruk yöneticisi adı olarak bir boşluk ya da yıldız işareti (*) belirtildiğinde, ön bağlantı çıkışı, LDAP sunucusunda IBM-amqIsClientDefault öznitelik değerinin TRUE olarak ayarlandığı tüm girişler için arama yapar.

Not: ibm-amqIsClientDefault TRUE olarak ayarlandıysa, ibm-amqQueueManagerName özniteliğini ayarlamayın ya da tanımlamayın.

İlgili başvurular

“ibm-amqQueueManagerName” sayfa 1107

Bu öznitelik, bir IBM MQ istemci uygulamasının bağlantı isteyebileceği kuyruk yöneticisinin ya da kuyruk yöneticisi grubunun adını belirtir.

ALW *ibm-amqHeaderSıkıştırma*

Bu LDAP özniteliği, kanal tarafından desteklenen üstbilgi veri sıkıştırma tekniklerinin bir listesidir.

Bu özniteliğin büyüklük üst sınırı 48 karakterdir. Önceden var olan bir öznitelik değil.

Bu öznitelik için tek bir değer belirtebilirsiniz.

Bu liste özniteliği, virgülle ayrılmış biçim kullanılarak dizin dizgileri olarak belirtilir. Örneğin, **ibm-amqHeaderCompression** için belirtilen değer, `NONE`ile eşlenen `0` değeridir. İzin verilen üst sınırı aşan değerler istemci tarafından yoksayılır. Örneğin, ibm-amqHeaderCompression, listede en çok 2 tamsayı içerir.

ALW *ibm-amqMessageSıkıştırma*

Bu LDAP özniteliği, kanal tarafından desteklenen ileti verileri sıkıştırma tekniklerinin bir listesidir.

Bu özniteliğin büyüklük üst sınırı 48 karakterdir. Önceden var olan bir öznitelik değil.

Bu öznitelik birden çok değeri desteklemiyor.

Bu liste özniteliği, virgülle ayrılmış biçim kullanılarak dizin dizgileri olarak belirtilir. Örneğin, bu öznitelik için belirtilen değer 1,2,4 olup bu, temel sıkıştırma sırası RLE, ZLIBFAST ve ZLIBHIGH ile eşlenir.

İzin verilen üst sınırı aşan değerler istemci tarafından yoksayılır. Örneğin, ibm-amqMessageSıkıştırma, listede en çok 16 tamsayı içerir.

ALW *ibm-amqSendExitUserVerileri*
Bu LDAP özniteliği, gönderme çıkışına geçirilen kullanıcı verilerini belirtir.

Bu LDAP özniteliği, büyük ve küçük harfe duyarlı olmayan en çok 999 karakter içeren tek bir dizgi değerine sahiptir. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

Not: ibm-amqSendExitName ve **ibm-amqSendExitUserData** çiftlerde eşitlenmeli. Kullanıcı verileri çıkış adıyla eşitlenmelidir. Bu nedenle, biri belirtilirse, veri içermese bile, diğeri de simetrik olarak belirtilmesi gerekir.

ALW *ibm-amqSendExitName*
Bu LDAP özniteliği, kanal gönderme çıkışı tarafından çalıştırılacak çıkış programının adını belirtir.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 999 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

Not: ibm-amqSendExitName ve **ibm-amqSendExitUserData** , çiftler halinde eşitlenmelidir. Kullanıcı verileri çıkış adıyla eşitlenmelidir. Bu nedenle, biri belirtilirse, diğeri veri içermese de simetrik olarak belirtilmesi gerekir.

ALW *ibm-amqReceiveExitUserVerileri*
Bu LDAP özniteliği, alma çıkışına geçirilen kullanıcı verilerini belirtir.

Bir alma çıkışları dizisi çalıştırabilirsiniz. Bir çıkış dizisine ilişkin kullanıcı verileri dizgisi virgülle, boşlukla ya da her ikisiyle ayrılır.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 999 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

Not: ibm-amqReceiveExitName ve **ibm-amqReceiveExitUserData** , çiftler halinde eşitlenmelidir. Kullanıcı verileri çıkış adıyla eşitlenmelidir. Bu nedenle, biri belirtilirse, diğeri veri içermese de simetrik olarak belirtilmesi gerekir.

ALW *ibm-amqReceiveExitName*
Bu LDAP özniteliği, kanal alma kullanıcı çıkışı tarafından çalıştırılacak kullanıcı çıkış programının adını belirtir.

Bu öznitelik, art arda çalıştırılacak programların adlarının listesidir. Geçerli bir kanal alma kullanıcı çıkışı yoksa boş bırakın.

Bu öznitelik, büyük ve küçük harfe duyarlı olmayan en çok 999 karakter içeren tek bir dizgi değeri içeriyor. Önceden var olan bir öznitelik değil.

Alt dizgi eşleştirmesi yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirten, alt şemada kullanılan bir eşleştirme kuralıdır.

Not: ibm-amqReceiveExitName ve **ibm-amqReceiveExitUserData** , çiftler halinde eşitlenmelidir. Kullanıcı verileri çıkış adıyla eşitlenmelidir. Bu nedenle, biri belirtilirse, veri içermese bile, diğeri simetrik olarak da belirtilmelidir.

z/OS z/OS için örnek programların kullanılması

IBM MQ for z/OS ile sağlanan örnek yordam uygulamaları, İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir.

Bu görev hakkında

IBM MQ for z/OS , “Veri dönüştürme çıkışları yazılıyor” sayfa 941içinde açıklanan örnek veri dönüştürme çıkışlarını da sağlar.

Tüm örnek uygulamalar kaynak biçimde sağlanır; birkaçı da yürütülebilir biçimde sağlanır. Kaynak modüller, program mantığını açıklayan sözde kod içerir.

Not: Bazı örnek uygulamalarda temel panel tabanlı arabirimler bulunmasına rağmen, uygulamalarınızın görünüşünün ve görünüşünün nasıl tasarlanması gerektiğini göstermeyi amaçlayamazlar. Programlanmayan uçbirimler için pano odaklı arabirimlerin nasıl tasarlanacağına ilişkin daha fazla bilgi için *SAA Common User Access: Basic Interface Design Guide* (SC26-4583) adlı belgeye ve onun ekine (GG22-9508) bakın. Bunlar, hem uygulama içinde hem de diğer uygulamalarda tutarlı uygulamalar tasarlamanıza yardımcı olacak yönergeler sağlar.

Yordam

- Örnek programlar hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:
  - “z/OS için örnek uygulamalarda gösterilen özellikler” sayfa 1111
  - “z/OS üzerinde toplu iş ortamı için örnek uygulamaların hazırlanması ve çalıştırılması” sayfa 1118
  - “z/OS üzerinde TSO ortamı için örnek uygulamaların hazırlanması” sayfa 1121
  - “z/OS üzerinde CICS ortamı için örnek uygulamaların hazırlanması” sayfa 1122
  - “z/OS üzerindeki IMS ortamı için örnek uygulamanın hazırlanması” sayfa 1126
  - “z/OS üzerindeki Put örnekleri” sayfa 1127
  - “z/OS üzerindeki Get örnekleri” sayfa 1129
  - “z/OS üzerindeki Göz At örneği” sayfa 1131
  - “z/OS üzerinde İleti Yazdır örneği” sayfa 1133
  - “z/OS üzerinde Kuyruk Öznitelikleri örneği” sayfa 1137
  - “z/OS üzerindeki Mail Manager örneği” sayfa 1138
  - “z/OS üzerindeki Kredi Denetimi örneği” sayfa 1145
  - “z/OS üzerinde İleti İşleyici örneği” sayfa 1155
  - “z/OS üzerinde Zamanuyumsuz Put örneği” sayfa 1158
  - “z/OS üzerinde Toplu Zamanuyumsuz Tüketim örneği” sayfa 1159
  - “CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örneği z/OS” sayfa 1161
  - “z/OS üzerinde Yayınla/Abone ol örneği” sayfa 1163
  - “z/OS üzerindeki Set and Inquire message özelliği örneği” sayfa 1165

İlgili görevler

“Multiplatforms üzerinde örnek programların kullanılması” sayfa 1014
Bu örnek yordam programları ürünle birlikte teslim edilir. Örnekler C ve COBOL dilinde yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımlarını gösterir.

z/OS *z/OS için örnek uygulamalarda gösterilen özellikler*

Bu kısım, örnek uygulamaların her birinde gösterilen MQI özelliklerini özetler, her bir örneğin yazıldığı programlama dillerini ve her bir örneğin çalıştırıldığı ortamı gösterir.

z/OS *z/OS üzerine örnek koyma*

Put örnekleri, iletilerin MQPUT çağrısı kullanılarak bir kuyruğa nasıl yerleştirileceğini gösterir.

Uygulama şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQPUT
- MQCLOSE
- MQDISC

Program COBOL ve C dillerinde sunulur ve toplu iş ortamında ve CICS ortamında çalışır. Toplu iş uygulaması için bkz. Çizelge 172 sayfa 1119 ve CICS uygulaması için Çizelge 179 sayfa 1123 .

z/OS *z/OS ' dan örnek alın*

Get örnekleri, MQGET çağrısı kullanılarak bir kuyruktan nasıl ileti alınacağını gösterir.

Uygulama şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQGet
- MQCLOSE
- MQDISC

Program COBOL ve C dillerinde sunulur ve toplu iş ortamında ve CICS ortamında çalışır. Toplu iş uygulaması için bkz. Çizelge 172 sayfa 1119 ve CICS uygulaması için Çizelge 179 sayfa 1123 .

z/OS *z/OS üzerindeki örneğe göz atın*

Göz At örneği, bir iletiyi bulmak, yazdırmak ve kuyruktaki iletilerde adım adım ilerlemek için Göz At seçeneğinin nasıl kullanılacağını gösterir.

Uygulama şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- İletilere göz atmak için MQGET
- MQCLOSE
- MQDISC

Program, COBOL, çevirici, PL/I ve C dillerinde sağlanır. Uygulama toplu iş ortamında çalışır. Toplu iş uygulaması için bkz. Çizelge 173 sayfa 1119 .

z/OS *İleti örneğini z/OS üzerinde yazdır*

İleti Yazdır örneği, ileti tanımlayıcısının tüm alanlarıyla birlikte, bir iletinin kuyruktan nasıl kaldırılacağını ve iletideki verilerin nasıl yazdırılacağını gösterir. İsteğe bağlı olarak, her iletiyle ilişkili tüm ileti özelliklerini görüntüleyebilir.

Kaynak modüldeki iki satırdan açıklama karakterlerini kaldırarak, programı, kuyruktaki iletilere göz atmak yerine göz atması için değiştirebilirsiniz. Bu program, iletileri kuyruğa koyan bir uygulamayla ilgili sorunları tanılamak için kullanılabilir.

Uygulama şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- Kuyruktan ileti kaldırmak için MQGET (göz atma seçeneğiyle)

- MQCLOSE
- MQDISC
- MQCRTMH
- MQDLTMH
- MQINQMP

Program C dilinde teslim edilir. Uygulama toplu iş ortamında çalışır. Toplu iş uygulaması için bkz. Çizelge 174 sayfa 1119 .

z/OS *z/OS üzerinde Kuyruk Öznitelikleri örneği*

Kuyruk Öznitelikleri örneği, IBM MQ for z/OS nesne özniteliklerinin değerlerinin nasıl sorgulacağını ve ayarlanacağını gösterir.

Uygulama şu MQI çağrılarını kullanır:

- MQOPEN
- MQINQ
- MQSET
- MQCLOSE

Program, COBOL, çevirici ve C dillerinde sağlanır. Uygulama CICS ortamında çalışır. CICS uygulaması için bkz. Çizelge 180 sayfa 1124 .

z/OS *z/OS üzerinde Mail Manager örneği*

Mail Manager örneği kullanılırken dikkat edilmesi gereken noktalar.

Mail Manager örneği aşağıdaki teknikleri gösterir:

- Diğer ad kuyruklarının kullanılması
- Geçici dinamik kuyruk yaratmak için model kuyruğunun kullanılması
- Yanıt kuyruklarının kullanılması
- CICS ve toplu iş ortamlarında eşitleme noktalarını kullanma
- Sistem komutu giriş kuyruğuna komut gönderilmesi
- Dönüş kodlarının sınanması
- Uzak kuyruk yöneticilerine, uzak kuyruğun yerel tanımlamasını kullanarak ve iletileri uzak kuyruk yöneticisindeki adlandırılmış bir kuyruğa doğrudan yerleştirerek ileti gönderilmesi

Uygulama şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQPUT1
- MQGet
- MQINQ
- MQCMIT
- MQCLOSE
- MQDISC

Uygulamanın üç sürümü sağlanır:

- COBOL dilinde yazılmış bir CICS uygulaması
- COBOL dilinde yazılmış bir TSO uygulaması
- C dilinde yazılmış bir TSO uygulaması

TSO uygulamaları IBM MQ for z/OS toplu iş bağdaştırıcısını kullanır ve bazı ISPF panolarını içerir.

TSO uygulaması için bkz. Çizelge 177 sayfa 1121 ve CICS uygulaması için Çizelge 181 sayfa 1124 .

z/OS *z/OS üzerinde Kredi Denetimi örneği*

Bu bilgiler, Kredi Denetimi örneğini kullanırken göz önünde bulundurulması gereken noktaları içerir.

Credit Check örneği, aşağıdaki teknikleri gösteren bir program grubudur:

- Birden çok ortamda çalışan bir uygulama geliştirilmesi
- Geçici dinamik kuyruk yaratmak için model kuyruğunun kullanılması
- İlinti tanıtıcısı kullanılması
- Bağlam bilgilerini ayarlama ve aktarma
- İleti önceliğini ve sürekliliğini kullanma
- Tetikleme kullanarak programları başlatma
- Yanıt kuyruklarının kullanılması
- Diğer ad kuyruklarının kullanılması
- Gönderilmeyen iletiler kuyruğunun kullanılması
- Ad listesi kullanılması
- Dönüş kodlarının sınanması

Uygulama şu MQI çağrılarını kullanır:

- MQOPEN
- MQPUT
- MQPUT1
- İletilere göz atmak ve ileti almak, bekleme ve sinyal seçeneklerini kullanmak ve belirli bir iletiyi almak için MQGET
- MQINQ
- MQSET
- MQCLOSE

Örnek, bağımsız bir CICS uygulaması olarak çalışabilir. Ancak, hem CICS hem de IMS ortamları tarafından sağlanan olanakları kullanan bir ileti kuyruklama uygulamasının nasıl tasarlanılacağını göstermek için, bir modül IMS toplu ileti işleme programı olarak da sağlanır.

CICS programları C ve COBOL dilinde sağlanır. Tek IMS programı C ile teslim edilir.

CICS uygulaması için bkz. Çizelge 182 sayfa 1125 ve IMS uygulaması için Çizelge 184 sayfa 1126 .

z/OS *z/OS üzerinde İleti İşleyici örneği*

İleti İşleyici örneği, bir kuyruktaki iletilere göz atmanızı, iletileri iletmenizi ve silmenizi sağlar.

Uygulama şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQINQ
- MQPUT1
- MQCMIT
- MQBACK
- MQGet
- MQCLOSE
- MQDISC

Program, C ve COBOL programlama dillerinde sağlanır. Uygulama TSO altında çalışır. TSO uygulaması için bkz.Çizelge 178 sayfa 1122.

z/OS *z/OS üzerinde dağıtılmış kuyruğa alma çıkış örnekleri*

Dağıtımlı kuyruğa alma çıkış örneklerinin kaynak programlarını içeren bir çizelge.

Dağıtılmış kuyruğa alma çıkış örneklerinin kaynak programlarının adları aşağıdaki çizelgede listelenir:

Çizelge 170. Dağıtılmış kuyruğa alma çıkış örneklerinin kaynağı

Üye adı	**Dil için**	**Açıklama**	**Kitaplıkta sağlandı**
CSQ4BAX0	Çevirici	Kaynak program	SCSQASMS
CSQ4BCX1	C	Kaynak program	SCSQC37S
CSQ4BCX2	C	Kaynak program	SCSQC37S
CSQ4BCX4	C	Kaynak program	SCSQC37S

Not: Kaynak programlar CSQXSTUB ile bağlantı düzenlenir.

z/OS *z/OS üzerinde veri dönüştürme çıkış örnekleri*

Bir veri dönüştürme çıkış yordamı için bir iskelet sağlanır ve IBM MQ ile birlikte MQXCNVC çağrılmasını gösteren bir örnek gönderilir.

Veri dönüştürme çıkış örneklerinin kaynak programlarının adları aşağıdaki çizelgede listelenir:

Çizelge 171. Veri dönüştürme çıkış örnekleri için kaynak (yalnızca çevirici dili)

Üye adı	**Açıklama**	**Kitaplıkta sağlandı**
CSQ4BAX8	Kaynak program	SCSQASMS
CSQ4BAX9	Kaynak program	SCSQASMS
CSQ4CAX9	Kaynak program	SCSQASMS

Not: Kaynak programlar CSQASTUB ile bağlantıyla düzenlenir.

Ek bilgi için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 941 .

z/OS *z/OS üzerinde örnekleri yayınla/abone ol*

Yayınla/Abone Ol örnek programları, IBM MQiçindeki yayınlama ve abone olma özelliklerinin kullanımını gösterir.

IBM MQ Yayınla/Abone Ol arabirimine nasıl programlanacağını gösteren dört C ve iki COBOL programlama dili örnek programı vardır.

Uygulamalar şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQPUT
- MQSUB
- MQGet
- MQCLOSE
- MQDISC
- MQCRTMH
- MQDLTMH
- MQINQMP

Genel/Abone olunan örnek programlar, C ve COBOL programlama dillerinde sağlanır. Örnek uygulamalar toplu iş ortamında çalışır. Toplu iş uygulamaları için Yayınlama/Abone Olma Örnekleri konusuna bakın.

z/OS *z/OS üzerinde istemci bağlantılarını kabul edecek bir kuyruk yöneticisinin yapılandırılması*

Örnek uygulamaları çalıştırabilmeniz için önce bir kuyruk yöneticisi yaratmanız gerekir. Bundan sonra, kuyruk yöneticisini istemci kipinde çalışan uygulamalardan gelen bağlantı isteklerini güvenli bir şekilde kabul edecek şekilde yapılandırabilirsiniz.

Başlamadan önce

Kuyruk yöneticisinin önceden var olduğunu ve başlatıldığını doğrulayın. MQSC komutunu vererek kanal kimlik doğrulama kayıtlarının önceden etkinleştirilip etkinleştirilmediğini saptayın:

```
DISPLAY QMGR CHLAUTH
```

Önemli: Bu görev, kanal kimlik doğrulama kayıtlarının etkinleştirilmesini bekler. Bu, diğer kullanıcılar ve uygulamalar tarafından kullanılan bir kuyruk yöneticisiyse, bu ayarın değiştirilmesi diğer tüm kullanıcıları ve uygulamaları etkiler. Kuyruk yöneticiniz kanal kimlik doğrulama kayıtlarından yararlanmazsa, 4 adımı, MCAUSER ' i *ayrıcalıklı olmayan kullanıcı kimliği* olarak ayarlayan alternatif bir kimlik doğrulama yöntemiyle (örneğin, bir güvenlik çıkışı) değiştirilebilir. Bu yöntem “1” sayfa 1116. adımda edineceksiniz.

Uygulamanın kanalı kullanmasına izin verilebilmesi için, uygulamanızın hangi kanal adını kullanmayı beklediğini bilmeniz gerekir. Uygulamanızın bunları kullanmasına izin verilebilmesi için, uygulamanızın hangi nesneleri (örneğin, kuyruklar ya da konular) kullanmayı beklediğini de bilmeniz gerekir.

Bu görev hakkında

Bu görev, kuyruk yöneticisine bağlanan bir istemci uygulaması için kullanılacak ayrıcalıklı olmayan bir kullanıcı kimliği yaratır. İstemci uygulamasına yalnızca bu kullanıcı kimliğini kullanarak gereksinim duyduğu kanalı ve kuyruğu kullanabilmesi için erişim verilir.

Yordam

1. Kuyruk yöneticinizin çalıştığı sistemde bir kullanıcı kimliği edinin.

   Bu görev için bu kullanıcı kimliği ayrıcalıklı bir yönetici kullanıcı olmamalıdır. Bu kullanıcı kimliği, kuyruk yöneticisinde istemci bağlantısının çalıştırılacağı yetkilidir.
2. Bir dinleyici programı başlatın.
   a) Kanal başlatıcınızın başlatıldığından emin olun. Değilse, **START CHINIT** komutunu vererek başlatın.
   b) Aşağıdaki komutu vererek dinleyici programını başlatın:

   ```
   START LISTENER TRPTYPE(TCP) PORT(nnnn)
   ```

   Burada *nnnn* , seçtiğiniz kapı numarasıdır.
3. Uygulamanız SYSTEM.DEF.SVRCONN , daha sonra bu kanal önceden tanımlanmıştır. Uygulamanız başka bir kanal kullanıyorsa, MQSC komutunu vererek yaratın:

   ```
   DEFINE CHANNEL(' channel-name ') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
   DESCR('Channel for use by sample programs')
   ```

   kanal-adı , kanalınızın adıdır.
4. MQSC komutunu vererek, istemci sisteminizin yalnızca IP adresinin kanalı kullanmasını sağlayacak bir kanal kimlik doğrulama kuralı yaratın:

```
SET CHLAUTH(' channel-name ') TYPE(ADDRESSMAP) ADDRESS(' client-machine-IP-address ') +
MCAUSER(' non-privileged-user-id ')
```

burada:

kanal-adı , kanalınızın adıdır.
istemci-makine-IP-adresi , istemci sisteminizin IP adresidir. Örnek istemci uygulamanız kuyruk yöneticisiyle aynı makinede çalışıyorsa, uygulamanız 'localhost' kullanarak bağlanacaksa '127.0.0.1' IP adresini kullanın. Birkaç farklı istemci makinesi bağlanacaksa, tek bir IP adresi yerine bir kalıp ya da aralık kullanabilirsiniz. Ayrıntılar için bkz. Soysal IP adresleri .
ayrıcalıklı olmayan kullanıcı kimliği , adım “1” sayfa 1116 ' de edindiğiniz kullanıcı kimliğidir.

5. Uygulamanız SYSTEM.DEFAULT.LOCAL.QUEUE, bu kuyruk önceden tanımlanmış. Uygulamanız başka bir kuyruk kullanıyorsa, MQSC komutunu vererek yaratın:

```
DEFINE QLOCAL(' queue-name ') DESCR('Queue for use by sample programs')
```

Burada *kuyruk-adı* , kuyruğunuzun adıdır.

6. Kuyruk yöneticisine bağlanmak ve bu yöneticiye soru sormak için erişim verin:
   a) Kanal başlatıcınızın başlatıldığından emin olun. Değilse, START CHINIT komutunu vererek kanal başlatıcıyı başlatın.
   b) Bir TCP dinleyicisi başlatın; örneğin, aşağıdaki komutu verin:

```
START LISTENER TRPTYPE(TCP) PORT(nnnn)
```

   Burada *nnnn* , seçtiğiniz kapı numarasıdır.

7. Uygulamanız noktadan noktaya iletişim uygulamasıysa, MQSC komutlarını kullanarak kuyruklardan yararlanır, sorgunun yapılmasına izin vermek için erişim verir ve kuyruğunuzu kullanılacak kullanıcı kimliğine göre koyma ve alma iletileri verir:

   RACF komutlarını verin:

```
RDEFINE MQQUEUE qmgr-name.QUEUE. queue-name UACC(NONE)

PERMIT qmgr-name.QUEUE. queue-name CLASS(MQQUEUE) ID(non-privileged-user-id) ACCESS(UPDATE)
```

   burada:

   qmgr-name , kuyruk yöneticinizin adıdır
   kuyruk-adı , kuyruğunuzun adıdır.
   ayrıcalıklı olmayan kullanıcı kimliği , adım “1” sayfa 1116 ' de edindiğiniz kullanıcı kimliğidir.

8. Uygulamanız bir yayınlama/abone olma uygulamasıysa, aşağıdaki RACF komutlarını kullanarak, konuları kullanır, konu başlıklarınızı kullanarak yayınlama ve abone olma izni verir:

```
RDEFINE MQTOPIC qmgr-name.PUBLISH.SYSTEM.BASE.TOPIC UACC(NONE)

PERMIT qmgr-name.PUBLISH.SYSTEM.BASE.TOPIC CLASS(MQTOPIC) ID(non-privileged-user-id)
ACCESS(UPDATE)

RDEFINE MQTOPIC qmgr-name.SUBSCRIBE.SYSTEM.BASE.TOPIC UACC(NONE)

PERMIT qmgr-name.SUBSCRIBE.SYSTEM.BASE.TOPIC CLASS(MQTOPIC) ID(non-privileged-user-id)
ACCESS(UPDATE)
```

   burada:

   qmgr-name , kuyruk yöneticinizin adıdır
   ayrıcalıklı olmayan kullanıcı kimliği , adım “1” sayfa 1116 ' de edindiğiniz kullanıcı kimliğidir.
   Bu, *ayrıcalıklı olmayan kullanıcı kimliği* ' ne konu ağacındaki herhangi bir konuya ilişkin erişim verir; diğer bir seçenek olarak, **DEFINE TOPIC** komutunu kullanarak bir konu nesnesi tanımlayabilir ve

yalnızca o konu nesnesinin başvurduğu konu ağacının kısmına erişim verebilirsiniz. Daha fazla bilgi için Konuların kullanıcı erişimini denetlemebaşlıklı konuya bakın.

Sonraki adım

İstemci uygulamanız artık kuyruk yöneticisine bağlanabilir ve kuyruğu kullanarak ileti koyabilir ya da alabilir.

İlgili kavramlar

z/OS z/OS üzerindeki IBM MQ nesneleriyle çalışma yetkisi

İlgili başvurular

CHLAUTH AYARLA

KANAL TANıMLAYıN

QLOCAL TANıMLAYıN

AUTHREC DEğERINI AYARLA

z/OS *z/OS üzerinde toplu iş ortamı için örnek uygulamaların hazırlanması ve çalıştırılması*

Toplu iş ortamında çalışan bir örnek uygulama hazırlamak için, herhangi bir toplu iş IBM MQ for z/OS uygulaması oluştururken aynı adımları gerçekleştirin.

Bu adımlar “z/OS toplu iş uygulamaları oluşturma” sayfa 980içinde listelenmiştir.

Diğer bir seçenek olarak, bir örneğin yürütülebilir bir biçimini sağladığımız yerde, bunu thlqual.SCSQLOAD yükleme kitaplığından çalıştırabilirsiniz.

Not: Göz At örneğinin çevirici dil sürümü, veri denetim bloklarını (DCB ' ler) kullandığından, RMODE(24) komutunu kullanarak bağlantı düzenlemeniz gerekir.

Kullanılacak kitaplık üyeleri Çizelge 172 sayfa 1119, Çizelge 173 sayfa 1119, Çizelge 174 sayfa 1119ve Çizelge 175 sayfa 1120içinde listelenir.

Kullanmak istediğiniz örnekler için sağlanan çalıştırma JCL ' sini düzenlemeniz gerekir (bkz. Çizelge 172 sayfa 1119, Çizelge 173 sayfa 1119, Çizelge 174 sayfa 1119ve Çizelge 175 sayfa 1120).

Sağlanan JCL ' deki PARM deyimi değiştirmeniz gereken bazı değiştirgeler içeriyor. C örnek programlarını çalıştırmak için parametreleri alanlarla ayırın; çevirici, COBOL ve PL/I örnek programlarını çalıştırmak için bunları virgüllerle ayırın. Örneğin, kuyruk yöneticinizin adı CSQ1 ise ve uygulamayı COBOL, PL/I ve çevirici dili JCL dillerinde LOCALQ1adlı bir kuyrukla çalıştırmak istiyorsanız, PARM deyiminiz şöyle olmalıdır:

```
PARM=(CSQ1,LOCALQ1)
```

C dilinde JCL, PARM deyiminizin şöyle görünmesi gerekir:

```
PARM=('CSQ1 LOCALQ1')
```

Artık işleri sunmaya hazırsınız.

z/OS *z/OS üzerindeki örnek toplu iş uygulamalarının adları*

Örnek toplu iş uygulamaları için sağlanan programların özeti.

Toplu iş uygulama programları aşağıdaki çizelgelerde özetlenir:

- Çizelge 172 sayfa 1119 Koyma ve Alma örnekleri
- Çizelge 173 sayfa 1119 Göz At örneği
- Çizelge 174 sayfa 1119 İleti örneğini yazdır
- Çizelge 175 sayfa 1120 Yayınlama/Abone Olma örnekleri
- Çizelge 176 sayfa 1120 Diğer örnekler

Çizelge 172. Batch Put ve Get örnekleri

Üye adı	Dil için	Açıklama	Kitaplıkta sağlanan kaynak dosya	Kitaplıkta sağlanan yürütülür dosya
CSQ4BCJ1	C	Kaynak programı al	SCSQC37S	SCSQLOAD
CSQ4BCK1	C	Kaynak programı koy	SCSQC37S	SCSQLOAD
CSQ4BCJR	C	CSQ4BCJ1 ve CSQBCK1 için JCL örnek çalıştırma	SCSQPROC	Yok
CSQ4BVJ1	COBOL	Kaynak programı al	SCSQCOBS	SCSQLOAD
CSQ4BVK1	COBOL	Kaynak programı koy	SCSQCOBS	SCSQLOAD
CSQ4BVJR	COBOL	CSQBVJ1 ve CSQBVK1 için örnek çalıştırma JCL	SCSQPROC	Yok

Çizelge 173. Toplu İş Gözat örneği

Üye adı	Dil için	Açıklama	Kitaplıkta sağlanan kaynak dosya	Kitaplıkta sağlanan yürütülür dosya
CSQ4BVA1	COBOL	Kaynak program	SCSQCOBS	SCSQLOAD
CSQ4BVAR	COBOL	CSQ4BVA1 için örnek çalıştırma JCL	SCSQPROC	Yok
CSQ4BAA1	Çevirici	Kaynak program	SCSQASMS	SCSQLOAD
CSQ4BAAR	Çevirici	CSQ4BAA1 için örnek çalıştırma JCL	SCSQPROC	Yok
CSQ4BCA1	C	Kaynak program	SCSQC37S	SCSQLOAD
CSQ4BCAR	C	CSQ4BCA1 için örnek çalıştırma JCL	SCSQPROC	Yok
CSQ4BPA1	PL/I	Kaynak program	SCSQPLIS	SCSQLOAD
CSQ4BPAR	PL/I	CSQ4BPA1 için örnek çalıştırma JCL	SCSQPROC	Yok

Çizelge 174. Toplu Yazdırma İletisi örneği (yalnızca C dili)

Üye adı	Açıklama	Kitaplıkta sağlanan kaynak dosya	Kitaplıkta sağlanan yürütülür dosya
CSQ4BCG1	Kaynak program	SCSQC37S	SCSQLOAD
CSQ4BCGR	CSQ4BCG1 için örnek çalıştırma JCL	SCSQPROC	Yok

Çizelge 174. Toplu Yazdırma İletisi örneği (yalnızca C dili) (devamı var)			
Üye adı	**Açıklama**	**Kitaplıkta sağlanan kaynak dosya**	**Kitaplıkta sağlanan yürütülür dosya**
CSQ4BCL1	Kaynak programa göz at	SCSQC37S	SCSQLOAD
CSQ4BCLR	CSQ4BCL1 için örnek çalıştırma JCL	SCSQPROC	Yok

Çizelge 175. Yayınlama/Abone Olma örnekleri					
Üye adı	**Dil için**	**Açıklama**	**Kitaplıkta sağlanan kaynak dosya**	**SCSQPROC içinde JCL**	**Kitaplıkta sağlanan yürütülür dosya**
CSQ4BCP1	C	Konu kaynak programına yayınla	SCSQC37S	CSQ4BCPP	SCSQLOAD
CSQ4BCP2	C	Konuya abone olun ve iletileri alın kaynak programı	SCSQC37S	CSQ4BCPS	SCSQLOAD
CSQ4BCP3	C	Kullanıcı tarafından sağlanan bir hedef ve ileti alma kaynak programını kullanarak konuya abone olun	SCSQC37S	CSQ4BCPD	SCSQLOAD
CSQ4BCP4	C	Genişletilmiş seçenekleri kullanarak konuya abone olun ve iletileri alın kaynak programı	SCSQC37S	CSQ4BCPE	SCSQLOAD
CSQ4BVP1	COBOL	Konu kaynak programına yayınla	SCSQCOBS	CSQ4BVPP	SCSQLOAD
CSQ4BVP2	COBOL	Konuya abone olun ve iletileri alın kaynak programı	SCSQCOBS	CSQ4BVPS	SCSQLOAD

Çizelge 176. Diğer örnekler					
Üye adı	**Dil için**	**Açıklama**	**Kitaplıkta sağlanan kaynak dosya**	**SCSQPROC içinde JCL**	**Kitaplıkta sağlanan yürütülür dosya**
CSQ4BCS1	C	Zamanuyumsuz tüketim kaynağı programı	SCSQC37S	CSQ4BCSC	SCSQLOAD
CSQ4BCS2	C	Zamanuyumsuz Put ve Check status kaynak programı	SCSQC37S	CSQ4BCSP	SCSQLOAD
CSQ4BCM1	C	İleti özellikleri kaynak programını sorgularken	SCSQC37S	CSQ4BCMP	SCSQLOAD
CSQ4BCM2	C	İleti özellikleri kaynak programını ayarla	SCSQC37S	CSQ4BCMP	SCSQLOAD

z/OS *z/OS üzerinde TSO ortamı için örnek uygulamaların hazırlanması*

TSO ortamında çalışan bir örnek uygulama hazırlamak için, herhangi bir toplu iş IBM MQ for z/OS uygulaması oluştururken gerçekleştirdiğiniz adımları gerçekleştirin.

Bu adımlar “z/OS toplu iş uygulamaları oluşturma” sayfa 980içinde listelenmiştir. Kullanılacak kitaplık üyeleri Çizelge 177 sayfa 1121içinde listelenir.

Diğer bir seçenek olarak, bir örneğin yürütülebilir bir biçimini sağladığımız yerde, bunu thlqual.SCSQLOAD yükleme kitaplığından çalıştırabilirsiniz.

Mail Manager örnek uygulaması için, kullandığı kuyrukların sisteminizde kullanılabilir olduğundan emin olun. Bunlar **thlqual**.SCSQPROC (CSQ4CVD) üyesinde tanımlanır. Bu kuyrukların her zaman kullanılabilir olmasını sağlamak için, bu üyeleri CSQINP2 kullanıma hazırlama giriş veri kümesine ekleyebilir ya da bu kuyruk tanımlamalarını yüklemek için CSQUTIL programını kullanabilirsiniz.

z/OS *z/OS üzerindeki örnek TSO uygulamalarının adları*

Örnek TSO uygulamalarının her biri için sağlanan programların adları ve kaynak, JCL ve yalnızca Message Handler örneği için yürütülür dosyaların bulunduğu kitaplıklar hakkında bilgi.

TSO uygulama programları aşağıdaki çizelgelerde özetlenmiştir:

- Çizelge 177 sayfa 1121 Mail Manager Sample
- Çizelge 178 sayfa 1122 İleti işleyici örneği

Bu örnekler ISPF panolarını kullanır. Bu nedenle, programlara bağlantı düzenlerken ISPF sınırlı kod öbeğini (ISPLINK) eklemeniz gerekir.

Çizelge 177. TSO Mail Manager Sample

Üye adı	**Dil için**	**Açıklama**	**Kitaplıkta sağlanan kaynak dosya**
CSQ4CVD	Bağımsız	IBM MQ for z/OS nesne tanımlamaları	SCSQPROC
CSQ40	Bağımsız	ISPF iletileri	SCSQMSGE
CSQ4RVD1	COBOL	CSQ4TVD1 ' i başlatmak için CLIST	SCSQCLST
CSQ4TVD1	COBOL	Menü programı için kaynak program	SCSQCOBS
CSQ4TVD2	COBOL	Posta Alma programı için kaynak program	SCSQCOBS
CSQ4TVD4	COBOL	Posta Gönderme programına ilişkin kaynak program	SCSQCOBS
CSQ4TVD5	COBOL	Nickname programına ilişkin kaynak program	SCSQCOBS
CSQ4VDP1-6	COBOL	Pano tanımları	SCSQPNLA
CSQ4VD0	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VD1	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VD2	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VD4	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4RCD1	C	CSQ4TCD1 ' i başlatmak için CLIST	SCSQCLST

Çizelge 177. TSO Mail Manager Sample (devamı var)			
Üye adı	**Dil için**	**Açıklama**	**Kitaplıkta sağlanan kaynak dosya**
CSQ4TCD1	C	Menü programı için kaynak program	SCSQC37S
CSQ4TCD2	C	Posta Alma programı için kaynak program	SCSQC37S
CSQ4TCD4	C	Posta Gönderme programına ilişkin kaynak program	SCSQC37S
CSQ4TCD5	C	Nickname programına ilişkin kaynak program	SCSQC37S
CSQ4CDP1-6	C	Pano tanımları	SCSQPNLA
CSQ4TC0	C	Dosya ekle	SCSQC370

Çizelge 178. TSO Message Handler Sample				
Üye adı	**Dil için**	**Açıklama**	**Kitaplıkta sağlanan kaynak dosya**	**Kitaplıkta sağlanan yürütülür dosya**
CSQ4TCH0	C	Veri tanımlaması	SCSQC370	Yok
CSQ4TCH1	C	Kaynak program	SCSQC37S	SCSQLOAD
CSQ4TCH2	C	Kaynak program	SCSQC37S	SCSQLOAD
CSQ4TCH3	C	Kaynak program	SCSQC37S	SCSQLOAD
CSQ4RCH1	C ve COBOL	CSQ4TCH1 ya da CSQ4TVH1 başlatmak için CLIST	SCSQCLST	Yok
CSQ4CHP1	C ve COBOL	Pano tanımı	SCSQPNLA	Yok
CSQ4CHP2	C ve COBOL	Pano tanımı	SCSQPNLA	Yok
CSQ4CHP3	C ve COBOL	Pano tanımı	SCSQPNLA	Yok
CSQ4CHP9	C ve COBOL	Pano tanımı	SCSQPNLA	Yok
CSQ4TVH0	COBOL	Veri tanımlaması	SCSQCOBC	Yok
CSQ4TVH1	COBOL	Kaynak program	SCSQCOBS	SCSQLOAD
CSQ4TVH2	COBOL	Kaynak program	SCSQCOBS	SCSQLOAD
CSQ4TVH3	COBOL	Kaynak program	SCSQCOBS	SCSQLOAD

z/OS *z/OS üzerinde CICS ortamı için örnek uygulamaların hazırlanması*

CICS örnek programlarını çalıştırmadan önce 32702 LOGMODE 'unu kullanarak CICS ' da oturum açın. Bunun nedeni, örnek programların 3270 kipi 2 ekranını kullanmak üzere yazılmış olmasıdır.

CICS ortamında çalışan bir örnek uygulama hazırlamak için aşağıdaki adımları izleyin:

1. BMS ekran tanımlaması kaynağını (**thlqual**.SCSQMAPS kitaplığında sağlanır; burada **thlqual** , kuruluşunuz tarafından kullanılan üst düzey niteleyicidir) birleştirerek, örnek için simgesel tanım

eşlemi ve fiziksel ekran eşlemi yaratın. Eşlemleri adlandırdığınızda, BMS ekran tanımlaması kaynağının adını kullanın (Put ve Get örnek programları için kullanılamaz), ancak bu adın son karakterini atlayın.

2. Herhangi bir CICS IBM MQ for z/OS uygulamasını oluştururken gerçekleştirdiğiniz adımların aynısını gerçekleştirin. Bu adımlar "z/OS içinde CICS uygulamaları oluşturma" sayfa 983içinde listelenmiştir. Kullanılacak kitaplık üyeleri Çizelge 179 sayfa 1123, Çizelge 180 sayfa 1124, Çizelge 181 sayfa 1124ve Çizelge 182 sayfa 1125içinde listelenir.

   Alternatif olarak, bir örneğin yürütülebilir bir biçimini sağladığımız yerde, bunu thlqual.SCSQCICS yükleme kitaplığından çalıştırabilirsiniz.

3. CICS sistem tanımlaması (CSD) veri kümesini güncelleyerek eşlem kümesini, programları ve hareketi CICS olarak tanımlayın. Gerek duyduğunuz tanımlar **thlqual**.SCSQPROC (CSQ4S100) üyesinde bulunur. Bunun nasıl yapılacağını öğrenmek için, CICS Transaction Server for z/OS 4.1 ürün belgelerinde *CICS-IBM MQ Adapter* bölümüne bakın: CICS Transaction Server for z/OS 4.1, The CICS-IBM MQ adapter.

   Not: Credit Check örnek uygulaması için, örneğin kullandığı VSAM veri kümesini önceden oluşturmadıysanız bu aşamada bir hata iletisi alırsınız.

4. Credit Check ve Mail Manager örnek uygulamaları için, kullandıkları kuyrukların sisteminizde kullanılabilir olduğundan emin olun. Credit Check örneği için, bunlar COBOL için **thlqual**.SCSQPROC (CSQ4CVB) ve C için **thlqual**.SCSQPROC (CSQ4CCB) üyesinde tanımlanır. Mail Manager örneği için, bunlar **thlqual**.SCSQPROC (CSQ4CVD) üyesinde tanımlanır. Bu kuyrukların her zaman kullanılabilir olmasını sağlamak için, bu üyeleri CSQINP2 kullanıma hazırlama giriş veri kümesine ekleyebilir ya da bu kuyruk tanımlamalarını yüklemek için CSQUTIL programını kullanabilirsiniz.

   Kuyruk Öznitelikleri örnek uygulaması için, diğer örnek uygulamalar için sağlanan kuyruklardan birini ya da daha fazlasını kullanabilirsiniz. Diğer bir seçenek olarak, kendi kuyruklarınızı da kullanabilirsiniz. Ancak, bu örnek, sağlandığı formda, adlarının ilk sekiz baytında CSQ4SAMP karakterlerini içeren kuyruklar ile çalışır.

z/OS *z/OS üzerindeki örnek CICS uygulamalarının adları*

Bu konuda, örnek CICS uygulamaları için sağlanan programların bir özeti sağlanır.

CICS uygulama programları aşağıdaki çizelgelerde özetlenir:

- Çizelge 179 sayfa 1123 Koyma ve Alma örnekleri
- Çizelge 180 sayfa 1124 Kuyruk Öznitelikleri örneği
- Çizelge 181 sayfa 1124 Mail Manager örneği (yalnızca COBOL)
- Çizelge 182 sayfa 1125 Kredi Denetimi örneği
- Çizelge 183 sayfa 1125 Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örnekleri

Çizelge 179. CICS Koyma ve Alma örnekleri

Üye adı	**Dil için**	**Açıklama**	**Kitaplıkta sağlanan kaynak dosya**	**Kitaplıkta sağlanan yürütülür dosya**
CSQ4CCK1	C	Kaynak programı koy	SCSQC37S	SCSQCICS
CSQ4CCJ1	C	Kaynak programı al	SCSQC37S	SCSQCICS
CSQ4CVJ1	COBOL	Kaynak programı al	SCSQCOBS	SCSQCICS
CSQ4CVK1	COBOL	Kaynak programı koy	SCSQCOBS	SCSQCICS
CSQ4S100	Bağımsız	CICS sistem tanımlaması veri kümesi	SCSQPROC	Yok

Çizelge 180. CICS Kuyruk Öznitelikleri örneği

Üye adı	Dil için	Açıklama	Kitaplıkta sağlanan kaynak dosya	Kitaplıkta sağlanan yürütülür dosya
CSQ4CVC1	COBOL	Kaynak program	SCSQCOBS	SCSQCICS
CSQ4VMSG	COBOL	İleti Tanımlaması	SCSQCOBC	Yok
CSQ4VCMS	COBOL	BMS ekran tanımı	SCSQMAPS	SCSQCICS (CSQ4ACMolarak adlandırılır)
CSQ4CAC1	Çevirici	Kaynak program	SCSQASMS	SCSQCICS
CSQ4AMSG	Çevirici	İleti Tanımlaması	SCSQMACS	Yok
CSQ4ACMS	Çevirici	BMS ekran tanımı	SCSQMAPS	SCSQCICS (CSQ4ACMolarak adlandırılır)
CSQ4CCC1	C	Kaynak program	SCSQC37S	SCSQCICS
CSQ4CMSG	C	İleti Tanımlaması	SCSQC370	Yok
CSQ4CCMS	C	BMS ekran tanımı	SCSQMAPS	SCSQCICS (CSQ4ACMolarak adlandırılır)
CSQ4S100	Bağımsız	CICS sistem tanımlaması veri kümesi	SCSQPROC	Yok

Çizelge 181. CICS Mail Manager örneği (yalnızca COBOL)

Üye adı	Açıklama	Kitaplıkta sağlanan kaynak dosya
CSQ4CVD	IBM MQ for z/OS nesne tanımlamaları	SCSQPROC
CSQ4CVD1	Menü programı için kaynak	SCSQCOBS
CSQ4CVD2	Posta Alma programına ilişkin kaynak	SCSQCOBS
CSQ4CVD3	İleti Görüntüleme Kaynağı programı	SCSQCOBS
CSQ4CVD4	Posta Gönderme programına ilişkin kaynak	SCSQCOBS
CSQ4CVD5	Takma ad programının kaynağı	SCSQCOBS
CSQ4VDMS	BMS ekran tanımlaması kaynağı	SCSQMAPS
CSQ4S100	CICS sistem tanımlaması veri kümesi	SCSQPROC
CSQ4VD0	Veri tanımlaması	SCSQCOBC
CSQ4VD3	Veri tanımlaması	SCSQCOBC
CSQ4VD4	Veri tanımlaması	SCSQCOBC

Çizelge 182. CICS Kredi Denetimi örneği

Üye adı	Dil için	Açıklama	Kitaplıkta sağlanan kaynak dosya
CSQ4CVB	Bağımsız	IBM MQ nesne tanımlamaları	SCSQPROC
CSQ4CCB	Bağımsız	IBM MQ nesne tanımlamaları	SCSQPROC
CSQ4CVB1	COBOL	Kullanıcı-arabirim programı için kaynak	SCSQCOBS
CSQ4CVB2	COBOL	Kredi uygulaması yöneticisi için kaynak	SCSQCOBS
CSQ4CVB3	COBOL	Hesap denetleme programı için kaynak	SCSQCOBS
CSQ4CVB4	COBOL	Dağıtım programı kaynağı	SCSQCOBS
CSQ4CVB5	COBOL	Ajans-sorgu programı kaynağı	SCSQCOBS
CSQ4CCB1	C	Kullanıcı-arabirim programı için kaynak	SCSQC37S
CSQ4CCB2	C	Kredi uygulaması yöneticisi için kaynak	SCSQC37S
CSQ4CCB3	C	Hesap denetleme programı için kaynak	SCSQC37S
CSQ4CCB4	C	Dağıtım programı kaynağı	SCSQC37S
CSQ4CCB5	C	Ajans-sorgu programı kaynağı	SCSQC37S
CSQ4CB0	C	Dosya ekle	SCSQC370
CSQ4CBMS	C	BMS ekran tanımlaması kaynağı	SCSQMAPS
CSQ4VBMS	COBOL	BMS ekran tanımlaması kaynağı	SCSQMAPS
CSQ4VB0	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VB1	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VB2	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VB3	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VB4	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VB5	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VB6	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VB7	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4VB8	COBOL	Veri tanımlaması	SCSQCOBC
CSQ4BAQ	Bağımsız	VSAM veri kümesi kaynağı	SCSQPROC
CSQ4FILE	Bağımsız	CSQ4CVB3 tarafından kullanılan VSAM veri kümesini oluşturmak için JCL	SCSQPROC
CSQ4S100	Bağımsız	CICS sistem tanımlaması veri kümesi	SCSQPROC

Çizelge 183. CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örnekleri

Üye adı	Açıklama	Kitaplıkta sağlanan kaynak dosya
CSQ4CVCN	Basit İleti Tüketimi Programı Kaynağı	SCSQCOBS

Çizelge 183. CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örnekleri (devamı var)		
Üye adı	**Açıklama**	**Kitaplıkta sağlanan kaynak dosya**
CSQ4CVCT	Denetim İletisi Tüketim Kaynağı programı	SCSQCOBS
CSQ4CVEV	Kaynak for Event Handler programı	SCSQCOBS
CSQ4CVPT	Message Put Client programı için kaynak	SCSQCOBS
CSQ4CVRG	Kayıt İçin Kaynak İstemcisi programı	SCSQCOBS
CSQ4S100	CICS Sistem Tanımlaması veri kümesi	SCSQPROC

z/OS *z/OS üzerindeki IMS ortamı için örnek uygulamanın hazırlanması*

Credit Check örnek uygulamasının bir parçası IMS ortamında çalışabilir.

Uygulamanın bu bölümünü CICS örneğiyle çalışacak şekilde hazırlamak için öncelikle "z/OS üzerinde CICS ortamı için örnek uygulamaların hazırlanması" sayfa 1122içinde açıklanan adımları gerçekleştirin.

Daha sonra aşağıdaki adımları gerçekleştirin:

1. Herhangi bir IMS IBM MQ for z/OS uygulamasını oluştururken gerçekleştirdiğiniz adımların aynısını gerçekleştirin. Bu adımlar "IMS (BMP ya da MPP) uygulamalarını oluşturma" sayfa 984içinde listelenmiştir. Kullanılacak kitaplık üyeleri Çizelge 184 sayfa 1126içinde listelenir.
2. IMSiçin uygulama programını ve veritabanını tanımlayın. Örnekler, bunu etkinleştirmek için PSBGEN, DBDGEN, ACB tanımı, IMSGEN ve IMSDALOC deyimleriyle birlikte sağlanır.
3. Bu amaçla sağlanan örnek JCL ' yi (CSQ4ILDB) uyarlayarak ve çalıştırarak CSQ4CA veritabanını yükleyin. Bu JCL, CSQ4BAQkütüğündeki verilerle veritabanını yükler. IMS denetim bölgesini CSQ4CAveritabanı için DD deyimiyle güncelleyin.
4. Bu amaçla sağlanan örnek JCL ' yi uyarlayarak ve çalıştırarak, denetim hesabı programını toplu ileti işleme (BMP) programı olarak başlatın. Bu JCL, toplu iş odaklı bir BMP programı başlatır. Programı ileti yönelimli bir BMP programı olarak çalıştırmak için, JCL ' de IN= deyimini içeren satırdan açıklama karakterlerini kaldırın.

z/OS *z/OS üzerindeki örnek IMS uygulamasının adları*

Bu bilgiler, Kredi Denetimi örnek IMS uygulaması için sağlanan kaynakların ve JCLlerin listesini içeren bir tablo sağlar.

Çizelge 184. Credit Check IMS örneği için Kaynak ve JCL (yalnızca C)		
Üye adı	**Açıklama**	**Kitaplıkta sağlandı**
CSQ4CVB	IBM MQ nesne tanımlamaları	SCSQPROC
CSQ4ICB3	Hesap denetleme programı için kaynak	SCSQC37S
CSQ4ICBL	Çek hesabı veritabanını yükleme kaynağı	SCSQC37S
CSQ4CBI	Veri tanımlaması	SCSQC370
CSQ4PSBL	Veritabanı yükleme programı için PSBGEN JCL	SCSQPROC

Çizelge 184. Credit Check IMS örneği için Kaynak ve JCL (yalnızca C) (devamı var)		
Üye adı	**Açıklama**	**Kitaplıkta sağlandı**
CSQ4PSB3	Hesap denetleme programı için PSBGEN JCL	SCSQPROC
CSQ4DBDS	CSQ4CA veritabanı için DBDGEN JCL	SCSQPROC
CSQ4GIMS	IMS CSQ4IVB3 ve CSQ4CA için GEN makro tanımlamaları	SCSQPROC
CSQ4ACBG	CSQ4IVB3 için uygulama denetim bloğu (ACB) tanımı	SCSQPROC
CSQ4BAQ	Veritabanı için kaynak	SCSQPROC
CSQ4ILDB	Veritabanı yükleme işi için JCL çalıştırma örneği	SCSQPROC
CSQ4ICBR	Çek hesabı programı için örnek çalıştırma JCL	SCSQPROC
CSQ4DYNA	IMSVeritabanı için DALOC makro tanımlamaları	SCSQPROC

z/OS *z/OS üzerindeki Put örnekleri*

Put örnek programları, MQPUT çağrılarını kullanarak iletileri kuyruğa koyar.

Kaynak programlar, toplu iş ve CICS ortamlarında C ve COBOL dilinde sağlanır (bkz. Çizelge 172 sayfa 1119 ve Çizelge 179 sayfa 1123).

Put örneğinin tasarımı

Program mantığından geçen akış:

1. MQCONN çağrıyı kullanarak kuyruk yöneticisine bağlanın. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun.

   Not: Örneği bir CICS ortamında çalıştırıyorsanız, bir MQCONN çağrısı yayınlamanız gerekmez; bunu yaparsanız, DEF_HCONN döndürülür. İzleyen MQI çağrıları için MQHC_DEF_HCONN bağlantı tanıtıcısını kullanabilirsiniz.
2. MQOO_OUTPUT seçeneğiyle MQOPEN çağrısıyla kuyruğu açın. Bu çağrıya girişte, program “1” sayfa 1129. adımda döndürülen bağlantı tanıtıcısını kullanır. Nesne tanımlayıcı yapısı (MQOD) için, programa parametre olarak geçirilen kuyruk adı alanı dışındaki tüm alanlar için varsayılan değerleri kullanır. MQOPEN çağrısı başarısız olursa, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun.
3. Gereken sayıda ileti kuyruğa konuncaya kadar, program içinde MQPUT çağrıları yayınlayan bir döngü yaratın. Bir MQPUT çağrısı başarısız olursa, döngüden erken vazgeçilir, başka MQPUT çağrısı denenmez ve tamamlanma ve neden kodları döndürülür.
4. “2” sayfa 1129. adımda döndürülen nesne tanıtıcısıyla MQCLOSE çağrısını kullanarak kuyruğu kapatın. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın.
5. “1” sayfa 1129. adımda döndürülen bağlantı tanıtıcısıyla MQDISC çağrısını kullanarak kuyruk yöneticisiyle bağlantıyı kesin. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın.

   Not: Örneği bir CICS ortamında çalıştırıyorsanız, MQDISC çağrısı yapmanıza gerek yoktur.

z/OS *z/OS üzerindeki toplu iş ortamı için Put örnekleri*

Toplu iş ortamı için Put örnekleri dikkate alındığında bu konuyu kullanın.

Örnekleri çalıştırmak için örnek JCL ' yi düzenleyin ve çalıştırın (bkz. “z/OS üzerinde toplu iş ortamı için örnek uygulamaların hazırlanması ve çalıştırılması” sayfa 1118).

Bu programlar, bir EXEC PARM içinde C ve COBOL içinde virgüllerle ayrılmış olarak aşağıdaki parametreleri alır:

1. Kuyruk yöneticisinin adı (4 karakter)
2. Hedef kuyruğun adı (48 karakter)
3. İleti sayısı (en çok 4 basamak)
4. İletide yazılması gereken doldurma karakteri (1 karakter)
5. İletiye yazılacak karakter sayısı (en çok 4 basamak)
6. İletinin kalıcılığı (1 karakter: Kalıcı için P ya da kalıcı olmayan için N)

Bu parametrelerden herhangi birini yanlış girerseniz, uygun hata iletileri alırsınız.

Örneklerden gelen iletiler SYSPRINT veri kümesine yazılır.

Kullanım notları

- Örnekleri basit tutmak için, dil sürümleri arasında bazı küçük işlevsel farklılıklar vardır. Ancak, örnek çalıştırma JCL, CSQ4BCJRve CSQ4BVJR' de gösterilen parametrelerin yerleşim düzenini kullanırsanız, bu farklılıklar en aza indirilir. Farkların hiçbiri MQI ile ilgili değil.
- CSQ4BCK1 , gönderilen ileti sayısı ve ileti uzunluğu için dörtten fazla sayı girmenizi sağlar.
- İki sayısal alan için, 1-9999 aralığında herhangi bir sayı girin. Girdiğiniz değer pozitif bir sayı olmalıdır. Örneğin, tek bir ileti koymak için değer olarak 1, 01, 001 ya da 0001 girebilirsiniz. Sayısal olmayan ya da negatif değerler girerseniz, bir hata alabilirsiniz. Örneğin, -1 değerini girerseniz, COBOL programı 1 baytlık bir ileti gönderir, ancak C programı bir hata alır.
- Her iki program için de, CSQ4BCK1 ve CSQ4BVK1, iletinin kalıcı olmasını istiyorsanız, kalıcılık değiştirgesine (+ + PER + +) P girmeniz gerekir. Bunu yapmazsanız, ileti kalıcı olmaz.

z/OS *z/OS üzerindeki CICS ortamı için Put örnekleri*

CICS ortamı için Put örnekleri dikkate alındığında bu konuyu kullanın.

Hareketler, virgülle ayrılmış olarak aşağıdaki parametreleri alır:

1. İleti sayısı (en çok 4 basamak)
2. İletide yazılması gereken doldurma karakteri (1 karakter)
3. İletiye yazılacak karakter sayısı (en çok 4 basamak)
4. İletinin kalıcılığı (1 karakter: Kalıcı için P ya da kalıcı olmayan için N)
5. Hedef kuyruğun adı (48 karakter)

Bu parametrelerden herhangi birini yanlış girerseniz, uygun hata iletileri alırsınız.

COBOL örneği için, aşağıdakileri girerek CICS ortamındaki Put örneğini çağırın:

```
MVPT,9999,*,9999,P,QUEUE.NAME
```

C örneği için, CICS ortamındaki Put örneğini aşağıdakileri girerek çağırın:

```
MCPT,9999,*,9999,P,QUEUE.NAME
```

Örneklerden gelen iletiler ekranda görüntülenir.

Kullanım notları

- Örnekleri basit tutmak için, dil sürümleri arasında bazı küçük işlevsel farklılıklar vardır. Farkların hiçbiri MQI ile ilgili değil.

- 48 karakterden uzun bir kuyruk adı girerseniz, uzunluğu 48 karakter üst sınırına kadar kesilir, ancak hata iletisi döndürülmez.
- Hareketi girmeden önce CLEAR tuşuna basın.
- İki sayısal alan için, 1-9999 aralığında herhangi bir sayı girin. Girdiğiniz değer pozitif bir sayı olmalıdır. Örneğin, tek bir ileti koymak için 1, 01, 001 ya da 0001 değerini girebilirsiniz. Sayısal olmayan ya da negatif değerler girerseniz, bir hata alabilirsiniz. Örneğin, -1 değerini girerseniz, COBOL programı 1 baytlık bir ileti gönderir ve C programı malloc () ile ilgili bir hata ile olağandışı biter.
- Her iki program için de, CSQ4CCK1 ve CSQ4CVK1, iletinin kalıcı olmasını istiyorsanız, kalıcılık değiştirgesine P girin. Kalıcı olmayan iletiler için, kalıcılık değiştirgesine N girin. Başka bir değer girerseniz bir hata iletisi alırsınız.
- Varsayılan değerler, program çağrısı sırasında ayarlananlar dışındaki tüm parametreler için kullanıldığından, iletiler eşitleme noktasına konur.

z/OS *z/OS üzerindeki Get örnekleri*

Örnek programları al, MQGET çağrıyı kullanarak kuyruktan ileti alır.

Kaynak programlar, toplu iş ve CICS ortamlarında C ve COBOL dilinde sağlanır (bkz. Çizelge 172 sayfa 1119 ve Çizelge 179 sayfa 1123).

z/OS *z/OS üzerinde Get örneğinin tasarımı*

"Get" örneğinin tasarımı ve göz önünde bulundurulması gereken bazı kullanım notları hakkında bilgi edinin.

Program mantığından geçen akış:

1. MQCONN çağrıyı kullanarak kuyruk yöneticisine bağlanın. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun.

   Not: Örneği bir CICS ortamında çalıştırıyorsanız, bir MQCONN çağrısı yayınlamanız gerekmez; bunu yaparsanız, DEF_HCONN döndürülür. İzleyen MQI çağrıları için MQHC_DEF_HCONN bağlantı tanıtıcısını kullanabilirsiniz.
2. MQOO_INPUT_SHARED ve MQOO_BROWSE seçenekleriyle MQOPEN çağrısıyla kuyruğu açın. Bu çağrıya girişte, program “1” sayfa 1129. adımda döndürülen bağlantı tanıtıcısını kullanır. Nesne tanımlayıcı yapısı (MQOD) için, programa parametre olarak geçirilen kuyruk adı alanı dışındaki tüm alanlar için varsayılan değerleri kullanır. MQOPEN çağrısı başarısız olursa, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun.
3. Kuyruktan gerekli sayıda ileti alınıncaya kadar, program içinde MQGET çağrıları yayınlayan bir döngü yaratın. Bir MQGET çağrısı başarısız olursa, döngüden erken vazgeçilir, başka MQGET çağrısı denenmez ve tamamlanma ve neden kodları döndürülür. MQGET çağrısında aşağıdaki seçenekler belirtildi:
   - MQGMO_NO_WAIT
   - MQGMO_ACCEPT_TRUNCATED_MESSAGE
   - MQGMO_SYNCPOINT ya da MQGMO_NO_SYNCPOINT
   - MQGMO_BROWSE_FIRST ve MQGMO_BROWSE_NEXT

   Bu seçeneklere ilişkin açıklamalar için bkz. MQGET. Her ileti için, ileti numarasının ardından iletinin uzunluğu ve ileti verileri yazılır.
4. “2” sayfa 1129. adımda döndürülen nesne tanıtıcısıyla MQCLOSE çağrısını kullanarak kuyruğu kapatın. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın.
5. “1” sayfa 1129. adımda döndürülen bağlantı tanıtıcısıyla MQDISC çağrısını kullanarak kuyruk yöneticisiyle bağlantıyı kesin. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın.

   Not: Örneği bir CICS ortamında çalıştırıyorsanız, MQDISC çağrısı yapmanıza gerek yoktur.

Kullanım notları

- Örnekleri basit tutmak için, dil sürümleri arasında bazı küçük işlevsel farklılıklar vardır. Ancak, JCL, CSQ4BCJRve CSQ4BVJRörnek çalıştırmasında gösterilen parametrelerin düzenini kullanırsanız, bu farklılıklar en aza indirilir. Farkların hiçbiri MQI ile ilgili değil.
- CSQ4BCJ1 , alınan iletilerin sayısı için dört basamaktan fazlasını girmenizi sağlar.
- 64 KB değerinden uzun iletiler kesilir.
- CSQ4BCJ1 yalnızca ilk NULL (\0) karakteri görüntüleninceye kadar görüntülendiği için karakter iletilerini doğru olarak görüntüleyebilir.
- Sayısal ileti sayısı alanında, 1-9999 aralığında herhangi bir sayı girin. Girdiğiniz değer pozitif bir sayı olmalıdır. Örneğin, tek bir ileti almak için değer olarak 1, 01, 001 ya da 0001 girebilirsiniz. Sayısal olmayan ya da negatif değerler girerseniz, bir hata alabilirsiniz. Örneğin, -1 girerseniz, COBOL programı bir ileti alır, ancak C programı herhangi bir ileti almaz.
- İletilere göz atmak istiyorsanız, her iki program için de CSQ4BCJ1 ve CSQ4BVJ1, get (alma) değiştirgesine (+ + GET + +) B girin.
- Her iki program için de CSQ4BCJ1 ve CSQ4BVJ1, eşitleme noktasında alınacak iletiler için syncpoint değiştirgesine (+ + SYNC + +) S girin.

z/OS *z/OS üzerinde toplu iş ortamı için örnekleri al*

Örnekleri çalıştırmak için örnek JCL ' yi düzenleyin ve çalıştırın (bkz. “z/OS üzerinde toplu iş ortamı için örnek uygulamaların hazırlanması ve çalıştırılması” sayfa 1118).

Bu programlar, bir EXEC PARM içinde C ve COBOL içinde virgüllerle ayrılmış olarak aşağıdaki parametreleri alır:

1. Kuyruk yöneticisinin adı (4 karakter)
2. Hedef kuyruğun adı (48 karakter)
3. Alacak iletilerin sayısı (en çok 4 basamak)
4. İleti tarama/alma seçeneği (göz atma için 1 karakter: B ya da iletileri yıkıcı bir şekilde almak için D)
5. Syncpoint denetimi (1 karakter: Syncpoint için S ya da eşitleme noktası için N)

Bu parametrelerden herhangi birini yanlış girerseniz, uygun hata iletileri alırsınız.

Örneklerden alınan çıktı SYSPRINT veri kümesine yazılır:

```
==================================
PARAMETERS PASSED :
QMGR    - VC9
QNAME    - A.Q
NUMMSGS   - 000000002
GET     - D
SYNCPOINT  - N
==================================
MQCONN SUCCESSFUL
MQOPEN SUCCESSFUL
000000000 : 000000010 : **********
000000001 : 000000010 : **********
000000002 MESSAGES GOT FROM QUEUE
MQCLOSE SUCCESSFUL
MQDISC SUCCESSFUL
```

z/OS *z/OS üzerindeki CICS ortamı için Get örnekleri*

CICS ortamına ilişkin Get örnekleri için dikkat edilmesi gereken özel noktalar.

Hareketler, virgülle ayrılmış bir EXEC PARM içinde aşağıdaki parametreleri alır:

1. Alacak iletilerin sayısı (en çok dört basamak)
2. İletiye göz at/iletiyi al seçeneği (bir karakter: Göz atmak için B ya da iletileri yıkıcı bir şekilde almak için D)

3. Syncpoint denetimi (bir karakter: syncpoint için S ya da eşitleme noktası için N)
4. Hedef kuyruğun adı (48 karakter)

Bu parametrelerden herhangi birini yanlış girerseniz, uygun hata iletileri alırsınız.

COBOL örneği için, aşağıdakileri girerek CICS ortamında Get örneğini çağırın:

```
MVGT,9999,B,S,QUEUE.NAME
```

C örneği için, aşağıdakileri girerek CICS ortamında Get örneğini çağırın:

```
MCGT,9999,B,S,QUEUE.NAME
```

İletiler kuyruktan alındığında, CICS işlemiyle aynı ada sahip bir CICS geçici depolama kuyruğuna yerleştirilir (örneğin, C örneği için MCGT).

Örneğin, Get örnekleri:

```
**************************** TOP OF QUEUE ************************
000000000 : 000000010: **********
000000001 : 000000010 :**********
**************************** BOTTOM OF QUEUE **********************
```

Kullanım notları

- Örnekleri basit tutmak için, dil sürümleri arasında bazı küçük işlevsel farklılıklar vardır. Farkların hiçbiri MQI ile ilgili değil.
- 48 karakterden uzun bir kuyruk adı girerseniz, uzunluğu 48 karakter üst sınırına kadar kesilir, ancak hata iletisi döndürülmez.
- Hareketi girmeden önce CLEAR tuşuna basın.
- CSQ4CCJ1 karakter iletilerini yalnızca ilk NULL (\0) karakteri görüntüleninceye kadar doğru olarak görüntüleyebilir.
- Sayısal alan için, 1-9999 aralığında herhangi bir sayı girin. Girdiğiniz değer pozitif bir sayı olmalıdır. Örneğin, tek bir ileti almak için 1, 01, 001 ya da 0001 değerini girebilirsiniz. Sayısal olmayan ya da negatif bir değer girerseniz, bir hata alabilirsiniz.
- C dilinde 24 526 bayttan ve COBOL dilinde 9 950 bayttan uzun iletiler kesilir. Bunun nedeni, CICS geçici depolama kuyruklarının kullanılma şeklidir.
- Her iki program için de CSQ4CCK1 ve CSQ4CVK1, iletilere göz atmak istiyorsanız get (alma) değiştirgesine B girin; tersi durumda D girin. Bu, yıkıcı MQGET çağrıları gerçekleştirir. Başka bir değer girerseniz bir hata iletisi alırsınız.
- Her iki program için de CSQ4CCJ1 ve CSQ4CVJ1, syncpoint değiştirgesinde iletileri almak için S girin. Syncpoint değiştirgesine N girerseniz, MQGET çağrıları eşitleme noktasından çıkar. Başka bir değer girerseniz bir hata iletisi alırsınız.

z/OS *z/OS üzerindeki Göz At örneği*

Göz At örneği, MQGET çağrısı kullanılarak bir kuyruktaki iletilere nasıl göz atılacağını gösteren bir toplu iş uygulamasıdır.

Uygulama, bir kuyruktaki tüm iletilerde adım adım ilerler ve her birinin ilk 80 baytını yazdırılır. Bu uygulamayı, kuyruktaki iletileri değiştirmeden görmek için kullanabilirsiniz.

Kaynak programlar ve örnek çalıştırma JCL, COBOL, çevirici, PL/I ve C dillerinde sağlanır (bkz. Çizelge 173 sayfa 1119).

Uygulamayı başlatmak için, örnek çalıştırma JCL ' yi düzenleyin ve çalıştırın (açıklamalar için bkz. “z/OS üzerinde toplu iş ortamı için örnek uygulamaların hazırlanması ve çalıştırılması” sayfa 1118). Çalıştırdığınız JCL ' de kuyruğun adını belirterek kendi kuyruklarınızdan birindeki iletilere bakabilirsiniz.

Uygulamayı çalıştırdığınızda (ve kuyrukta bazı iletiler olduğunda), çıkış veri kümesi şu şekilde görünür:

```
07/12/1998            SAMPLE QUEUE REPORT     PAGE  1
QUEUE MANAGER NAME : VC4
QUEUE NAME : CSQ4SAMP.DEAD.QUEUE
RELATIVE
MESSAGE  MESSAGE
NUMBER  LENGTH ------------------- MESSAGE DATA -------------

1    740 HELLO. PLEASE CALL ME WHEN YOU GET BACK.
2    429 CSQ4BQRM
3    429 CSQ4BQRM
4    429 CSQ4BQRM
5    22 THIS IS A TEST MESSAGE
6     8 CSQ4TEST
7    36 CSQ4MSG - ANOTHER TEST MESSAGE.....
!8     9 CSQ4STOP
********** END OF REPORT **********
```

Kuyrukta ileti yoksa, veri kümesi yalnızca başlıkları ve `Rapor sonu` iletisini içerir. MQI çağrılarından herhangi birinde bir hata oluşursa, tamamlanma ve neden kodları çıkış veri kümesine eklenir.

z/OS *z/OS üzerinde Göz At örneğinin tasarımı*

Göz At örnek uygulaması tek bir program modülü kullanır; desteklenen programlama dillerinin her birinde bir tane sağlanır.

Program mantığından geçen akış:

1. Bir yazdırma verileri kümesi açın ve raporun başlık satırını yazdırın. Çalıştırma JCL ' den kuyruk yöneticisi ve kuyruk adlarının iletildiğini denetleyin. Her iki ad da iletildiyse, adları içeren raporun satırlarını yazdırın. Yoksa, bir hata iletisi yazdırın, yazdırma verileri kümesini kapatın ve işlemeyi durdurun.

   Programın JCL ' den geçirilen parametreleri sınama yöntemi, programın yazıldığı dile bağlıdır; daha fazla bilgi için bkz. “z/OS üzerinde dile bağımlı tasarımla ilgili önemli noktalar” sayfa 1133.
2. MQCONN çağrıyı kullanarak kuyruk yöneticisine bağlanın. Bu çağrı başarılı olmazsa, tamamlanma ve neden kodlarını yazdırın, yazdırma verileri kümesini kapatın ve işlemeyi durdurun.
3. MQOO_BROWSE seçeneğiyle MQOPEN çağrısıyla kuyruğu açın. Bu çağrıya girişte, program “2” sayfa 1132. adımda döndürülen bağlantı tanıtıcısını kullanır. Nesne tanımlayıcı yapısı (MQOD) için, kuyruk adı (adım “1” sayfa 1132 içinde geçirilen) dışındaki tüm alanlar için varsayılan değerleri kullanır. Bu çağrı başarılı olmazsa, tamamlanma ve neden kodlarını yazdırın, yazdırma verileri kümesini kapatın ve işlemeyi durdurun.
4. MQGET çağrıyı kullanarak kuyruktaki ilk iletiye göz atın. Bu çağrıya giriş sırasında, program şunları belirtir:
   - “2” sayfa 1132 ve “3” sayfa 1132 adımlarındaki bağlantı ve kuyruk tanıtıcıları
   - Tüm alanları başlangıç değerlerine ayarlanmış bir MQMD yapısı
   - İki seçenek:
     - MQGMO_BROWSE_FIRST
     - MQGMO_ACCEPT_TRUNCATED_MSG
   - İletiden kopyalanan verileri tutmak için 80 bayt büyüklüğünde bir arabellek

   MQGMO_ACCEPT_TRUNCATED_MSG seçeneği, ileti, çağrıda belirtilen 80 baytlık arabellekten daha uzun olsa da, çağrının tamamlanmasını sağlar. İleti arabellekten uzunsa, ileti arabelleğe sığacak şekilde kesilir ve tamamlanma ve neden kodları bunu gösterecek şekilde ayarlanır. Örnek, raporun okunmasını kolaylaştırmak için iletilerin 80 karaktere kısaltılması için tasarlanmıştır. Arabellek büyüklüğü bir `DEFINE` deyimiyle belirlenir; bu nedenle, isterseniz kolayca değiştirebilirsiniz.
5. MQGET çağrısı başarısız oluncaya kadar aşağıdaki döngüyü gerçekleştirin:

   a. Aşağıdakileri gösteren bir rapor satırı yazdırın:

- İletinin sıra numarası (bu, göz atma işlemlerinin sayısıdır).
- İletinin gerçek uzunluğu (kısaltılmış uzunluğu değil). Bu değer, MQGET çağrısının `DataLength` alanında döndürülür.
- İleti verilerinin ilk 80 baytı.

b. MQMD yapısının `MsqId` ve `CorrelId` alanlarını boş değere sıfırlayın

c. MQGET çağrısıyla aşağıdaki iki seçeneği kullanarak sonraki iletiye göz atın:

- MQGMO_BROWSE_NEXT
- MQGMO_ACCEPT_TRUNCATED_MSG

6. MQGET çağrısı başarısız olursa, göz atma imleci kuyruğun sonuna geldiğinden, çağrı başarısız olup olmadığını görmek için neden kodunu test edin. Bu durumda, `Raporun sonu` iletisini yazdırın ve “7” sayfa 1133 . adıma gidin; Tersi durumda, tamamlanma ve neden kodlarını yazdırın, yazdırma verileri kümesini kapatın ve işlemeyi durdurun.
7. “3” sayfa 1132. adımda döndürülen nesne tanıtıcısıyla MQCLOSE çağrısını kullanarak kuyruğu kapatın.
8. “2” sayfa 1132. adımda döndürülen bağlantı tanıtıcısıyla MQDISC çağrısını kullanarak kuyruk yöneticisiyle bağlantıyı kesin.
9. Yazdırma verileri kümesini kapatın ve işlemeyi durdurun.

z/OS *z/OS üzerinde dile bağımlı tasarımla ilgili önemli noktalar*

Göz At örneği için dört programlama dilinde kaynak modüller sağlanır.

Kaynak modüller arasında iki ana fark vardır:

- Çalıştırma JCL 'sinden geçirilen parametreleri test ederken COBOL, PL/I ve çevirici dil modülleri, virgül karakterini (,) arar. JCL `PARM=(,LOCALQ1)` değerini geçerse, uygulama varsayılan kuyruk yöneticisinde LOCALQ1 kuyruğunu açmayı dener. Virgülden sonra ad yoksa (ya da virgül yoksa), uygulama bir hata döndürür. C modülü, virgül karakterini aramaz. JCL tek bir parametre geçirirse (örneğin, `PARM=('LOCALQ1'))`, C modülü bunu varsayılan kuyruk yöneticisinde kuyruk adı olarak kullanır.
- Çevirici dil modülünü basit tutmak için, yazdırma raporunu oluştururken *yy/ddd* tarih biçimini (örneğin, 05/116) kullanır. Diğer modüller takvim tarihini *aa/gg/yy* biçiminde kullanır.

z/OS *z/OS üzerinde İleti Yazdır örneği*

İleti Yazdır örneği, MQGET çağrısını kullanarak bir kuyruktan tüm iletilerin nasıl kaldırılacağını gösteren bir toplu uygulamadır.

Print Message örneği üç parametre kullanır:

1. Kuyruk yöneticisinin adı
2. Kaynak kuyruğun adı
3. Özellikler için isteğe bağlı bir parametre

Ayrıca, her ileti için, ileti tanımlayıcısının alanlarını ve ardından ileti verilerini yazdırır. Program, verileri hem onaltılı hem de karakter olarak (yazdırılabilirse) yazdırır. Bir karakter yazdırılamıyorsa, program karakteri nokta (.) karakteriyle değiştirir. Bu programı, iletileri kuyruğa koyan bir uygulamayla ilgili sorunları tanılarken kullanabilirsiniz.

Özellik değiştirgesi için izin verilebilir değerler şunlardır:

Çizelge 185. Özellik parametresi için izin verilebilir değerler

Değer	**Davranış**
0	Varsayılan davranış. Uygulamaya teslim edilen özellikler, iletinin alındığı **`PropertyControl`** kuyruk özniteliğine bağlıdır.

Çizelge 185. Özellik parametresi için izin verilebilir değerler (devamı var)	
Değer	**Davranış**

1	Bir ileti tanıtıcısı yaratılır ve MQGET ile kullanılır. İleti tanımlayıcısında (ya da uzantıda) bulunanlar dışında, iletinin özellikleri, ileti tanımlayıcısına benzer bir şekilde görüntülenir. Örneğin:
	`****Message properties****`
	`  property name: property value`
	Ya da herhangi bir özellik yoksa:
	`****Message properties****`
	`  None`
	Sayısal değerler `printf`kullanılarak, dize değerleri tek tırnak işareti içinde, bayt dizgileri ise ileti tanımlayıcısı olarak X ve tek tırnak işareti ile çevrelenir.

2	MQGMO_NO_PROPERTIES belirtildiğinden yalnızca ileti tanımlayıcı özellikleri döndürülecek.
3	MQGMO_PROPERTIES_FORCE_MQRFH2 , ileti verilerinde tüm özelliklerin döndürülmesi için belirtilir.
4	MQGMO_PROPERTIES_COMPATIBILITY belirtildiğinden, IBM MQ özelliğinin içerilip içerilmediğine bağlı olarak tüm özellikler döndürülebilir, tersi durumda özellikler atılır.

Uygulamayı, iletileri kuyruktan kaldırmak yerine, iletilere göz atmak için değiştirebilirsiniz. Bunu yapmak için, -DBROWSE seçeneğiyle derleyin ve “z/OS üzerinde Yazdırma İletisi örneğinin tasarımı” sayfa 1135içinde gösterildiği gibi BROWSE makrosunu tanımlayın. Yürütülür dosya kodu, SCSQLOAD kitaplığında sizin için sağlanır. CSQ4BCG0 modülü -DBROWSE; CSQ4BCG1 modülüyle oluşturuldu ve kuyruğu yıkıcı bir şekilde okuyor.

Uygulamanın C dilinde yazılmış tek bir kaynak programı vardır. Örnek çalıştırma JCL kodu da sağlanır (bkz. Çizelge 174 sayfa 1119).

Uygulamayı başlatmak için, örnek çalıştırma JCL ' yi düzenleyin ve çalıştırın (açıklamalar için bkz. “z/OS üzerinde toplu iş ortamı için örnek uygulamaların hazırlanması ve çalıştırılması” sayfa 1118). Uygulamayı çalıştırdığınızda (ve kuyrukta bazı iletiler olduğunda), çıkış veri kümesi Şekil 139 sayfa 1135içinde bu şekilde görünür.

```
CSQ4BCG1 - starts here
**********************

MQCONN to MQ1E
MQOPEN - 'TEST.QUEUE'
MQCRTMH


MQGET of message number 1
****Message descriptor****
StrucId : 'MD ' Version : 2
Report  : 0 MsgType : 8
Expiry  : -1 Feedback : 0
Encoding : 785 CodedCharSetId : 500
Format : 'MQSTR   '
Priority : 0 Persistence : 0
MsgId : X'C3E2D840D4D8F1C540404040404040C1EA537F03167D88'
CorrelId : X'C3E2D840D4D8F1C540404040404040C1EA537F0317A928'
BackoutCount : 0
ReplyToQ    : '                          '
ReplyToQMgr  : ''
** Identity Context
UserIdentifier : 'FRED    '
AccountingToken :
X'0000000000000000000000000000000000000000000000000000000000000000'
ApplIdentityData : '                 '
** Origin Context
PutApplType  : '2'
PutApplName  : 'FRED6            '
PutDate : '20080207'  PutTime : '17373745'
ApplOriginData : '   '
GroupId : X'000000000000000000000000000000000000000000000000'
MsgSeqNumber  : '1'
Offset     : '0'
MsgFlags    : '0'
OriginalLength : '-1'

****Message properties****
None

****  Message   ****

length - 30 bytes

00000000: E388 89A2 4089 A240 8140 A289 9497 9385 'This is a simple'
00000010: 40A3 85A2 A340 9485 A2A2 8187 855A   ' test message! '



No more messages
MQDLTMH
MQCLOSE
MQDISC
```

Şekil 139. İleti Yazdır örnek uygulamasından bir rapor örneği

z/OS *z/OS üzerinde Yazdırma İletisi örneğinin tasarımı*

Print Message örnek uygulaması, C dilinde yazılmış tek bir program kullanır.

Program mantığından geçen akış:

1. Çalıştırma JCL ' den kuyruk yöneticisi ve kuyruk adlarının iletildiğini denetleyin. Yoksa, bir hata iletisi yazdırın ve işlemeyi durdurun.
2. MQCONN çağrıyı kullanarak kuyruk yöneticisine bağlanın. Bu çağrı başarılı olmazsa, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun; tersi durumda, kuyruk yöneticisinin adını yazdırın.
3. MQOO_INPUT_SHARED seçeneğiyle MQOPEN çağrısıyla kuyruğu açın.

   Not: Uygulamanın iletilere kuyruktan kaldırmak yerine göz atmasını istiyorsanız, örneği -DBROWSE ile derleyin ya da kaynağın başına `#define BROWSE` ekleyin. Bunu yaptığınızda, makro ön işlemcisi derlemede MQOO_BROWSE seçeneğini seçen satırı programa ekler.

   Bu çağrıya girişte, program “2” sayfa 1135. adımda döndürülen bağlantı tanıtıcısını kullanır. Nesne tanımlayıcı yapısı (MQOD) için, kuyruk adı (adım “1” sayfa 1135 içinde geçirilen) dışındaki tüm alanlar

için varsayılan değerleri kullanır. Bu çağrı başarılı olmazsa, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun; tersi durumda, kuyruğun adını yazdırın.

4. İleti özelliklerini almak için bir ileti tanıtıcısı kullanırsanız, sonraki MQGET çağrılarıyla kullanılmak üzere böyle bir tanıtıcı yaratmak için MQCRTMH ' yi kullanın. Bu çağrı başarılı olmazsa, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun.
5. İleti alma seçeneklerini, herhangi bir ileti özelliği için istek işlemini yansıtacak şekilde ayarlayın.
6. MQGET çağrısı başarısız oluncaya kadar aşağıdaki döngüyü gerçekleştirin:
   a. Arabelleği, ileti verilerinin arabellekteki herhangi bir veri tarafından bozulmaması için boş olacak şekilde kullanıma hazırlayın.
   b. MQMD yapısının `MsgId` ve `CorrelId` alanlarını boş değere ayarlayın; böylece MQGET çağrısı kuyruktan ilk iletiyi seçer.
   c. MQGET çağrıyı kullanarak kuyruktan bir ileti alın. Bu çağrıya giriş sırasında, program şunları belirtir:
      - "2" sayfa 1135 ve "3" sayfa 1135adımlarından bağlantı ve nesne işler.
      - Tüm alanları başlangıç değerlerine ayarlanmış bir MQMD yapısı. (`MsgId` ve `CorrelId` , her MQGET çağrısı için boş değere sıfırlanır.)
      - MQGMO_NO_WAIT seçeneği.

        Not: Uygulamanın iletilere kuyruktan kaldırmak yerine göz atmasını istiyorsanız, örneği -DBROWSE ile derleyin ya da kaynağın başına `#define BROWSE` ekleyin. Bunu yaptığınızda, makro ön işlemcisi, derlemeye MQGMO_BROWSE_NEXT seçeneğini seçen programı ekler. Bu seçenek, yürürlükteki nesne tanıtıcısıyla daha önce göz atma imleci kullanılmamış bir kuyruğa yönelik çağrıda kullanıldığında, göz atma imleci mantıksal olarak ilk iletiden önce konumlandırılır.
      - İletiden kopyalanan verileri tutmak için 64KB büyüklüğünde bir arabellek.
   d. printMD alt yordamını çağırın. Bu, ileti tanımlayıcısındaki her alanın adını ve ardından içeriğini yazdırır.
   e. "4" sayfa 1136 . adımda bir ileti tanıtıcısı yarattıysanız, ileti özelliklerini görüntülemek için printProperties alt yordamını çağırın.
   f. İletinin uzunluğunu ve ardından ileti verilerini yazdırın. İleti verilerinin her satırı şu biçimdedir:
      - Verilerin bu bölümünün göreli konumu (onaltılı olarak)
      - 16 baytlık onaltılı veri
      - Yazdırılabilirse, karakter biçiminde aynı 16 bayt veri (yazdırılamayan karakterler nokta imiyle değiştirilir)
7. MQGET çağrısı başarısız olursa, kuyrukta başka ileti olmadığı için aramanın başarısız olup olmadığını görmek için neden kodunu test edin. Bu durumda, şu iletiyi yazdırın: Başka ileti yok; tersi durumda, tamamlanma ve neden kodlarını yazdırın. Her iki durumda da "9" sayfa 1136. adıma gidin.

   Not: MQGET çağrısı, 64KB ' den fazla veri içeren bir ileti bulursa başarısız olur. Programı daha büyük iletileri işleyecek şekilde değiştirmek için aşağıdakilerden birini yapabilirsiniz:

   - MQGMO_ACCEPT_TRUNCATED_MSG seçeneğini MQGET çağrısına ekleyin, böylece çağrı ilk 64KB veri alır ve geri kalan verileri atar
   - Bu miktarda veri içeren bir ileti bulunduğunda, programın iletiyi kuyrukta bırakmasını sağlayın
   - Arabelleğin büyüklüğünü artırın
8. "4" sayfa 1136 . adımda bir ileti tanıtıcısı yarattıysanız, bunu silmek için MQDLTMH ' yi çağırın.
9. "3" sayfa 1135. adımda döndürülen nesne tanıtıcısıyla MQCLOSE çağrısını kullanarak kuyruğu kapatın.
10. "2" sayfa 1135. adımda döndürülen bağlantı tanıtıcısıyla MQDISC çağrısını kullanarak kuyruk yöneticisiyle bağlantıyı kesin.

z/OS *z/OS üzerinde Kuyruk Öznitelikleri örneği*

Kuyruk Öznitelikleri örneği, MQINQ ve MQSET çağrılarının kullanımını gösteren etkileşimli kipli bir CICS uygulamasıdır.

Kuyrukların **InhibitPut** ve **InhibitGet** özniteliklerinin değerlerini nasıl sorgulayacağınızı ve bu değerlerin nasıl değiştirileceğini gösterir; böylece, programlar bir kuyruğa ileti koyamaz ya da kuyruktan ileti alamaz. Bir programı sınarken bu şekilde bir kuyruğu *kilitlemek* isteyebilirsiniz.

Kendi kuyruklarınızla yanlışlıkla etkileşimde bulunmayı önlemek için, bu örnek yalnızca adının ilk sekiz baytında CSQ4SAMP karakterleri bulunan bir kuyruk nesnesinde çalışır. Ancak kaynak kod, bu kısıtlamayı nasıl kaldıracağınızı gösteren açıklamalar içerir.

Kaynak programlar COBOL, çevirici ve C dillerinde sağlanır (bkz. Çizelge 180 sayfa 1124).

Örneğin çevirici dili sürümü yeniden girilebilir kod kullanır. Bunu yapmak için, örneğin bu sürümündeki her MQI çağrısına ilişkin kodun MF anahtar sözcüğünü içerdiğini fark edeceksiniz; örneğin:

```
CALL MQCONN,(NAME,HCONN,COMPCODE,REASON),MF=(E,PARMAREA),VL
```

(VL anahtar sözcüğü, programda hata ayıklamak için CICS Execution Diagnostic Facility (CEDF) tarafından sağlanan hareketi kullanabileceğiniz anlamına gelir.) Yeniden girilebilir programlar yazmaya ilişkin ek bilgi için bkz. System/390 çevirici dilinde kodlama.

Uygulamayı başlatmak için CICS sisteminizi başlatın ve aşağıdaki CICS hareketlerini kullanın:

- COBOL için MVC1
- Çevirici dili için, MAC1
- C için, MCC1

3. adımda belirtilen CSD veri kümesini değiştirerek bu hareketlerden herhangi birinin adını değiştirebilirsiniz.

Örneğin tasarımı

Örneği başlattığınızda, aşağıdakilere ilişkin alanları içeren bir ekran eşlemi görüntülenir:

- Kuyruğun adı
- Kullanıcı isteği (geçerli işlemler şunlardır: insorire, allow ya da inhibe)
- Kuyruğa ilişkin koyma işlemlerinin yürürlükteki durumu
- Kuyruğa ilişkin alma işlemlerinin yürürlükteki durumu

İlk iki alan kullanıcı girişi içindir. Son iki alanı uygulama doldurur: `INHIBITED` ya da `ALLOWED`sözcüğünü gösterir.

Uygulama, ilk iki alana girdiğiniz değerleri doğrular. Kuyruk adının CSQ4SAMP karakterleriyle başlayıp başlamadığını ve İşlem alanına üç geçerli istekten birini girdiğinizi denetler. Uygulama, tüm girişinizi büyük harfe çevirdiğinden, küçük harfli karakterler içeren adlara sahip kuyruklar kullanamazsınız.

İşlem alanına `inquire` girerseniz, program mantığından geçen akış şöyledir:

1. MQOO_INQUIRE seçeneğiyle MQOPEN çağrısıyla kuyruğu açın
2. MQIA_INHIBIT_GET ve MQIA_INHIBIT_PUT seçicilerini kullanarak MQINQ ' yu çağırın
3. MQCLOSE çağrıyı kullanarak kuyruğu kapat
4. MQINQ çağrısının **IntAttrs** değiştirgesinde döndürülen öznitelikleri çözümleyin ve INALLOWED ya da ALLOWED sözcüklerini ilgili ekran alanlarına taşıyın

İşlem alanına `inhibit` girerseniz, program mantığından geçen akış şöyledir:

1. MQOO_SET seçeneğiyle MQOPEN çağrısıyla kuyruğu açın
2. MQIA_INHIBIT_GET ve MQIA_INHIBIT_PUT seçicilerini kullanarak ve **IntAttrs** değiştirgesinde MQQA_GET_INHIBIT_VE MQQA_PUT_INEDINDIRME_DEĞERLERINI kullanarak MQSET ' i çağırın

3. MQCLOSE çağrıyı kullanarak kuyruğu kapat
4. INÇEKINGEN sözcüğünü ilgili ekran alanlarına taşı

İşlem alanına `allow` girerseniz, uygulama, engelleme isteği için buna benzer işlemler gerçekleştirir. Tek fark, özniteliklerin ve ekranda görüntülenen sözcüklerin ayarlarıdır.

Uygulama kuyruğu açtığında, kuyruk yöneticisine yönelik varsayılan bağlantı tanıtıcısını kullanır. (CICS , CICS sisteminizi başlattığınızda kuyruk yöneticisiyle bağlantı kurar.) Uygulama bu aşamada aşağıdaki hataları tuzağa düşürebilir:

- Uygulama kuyruk yöneticisine bağlı değil
- Kuyruk yok
- Kullanıcının kuyruğa erişme yetkisi yok
- Uygulamanın kuyruğu açma yetkisi yok

Diğer MQI hataları için uygulama, tamamlanma ve neden kodlarını görüntüler.

z/OS *z/OS üzerindeki Mail Manager örneği*

Mail Manager örnek uygulaması, hem tek bir ortamda hem de farklı ortamlarda ileti göndermeyi ve almayı gösteren bir program grubudur. Uygulama, kullanıcıların farklı kuyruk yöneticileri kullansalar bile ileti alışverişinde bulunmalarını sağlayan basit bir elektronik posta sistemidir.

Uygulama, MQOPEN çağrısıyla ve IBM MQ for z/OS komutlarını sistem komutu giriş kuyruğuna koyarak kuyrukların nasıl yaratılacağını gösterir.

Uygulamanın üç sürümü sağlanır:

- COBOL dilinde yazılmış bir CICS uygulaması
- COBOL dilinde yazılmış bir TSO uygulaması
- C dilinde yazılmış bir TSO uygulaması

z/OS *z/OS üzerinde Mail Manager örneğini hazırlama*

Mail Manager, iki ortamda çalışan sürümlerde sağlanır. Uygulamayı çalıştırmadan önce gerçekleştirmeniz gereken hazırlık, kullanmak istediğiniz ortama bağlıdır.

Kullanıcılar, oturum açma kullanıcı kimlikleri her sistemde aynı olduğu sürece, posta kuyruklarına ve takma ad kuyruklarına hem TSO hem de CICS içinden erişebilirler.

Başka bir kuyruk yöneticisine ileti göndermeden önce, o kuyruk yöneticisine bir ileti kanalı ayarlamanız gerekir. Bunu yapmak için, Kanal denetimi işlevindeaçıklanan IBM MQkanal denetimi işlevini kullanın.

Örnek TSO ortamı için hazırlanıyor

Aşağıdaki adımları izleyin:

1. Örneği, “z/OS üzerinde TSO ortamı için örnek uygulamaların hazırlanması” sayfa 1121içinde açıklandığı gibi hazırlayın.
2. Örneğin tanımlaması için sağlanan CLIST ' i uyarla:
   - Panoların konumu
   - İleti dosyasının yeri
   - Yükleme modüllerinin konumu
   - Uygulamayla kullanmak istediğiniz kuyruk yöneticisinin adı

   Örneğin her dil sürümü için ayrı bir CLIST sağlanır:
   - COBOL sürümü için: CSQ4RVD1
   - C sürümü için: CSQ4RCD1

3. Uygulama tarafından kullanılan kuyrukların kuyruk yöneticisinde var olduğunu doğrulayın. (Kuyruklar CSQ4CVDiçinde tanımlanır.)

Not: VS COBOL II, ISPFile çoklu görevi desteklemez. Bu, bölünmüş ekranın her iki tarafında Posta Yöneticisi örnek uygulamasını kullanamayacağınız anlamına gelir. Eğer yaparsanız, sonuçlar önceden kestirilemez.

z/OS *z/OS üzerinde Mail Manager örneğini çalıştırma*

Örneği CICS Transaction Server for z/OS ortamında başlatmak için MAIL işlemini çalıştırın. CICS' da henüz oturum açmadıysanız, uygulama sizden postanızı gönderebileceği bir kullanıcı kimliği girmenizi ister.

Uygulamayı başlattığınızda, posta kuyruğunuzu açar. Bu kuyruk yoksa, uygulama sizin için bir kuyruk yaratır. Posta kuyruklarının adları CSQ4SAMP.MAILMGR. *userid*; burada *userid* ortama bağlıdır:

TSO ' da
Kullanıcının TSO Kimliği

İçindeCICS
Kullanıcının CICS oturum açma kimliği ya da Posta Yöneticisi başladığında kullanıcı tarafından girilen kullanıcı kimliği

Posta Yöneticisi 'nin kullandığı kuyruk adlarının tüm bölümleri büyük harfli olmalıdır.

Daha sonra uygulama, aşağıdakilere ilişkin seçenekleri olan bir menü panosu sunar:

- Gelen postayı oku
- Posta Gönder
- Takma ad yarat

Menü panosu, posta kuyruğunuzda kaç ileti beklediğini de gösterir. Menü seçeneklerinin her biri başka bir pano görüntüler:

Gelen postayı oku
Posta Yöneticisi, posta kuyruğunuzda bulunan iletilerin bir listesini görüntüler. (Yalnızca kuyruktaki ilk 99 ileti görüntülenir.) Bu panonun bir örneği için bkz. Şekil 142 sayfa 1143. Bu listeden bir ileti seçtiğinizde, iletinin içeriği görüntülenir (bkz. Şekil 143 sayfa 1144).

Posta Gönder
Bir pano girmenizi ister:

- İleti göndermek istediğiniz kullanıcının adı
- Posta kuyruğunun sahibi olan kuyruk yöneticisinin adı
- İletinizin metni

Kullanıcı adı alanında, Posta Yöneticisi 'ni kullanarak oluşturduğunuz bir kullanıcı kimliği ya da takma ad girebilirsiniz. Kullanıcının posta kuyruğu kullandığınız aynı kuyruk yöneticisine aitse, kuyruk yöneticisi adı alanını boş bırakabilirsiniz ve kullanıcı adı alanına bir takma ad girdiyseniz bu alanı boş bırakmanız gerekir:

- Yalnızca bir kullanıcı adı belirlerseniz, program önce adın bir takma ad olduğunu varsayar ve iletiyi o adla tanımlanan nesneye gönderir. Böyle bir takma ad yoksa, program iletiyi o adı taşıyan yerel bir kuyruğa göndermeyi dener.
- Hem bir kullanıcı adı, hem de bir kuyruk yöneticisi adı belirlerseniz, program iletiyi bu iki adla tanımlanan posta kuyruğuna gönderir.

Örneğin, QM12uzak kuyruk yöneticisindeki JONESM kullanıcısına bir ileti göndermek isterseniz, aşağıdaki iki yoldan biriyle ileti gönderebilirsiniz:

- QM12kuyruk yöneticisinde kullanıcı JONESM 'si belirtmek için her iki alanı da kullanın.
- Bu kullanıcı için bir takma ad (örneğin, MARY) tanımlayın ve kullanıcı adı alanına MARY koyarak ve kuyruk yöneticisi adı alanında hiçbir şey yazarak bir ileti gönderin.

Takma ad yarat

Sık sık iletişim kuracağınız başka bir kullanıcıya ileti gönderirken kullanabileceğiniz, hatırlaması kolay bir ad tanımlayabilirsiniz. Diğer kullanıcının kullanıcı kimliğini ve posta kuyruğunun sahibi olan kuyruk yöneticisinin adını girmeniz istenir.

Takma adlar, CSQ4SAMP.MAILMGR. *userid.nickname*; burada *userid* kendi kullanıcı kimliğiniz ve *nickname* kullanmak istediğiniz takma addır. Bu şekilde yapılandırılmış adlarla, kullanıcıların her birinin kendi takma adları olabilir.

Programın yarattığı kuyruk tipi, Takma Ad Yarat panosundaki alanları nasıl dolduracağınıza bağlıdır:

- Yalnızca bir kullanıcı adı belirtirseniz ya da kuyruk yöneticisi adı, Posta Yöneticisi 'nin bağlı olduğu kuyruk yöneticisiyle aynıysa, program bir diğer ad kuyruğu yaratır.
- Hem kullanıcı adı, hem de kuyruk yöneticisi adı belirlerseniz (ve kuyruk yöneticisi Posta Yöneticisi 'nin bağlı olduğu kuyruk yöneticisi değilse), program uzak bir kuyruk için yerel bir tanım yaratır. Program, bu tanımlamanın çözüldüğü kuyruğun varlığını ya da uzak kuyruk yöneticisinin var olup olmadığını denetlemez.

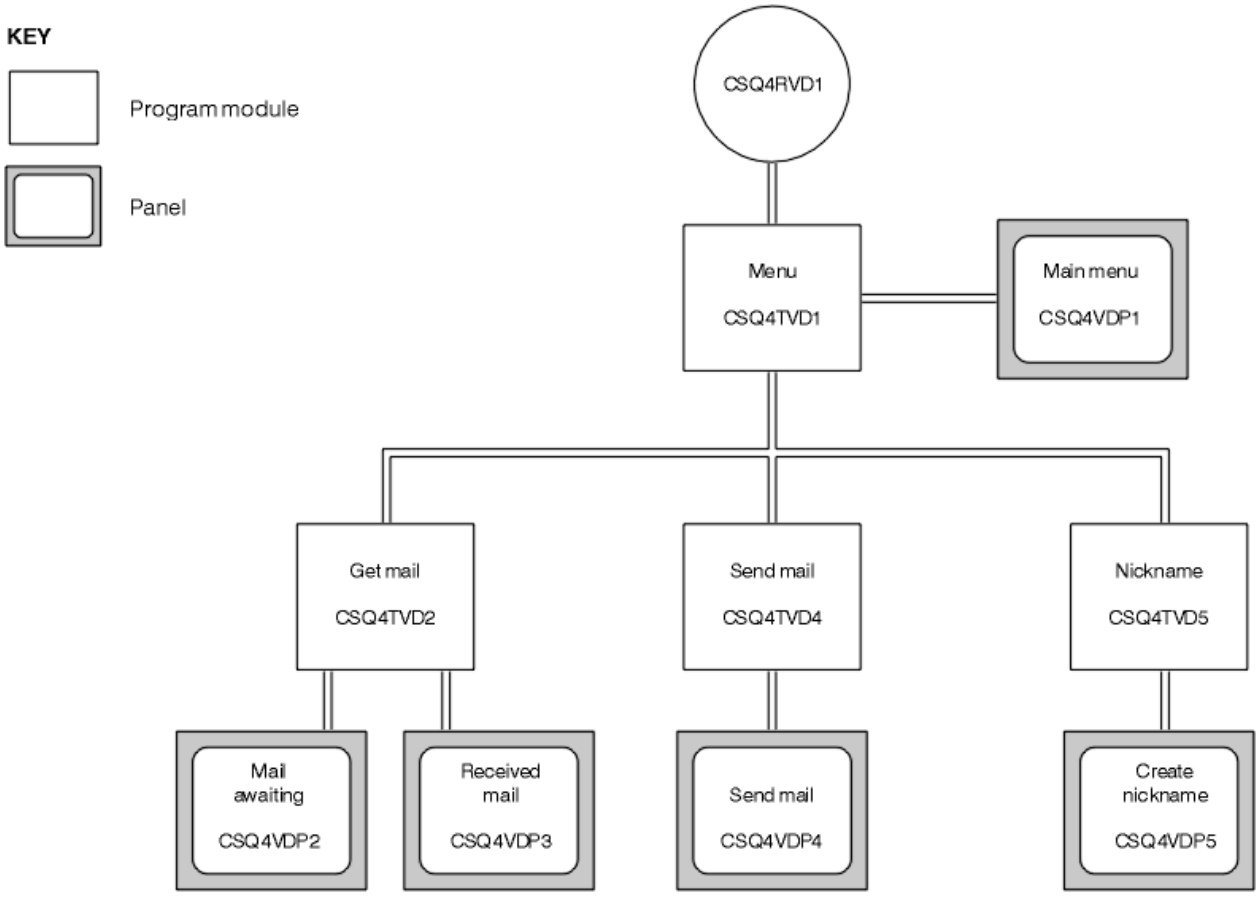
Örneğin, kendi kullanıcı kimliğiniz SMITHK ise ve kullanıcı JONESM (uzak kuyruk yöneticisi QM12' yi kullanan) için MARY adlı bir takma ad yaratırsanız, takma ad programı CSQ4SAMP.MAILMGR.SMITHK.MARY. Bu tanımlama, Mary 'nin kuyruk yöneticisindeki CSQ4SAMP.MAILMGR.JONESM posta kuyruğuna QM12çözülür. QM12 kuyruk yöneticisini kendiniz kullanıyorsanız, program aynı ada (CSQ4SAMP.MAILMGR.SMITHK.MARY).

TSO uygulamasının C sürümü, ISPF' nin ileti işleme yeteneklerinden COBOL sürümünden daha fazla yararlanır. C ve COBOL sürümlerinde farklı hata iletilerinin görüntülendiğini fark etmiş olabilirsiniz.

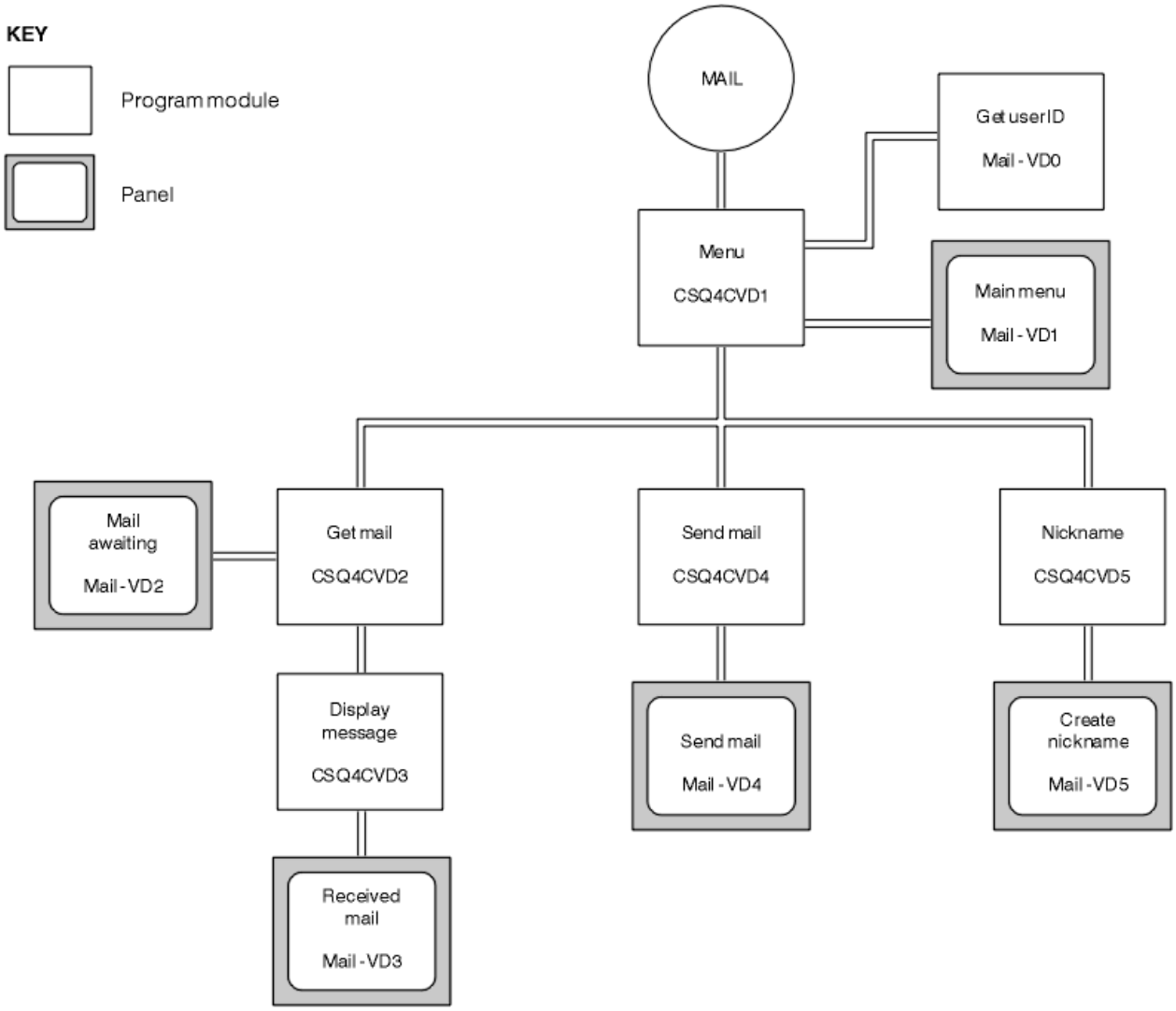
z/OS *z/OS üzerindeki Mail Manager örneğinin tasarımı*

Aşağıdaki bölümlerde, Mail Manager örnek uygulamasını oluşturan programların her biri açıklanmaktadır.

Programlarla uygulamanın kullandığı panolar arasındaki ilişkiler, TSO sürümü için Şekil 140 sayfa 1141 ve CICS Transaction Server for z/OS sürümü için Şekil 141 sayfa 1142 içinde gösterilir.



Şekil 140. Mail Manager 'ın TSO sürümlerine ilişkin programlar ve panolar



Şekil 141. Mail Manager 'ın CICS sürümüne ilişkin programlar ve panolar

z/OS *z/OS üzerindeki menü programı*

TSO ortamında, menü programı CLIST tarafından çağrılır. CICS ortamında, program MAIL işlemi tarafından başlatılır.

Menü programı (TSO içinCSQ4TVD1 , CICSiçin CSQ4CVD1), ürün grubundaki ilk programdır. Menüde (TSO içinCSQ4VDP1 , CICSiçin VD1) görüntülenir ve menüden seçildiklerinde diğer programlar çağrılır.

Program önce kullanıcının kimliğini alır:

- Programın CICS sürümünde kullanıcı CICS' ta oturum açmışsa, kullanıcı kimliği CICS ASSIGN USERID komutu kullanılarak elde edilir. Kullanıcı oturum açmamışsa, kullanıcıdan kullanıcı kimliği girmesini isteyen oturum açma panosunu (CSQ4VD0) görüntüler. Bu program içinde güvenlik işlemesi yoktur; kullanıcı herhangi bir kullanıcı kimliği verebilir.
- TSO sürümünde, kullanıcı kimliği CLIST içindeki TSO ' dan alınır. Menü programına, ISPF paylaşılan havuzunda bir değişken olarak geçirilir.

Program kullanıcı kimliğini aldıktan sonra, kullanıcının bir posta kuyruğu (CSQ4SAMP.MAILMGR. *kullanıcı kimliği*). Bir posta kuyruğu yoksa, program, sistem komutu giriş kuyruğuna bir ileti koyarak bir ileti yaratır. İleti, IBM MQ for z/OS DEFINE QLOCAL komutunu içerir. Bu komutun kullandığı nesne tanımlaması, kuyruğun derinlik üst sınırını 9999 iletiye ayarlar.

Program, sistem komutu giriş kuyruğundan gelen yanıtları işlemek için geçici bir dinamik kuyruk da yaratır. Bunu yapmak için, program SYSTEM.DEFAULT.MODEL.QUEUE . Kuyruk yöneticisi, geçici dinamik kuyruğu CSQ4SAMP; önekine sahip bir adla yaratır; adın geri kalan kısmı kuyruk yöneticisi tarafından üretilir.

Program daha sonra kullanıcının posta kuyruğunu açar ve kuyruktaki ileti sayısını, kuyruğun yürürlükteki derinliğini sorgulayarak bulur. Bunu yapmak için program, MQIA_CURRENT_Q_DEPTH seçicisini belirterek MQINQ çağrısını kullanır.

Program daha sonra menüyü görüntüleyen bir döngü gerçekleştirir ve kullanıcının yaptığı seçimi işler. Döngü, kullanıcı PF3 tuşuna bastığında durdurulur. Geçerli bir seçim yapıldığında, uygun program başlatılır; tersi durumda bir hata iletisi görüntülenir.

z/OS *z/OS üzerinde get-mail (posta) ve display-message (ileti görüntüleme) programları*

Uygulamanın TSO sürümlerinde, get-mail ve display-message işlevleri aynı program (CSQ4TVD2) tarafından gerçekleştirilir. Uygulamanın CICS sürümünde bu işlevler ayrı programlar (CSQ4CVD2 ve CSQ4CVD3) tarafından gerçekleştirilir.

Mail ABekleme panosu (TSO içinCSQ4VDP2 , CICS için VD2 ; Örneğin, Şekil 142 sayfa 1143 başlıklı konuya bakın) kullanıcının posta kuyruğundaki tüm iletileri gösterir. Bu listeyi yaratmak için, program, kuyruktaki tüm iletilere göz atmak ve her bir iletiye ilişkin bilgileri saklamak için MQGET çağrıyı kullanır. Program, görüntülenen bilgilere ek olarak, her iletinin `MsgId` ve `CorrelId` bilgilerini de kaydeder.

```
---------------------- IBM MQ for z/OS Sample Programs ------- ROW 16 OF 29
COMMAND ==>                          Scroll  ===> PAGE
USERID - NTSFV02
Mail Manager System      QMGR  - VC4
Mail Awaiting

Msg   Mail    Date    Time
No    From    Sent    Sent
16
16   Deleted
17   JOHNJ   01/06/1993 12:52:02
18   JOHNJ   01/06/1993 12:52:02
19   JOHNJ   01/06/1993 12:52:03
20   JOHNJ   01/06/1993 12:52:03
21   JOHNJ   01/06/1993 12:52:03
22   JOHNJ   01/06/1993 12:52:04
23   JOHNJ   01/06/1993 12:52:04
24   JOHNJ   01/06/1993 12:52:04
25   JOHNJ   01/06/1993 12:52:05
26   JOHNJ   01/06/1993 12:52:05
27   JOHNJ   01/06/1993 12:52:05
28   JOHNJ   01/06/1993 12:52:06
29   JOHNJ   01/06/1993 12:52:06
```

Şekil 142. Bekleyen iletilerin listesini gösteren pano örneği

Posta Bekleme panosundan kullanıcı bir ileti seçebilir ve iletinin içeriğini görüntüleyebilir (örnek için bkz. Şekil 143 sayfa 1144). Program, tüm iletilere göz attığında programın belirttiği `MsgId` ve `CorrelId` öğelerini kullanarak bu iletiyi kuyruktan kaldırmak için MQGET çağrısını kullanır. Bu MQGET çağrısı MQGMO_SYNCPOINT seçeneği kullanılarak gerçekleştirilir. Program iletinin içeriğini görüntüler ve bir eşitleme noktası bildirir: Bu, MQGET çağrısını kesinleştirir, bu nedenle ileti artık yok.

```
--------------------- IBM MQ for z/OS Sample Programs ---------------------
COMMAND ==>
USERID - NTSFV02
Mail Manager System      QMGR  - VC4
Received Mail

Mail sent from JOHNJ  at VC4

Sent on the 01/06/1993 at 12:52:02
----------------------------------- Message ------------------------------
| HELLO FROM JOHNJ                                |
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
|                                     |
'-------------------------------------------------------------------------'
```

Şekil 143. Bir iletinin içeriğini gösteren pano örneği

Posta Yöneticisi tarafından sağlanan işlevin belirgin bir uzantısı, kullanıcıya, içeriğini görüntüledikten sonra iletiyi kuyrukta bırakma seçeneği vermektir. Bunu yapmak için, iletiyi görüntüledikten sonra kuyruktan kaldıran MQGET çağrısından geri çekilmeniz gerekir.

z/OS *z/OS üzerinde posta gönderme programı*

Kullanıcı Posta Gönder panosunu tamamladığında (TSO içinCSQ4VDP4 , CICS için VD4), posta gönderme programını (TSO içinCSQ4TVD4 , CICS için CSQ4CVD4) iletiyi alıcının posta kuyruğuna koyar.

Bunu yapmak için, program MQPUT1 çağrıyı kullanır. İletinin hedefi, kullanıcının Posta Gönder panosunda alanları nasıl doldurduğuna bağlıdır:

- Kullanıcı yalnızca bir kullanıcı adı belirlediyse, program önce adın bir takma ad olduğunu varsayar ve iletiyi o adla tanımlanan nesneye gönderir. Böyle bir takma ad yoksa, program iletiyi o adı taşıyan yerel bir kuyruğa göndermeyi dener.
- Kullanıcı hem bir kullanıcı adı hem de bir kuyruk yöneticisi adı belirlediyse, program iletiyi bu iki adla tanımlanan posta kuyruğuna gönderir.

Program boş iletileri kabul etmez ve ileti metninin her satırındaki baştaki boşlukları kaldırır.

MQPUT1 çağrısı başarılı olursa, program iletinin konduğu kullanıcı adı ve kuyruk yöneticisi adını gösteren bir ileti görüntüler. Çağrı başarısız olursa, program özellikle kuyruğun ya da kuyruk yöneticisinin var olmadığını belirten neden kodlarını denetler; bunlar MQRC_UNKNOWN_OBJECT_NAME ve MQRC_UNKNOWN_OBJECT_Q_MGR. Program, bu hataların her biri için kendi hata iletisini görüntüler; diğer hatalar için, çağrı tarafından döndürülen tamamlanma ve neden kodlarını görüntüler.

z/OS *z/OS üzerinde takma ad programı*

Kullanıcı bir takma ad tanımladığında, takma ad programı (TSO içinCSQ4TVD5 , CICS için CSQ4CVD5) Takma adı adının bir parçası olarak içeren bir kuyruk yaratır.

Program bunu, sistem komutu giriş kuyruğuna bir ileti koyarak yapar. İleti, IBM MQ for z/OS DEFINE QALIAS ya da DEFINE QREMOTE komutunu içerir. Programın yarattığı kuyruk tipi, kullanıcının Create Nickname (takma ad yarat) panosundaki alanları nasıl doldurduğuna bağlıdır (TSO içinCSQ4VDP5 , CICS için VD5):

- Kullanıcı yalnızca bir kullanıcı adı belirttiyse ya da kuyruk yöneticisi adı, Posta Yöneticisi 'nin bağlı olduğu kuyruk yöneticisiyle aynıysa, program bir diğer ad kuyruğu oluşturur.
- Kullanıcı hem bir kullanıcı adı, hem de bir kuyruk yöneticisi adı belirlediyse (ve kuyruk yöneticisi, Posta Yöneticisi 'nin bağlı olduğu kuyruk yöneticisi değilse), program uzak bir kuyruk için yerel bir tanımlama

yaratır. Program, bu tanımlamanın çözüldüğü kuyruğun varlığını ya da uzak kuyruk yöneticisinin var olup olmadığını denetlemez.

Program, sistem komutu giriş kuyruğundan gelen yanıtları işlemek için geçici bir dinamik kuyruk da yaratır.

Kuyruk yöneticisi takma ad kuyruğunu programın beklediği bir nedenle yaratamıyorsa (örneğin, kuyruk zaten var), program kendi hata iletisini görüntüler. Kuyruk yöneticisi, programın beklemediği bir nedenle kuyruğu yaratamadıysa, program, komut sunucusu tarafından programa döndürülen hata iletilerinden en çok ikisini görüntüler.

Not: Her takma ad için, takma ad programı yalnızca bir diğer ad kuyruğu ya da uzak kuyruk için yerel bir tanımlama yaratır. Bu kuyruk adlarının çözüldüğü yerel kuyruklar, yalnızca takma addaki kullanıcı kimliği Posta Yöneticisi uygulamasını başlatmak için kullanıldığında oluşturulur.

z/OS *z/OS üzerindeki Kredi Denetimi örneği*

Credit Check örnek uygulaması, IBM MQ for z/OStarafından sağlanan özelliklerin çoğunun nasıl kullanılacağını gösteren bir program grubudur. Bir uygulamanın birçok bileşen programının ileti kuyruklama tekniklerini kullanarak iletileri birbirine nasıl iletebileceğini gösterir.

Örnek, bağımsız bir CICS uygulaması olarak çalışabilir. Ancak, hem CICS hem de IMS ortamları tarafından sağlanan olanakları kullanan bir ileti kuyruklama uygulamasının nasıl tasarlanılacağını göstermek için, bir modül IMS toplu ileti işleme programı olarak da sağlanır. Örneğin bu uzantısı “z/OS üzerindeki Kredi Denetimi örneğinin IMS uzantısı” sayfa 1154içinde açıklanmıştır.

Örneği birden çok kuyruk yöneticisinde çalıştırabilir ve uygulamanın her eşgörünümü arasında ileti gönderebilirsiniz. Bunu yapmak için bkz. “z/OS üzerinde birden çok kuyruk yöneticisiyle Kredi Denetimi örneği” sayfa 1154.

CICS programları C ve COBOL dilinde sağlanır. Tek IMS programı yalnızca C ' de teslim edilir. Sağlanan veri kümeleri Çizelge 182 sayfa 1125 ve Çizelge 184 sayfa 1126içinde gösterilir.

Uygulama, banka müşterileri kredi istediklerinde riski değerlendirme yöntemini gösterir. Başvuru, bir bankanın kredi isteklerini işlemek için iki şekilde nasıl çalışabileceğini gösterir:

- Banka personeli, doğrudan bir müşteriyle ilgilenirken, hesap ve kredi riski bilgilerine anında erişim ister.
- Banka personeli, yazılı uygulamalarla ilgilenirken, hesap ve kredi riski bilgileri için bir dizi istek gönderebilir ve yanıtlarla daha sonra ilgilenebilir.

Uygulamadaki finansal ve güvenlik ayrıntıları, ileti kuyruklama tekniklerinin net olması için basit tutuldu.

z/OS *z/OS üzerinde Kredi Denetimi örneğinin hazırlanması ve çalıştırılması*

Credit Check örneğini hazırlamak ve çalıştırmak için aşağıdaki adımları gerçekleştirin:

1. Bazı örnek hesaplarla ilgili bilgileri bulunduran VSAM veri kümesini oluşturun. Bunu, CSQ4FILEveri kümesinde sağlanan JCL ' yi düzenleyip çalıştırarak yapın.
2. “z/OS üzerinde CICS ortamı için örnek uygulamaların hazırlanması” sayfa 1122içindeki adımları gerçekleştirin. (Örneğe ilişkin IMS uzantısını kullanmak istiyorsanız gerçekleştirmeniz gereken ek adımlar “z/OS üzerindeki Kredi Denetimi örneğinin IMS uzantısı” sayfa 1154içinde açıklanmıştır.)
3. CKTI tetikleyici izleyicisini başlat (IBM MQ for z/OS ile birlikte verilir) CSQ4SAMP.INITIATION.QUEUE, CICS hareket CKQC kullanılarak.
4. Uygulamayı başlatmak için CICS sisteminizi başlatın ve MVB1hareketini kullanın.
5. İlk panodan **Hemen** ya da **Toplu** sorguyu seçin.

   Anında ve toplu sorgu panoları benzerdir; Şekil 144 sayfa 1146 , Anında Sorgu panelini gösterir.

```
CSQ4VB2       IBM MQ for z/OS Sample Programs

Credit Check - Immediate Inquiry

Specify details of the request, then press Enter.
Name . . . . . _____________________
Social security number ___ __ ____
Bank account name . . ______________________________
Account number . . . . __________
Amount requested . . . 012345
Response from CHECKING ACCOUNT for name : ____________________
Account information not found
Credit worthiness index - NOT KNOWN
..
..
..
..
..
..
..
..
..
MESSAGE LINE
F1=Help F3=Exit F5=Make another inquiry
```

Şekil 144. Kredi Denetimi örnek uygulaması için Anında Sorgu panosu

6. Uygun alanlara bir hesap numarası ve kredi tutarı girin. Bu alanlara hangi bilgilerin girileceği konusunda yol gösterici bilgiler için bkz. “Sorgu panolarına bilgi girme” sayfa 1146 .

Sorgu panolarına bilgi girme

Kredi Denetimi örnek uygulaması, sorgu panolarının **İstenen Tutar** alanına girdiğiniz verilerin tamsayı biçiminde olup olmadığını denetler.

Aşağıdaki hesap numaralarından birini girerseniz, uygulama CSQ4BAQ: VSAM veri kümesinde uygun hesap adını, ortalama hesap bakiyesini ve kredi değeri dizinini bulur:

- 2222222222
- 3111234329
- 3256478962
- 3333333333
- 3501676212
- 3696879656
- 4444444444
- 5555555555
- 6666666666
- 7777777777

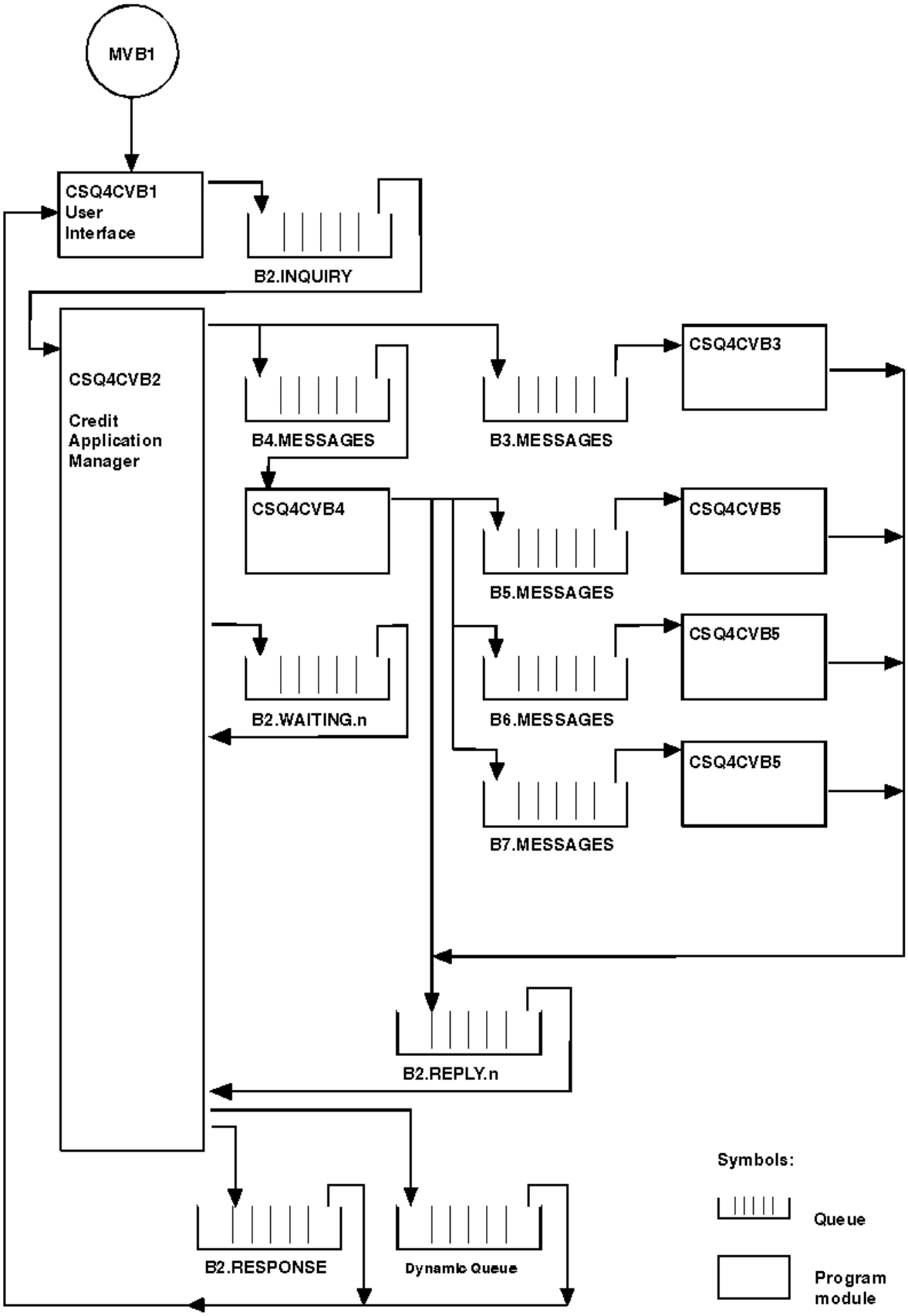
Diğer alanlara herhangi bir bilgi girebilir ya da girebilirsiniz. Uygulama, girdiğiniz bilgileri saklar ve oluşturduğu raporlarda aynı bilgileri döndürür.

z/OS *z/OS üzerinde Kredi Denetimi örneğinin tasarımı*

Bu bölümde, Kredi Denetimi örnek uygulamasını oluşturan her bir programın tasarımı açıklanmaktadır.

Uygulama tasarımı sırasında dikkate alınan tekniklerden bazıları hakkında daha fazla bilgi için bkz. “z/OS üzerinde Kredi denetimi örneği için tasarımla ilgili dikkat edilecek noktalar” sayfa 1152.

Şekil 145 sayfa 1147 , uygulamayı oluşturan programları ve bu programların hizmet vermekte olduğu kuyrukları gösterir. Bu şekilde, rakamın anlaşılmasını kolaylaştırmak için CSQ4SAMP öneki tüm kuyruk adlarından çıkarılmıştır.



Şekil 145. Credit Check örnek uygulamasına ilişkin programlar ve kuyruklar (yalnızca COBOL programları)

z/OS *z/OS üzerinde kullanıcı arabirimi programı (CSQ4CVB1)*

Etkileşimli kip CICS hareket MVB1hareketini başlattığınızda, uygulamaya ilişkin kullanıcı arabirimi programı başlatılır.

Bu program sorgu iletilerini CSQ4SAMP.B2.INQUIRY ve bu sorguları, sorguyu ne zaman yaptığını belirttiği bir yanıt kuyruğundan alır. Kullanıcı arabiriminden hemen ya da toplu sorgular gönderebilirsiniz:

- Anında sorgular için program, yanıt kuyruğu olarak kullandığı geçici bir dinamik kuyruk oluşturur. Bu, her sorgunun kendi yanıt kuyruğuna sahip olduğu anlamına gelir.
- Toplu sorgular için, kullanıcı arabirimi programı CSQ4SAMP.B2.RESPONSE. Basitçe, program, bu tek yanıt kuyruğundan tüm sorguları için yanıtlar alır. Bir bankanın MVB1' in her kullanıcısı için ayrı bir yanıt kuyruğu kullanmak isteyebileceğini görmek kolaydır, böylece her biri yalnızca başlattıkları sorguları görebilir.

Toplu kipte ve anında kipte kullanılan iletilerin özellikleri arasındaki önemli farklılıklar şunlardır:

- Toplu iş için iletiler düşük önceliğe sahiptir, bu nedenle anlık kipte girilen kredi isteklerinden sonra işlenir. Ayrıca, uygulama ya da kuyruk yöneticisinin yeniden başlatılması gerekirse, iletiler kalıcı olarak kurtarılır.
- Anında çalışma için iletiler yüksek önceliğe sahiptir, bu nedenle toplu kipte girilen kredi isteklerinden önce işlenir. Ayrıca, uygulama ya da kuyruk yöneticisinin yeniden başlatılması gerekirse, iletiler kalıcı olmadığı için atılır.

Ancak, her durumda kredi isteği iletilerinin özellikleri başvuru boyunca yayılır. Örneğin, yüksek öncelikli bir istekten elde edilen tüm iletilerin de yüksek önceliği vardır.

z/OS *z/OS üzerinde kredi uygulaması yöneticisi (CSQ4CVB2)*

Credit Application Manager (CAM) programı, Kredi Denetimi uygulamasına ilişkin işlemlerin çoğunu gerçekleştirir.

CAM, CSQ4SAMP.B2.INQUIRY ya da kuyruk CSQ4SAMP.B2.REPLYkuyruğunda bir tetikleme olayı ortaya çıktığında CKTI tetikleyici izleyicisi (IBM MQ for z/OSile birlikte verilir) tarafından başlatılır. *n*, burada *n* , bir yanıt kuyrukları kümesinden birini tanımlayan bir tamsayıdır. Tetikleme iletisi, tetikleme olayının oluştuğu kuyruğun adını içeren verileri içerir.

CAM, CSQ4SAMP.B2.WAITING.n . Kuyrukların her biri bir yanıt kuyruğuyla (örneğin, kuyruk CSQ4SAMP.B2.WAITING.3 , belirli bir sorguya ilişkin giriş verilerini ve CSQ4SAMP.B2.REPLY.3 , aynı sorguyla ilgili tüm yanıt iletilerini (veritabanlarını sorgulayan programlardan) içerir. Bu tasarımın ardındaki nedenleri anlamak için bkz. “CAM ' de ayrı sorgu ve yanıt kuyrukları” sayfa 1152.

Başlatma mantığı

Tetikleme olayı CSQ4SAMP.B2.INQUIRY, CAM, paylaşılan erişim için kuyruğu açar. Daha sonra, boş bir yanıt kuyruğu bulununcaya kadar her yanıt kuyruğunu açmayı dener. Boş bir yanıt kuyruğu bulamazsa, CAM bu gerçeği günlüğe kaydeder ve olağan şekilde sona erer.

Tetikleme olayı CSQ4SAMP.B2.REPLY.n, CAM kuyruğu dışlayıcı erişim için açar. Dönüş kodu nesnenin zaten kullanımda olduğunu bildirirse, CAM normal şekilde sonlandırılır. Başka bir hata oluşursa, CAM hatayı günlüğe kaydeder ve sonlandırır. CAM, ilgili bekleme kuyruğunu ve sorgu kuyruğunu açar, ardından iletileri almaya ve işlemeye başlar. CAM, bekleme kuyruğundan, kısmen tamamlanmış sorgulardaki ayrıntıları kurtarır.

Bu örnekteki basitlik için, kullanılan kuyrukların adları programda tutulur. Bir iş ortamında, kuyruk adları büyük olasılıkla program tarafından erişilen bir dosyada tutulur.

Sorgu kuyruğundan ileti alma

CAM önce, MQGET çağrısıyla MQGMO_SET_SIGNAL seçeneğini kullanarak sorgu kuyruğundan bir ileti alma girişiminde bulunur. Bir ileti hemen varsa, ileti işlenir; ileti yoksa, bir sinyal ayarlanır.

Daha sonra CAM, MQGET çağrısıyla aynı seçeneği kullanarak yanıt kuyruğundan bir ileti alma girişiminde bulunur. Bir ileti hemen kullanılabiliyorsa, ileti işlenir; tersi durumda bir sinyal ayarlanır.

Her iki sinyal de ayarlandığında, program sinyallerden biri gönderilinceye kadar bekler. Bir iletinin kullanılabilir olduğunu belirtmek için bir sinyal gönderilirse, ileti alınır ve işlenir. Sinyal sona ererse ya da kuyruk yöneticisi sona ererse, program sona erer.

CAM tarafından alınan ileti işleniyor

CAM tarafından alınan bir ileti dört tipte olabilir:

- Sorgu iletisi
- Yanıt iletisi
- Yayma iletisi
- Beklenmeyen ya da istenmeyen bir ileti

CAM, bu iletileri “z/OS üzerinde CAM tarafından alınan ileti işleniyor” sayfa 1149içinde açıklandığı gibi işler.

Yanıt gönderilmesi

CAM, bir sorgu için beklediği tüm yanıtları aldığında, yanıtları işler ve tek bir yanıt iletisi oluşturur. Aynı `CorrelId`iletisine sahip tüm yanıt iletilerinden alınan tüm verileri tek bir iletide birleştirir. Bu yanıt, özgün kredi isteğinde belirtilen yanıt kuyruğuna konmuştur. Yanıt iletisi, son yanıt iletisinin alınmasını içeren aynı iş birimi içine konmuştur. Bu, CSQ4SAMP.B2.WAITING.n.

Kısmen tamamlanmış sorguları kurtarma

CAM, CSQ4SAMP.B2.WAITING.n aldığı tüm iletiler. İleti tanımlayıcısının alanlarını şu şekilde ayarlar:

- *`Priority`* , ileti tipine göre belirlenir:
  - İstek iletileri için öncelik = 3
  - Veri paketleri için öncelik = 2
  - Yanıt iletileri için öncelik = 1
- *`CorrelId`* , kredi isteği iletisinin *`MsgId`* olarak ayarlanır
- Diğer MQMD alanları, alınan iletinin alanlarından kopyalanır

Bir sorgu tamamlandığında, belirli bir sorguya ilişkin iletiler yanıt işleme sırasında bekleme kuyruğundan kaldırılır. Bu nedenle, bekleme kuyruğu herhangi bir zamanda, devam eden sorgularla ilgili tüm iletileri içerir. Bu iletiler, programın yeniden başlatılması gerekirse, devam eden sorguların ayrıntılarını kurtarmak için kullanılır. Farklı öncelikler, sorgu iletilerinin yayımlar ya da yanıt iletilerinden önce kurtarılması için ayarlanır.

z/OS *z/OS üzerinde CAM tarafından alınan ileti işleniyor*

Credit Application Manager (CAM) tarafından alınan bir ileti dört tipte olabilir. CAM ' nin bir iletiyi nasıl işleyeceği, iletinin tipine bağlıdır.

CAM tarafından alınan bir ileti dört tipte olabilir:

- Sorgu iletisi
- Yanıt iletisi
- Yayma iletisi
- Beklenmeyen ya da istenmeyen bir ileti

CAM, bu iletileri aşağıdaki gibi işler:

Sorgu iletisi
Sorgu iletileri kullanıcı arabirimi programından gelir. Her kredi isteği için bir sorgu iletisi oluşturur.

Tüm kredi talepleri için CAM, müşterinin çek hesabının ortalama bakiyesini talep ediyor. Bunu, CSQ4SAMP.B2.OUTPUT.ALIAS. Bu kuyruk adı, CSQ4CVB3denetim hesabı programı tarafından işlenen CSQ4SAMP.B3.MESSAGESkuyruğuna çözülür. CAM bu diğer ad kuyruğuna bir ileti yerleştirdiğinde, uygun CSQ4SAMP.B2.REPLY.n kuyruğu. Burada, CSQ4CVB3 programının, farklı bir addaki temel kuyruğu işleyen başka bir programla kolayca değiştirilebilmesi için bir diğer ad kuyruğu kullanılır. Bunu yapmak için, diğer ad kuyruğunu yeni kuyruğa çözülebilecek şekilde yeniden tanımlayın. Ayrıca, diğer ad kuyruğuna ve temel kuyruğa farklı erişim yetkileri atayabilirsiniz.

Bir kullanıcı 10000 birimden büyük bir kredi isterse, CAM diğer veritabanlarında da denetimleri başlatır. Bunu, CSQ4SAMP.B4.MESSAGES, CSQ4CVB4. Bu kuyruğa hizmet veren işlem, kredi kartı geçmişi, tasarruf hesapları ve ipotek ödemeleri gibi diğer kayıtlara erişimi olan programların hizmet verdiği kuyruklara iletiyi yayacaktır. Bu programlardaki veriler, koyma işleminde belirlenen yanıt kuyruğuna döndürülür. Ayrıca, kaç yayma iletisinin gönderildiğini belirtmek için bu program tarafından yanıt kuyruğuna bir yayma iletisi gönderilir.

Bir iş ortamında, dağıtım programı, sağlanan verileri, diğer banka hesabı tiplerinin her birinin gerektirdiği biçimle eşleşecek şekilde yeniden biçimler.

Gönderme yapılan kuyruklardan herhangi biri uzak sistemde olabilir.

Her sorgu iletisi için CAM, bellekte yerleşik Sorgu Kaydı Tablosu 'nda (IRT) bir giriş başlatır. Bu kayıt şunları içerir:

- Sorgu iletisinin `MsgId`
- ReplyExp alanında, beklenen yanıt sayısı (gönderilen ileti sayısına eşit)
- ReplyRec alanında alınan yanıt sayısı (bu aşamada sıfır)
- PropsOut alanında, bir yayma iletisinin beklenip beklenmediğini gösteren bir gösterge

CAM, sorgu iletisini aşağıdaki işlemleri içeren bekleme kuyruğuna kopyalar:

- `Priority` 3 olarak ayarla
- `CorrelId` sorgu iletisinin `MsgId` değerine ayarlayın
- Sorgu iletisindeki diğer ileti tanımlayıcı alanları

Yayma iletisi

Yayılım iletisi, dağıtım programının sorguyu ilettiği kuyrukların sayısını içerir. İleti aşağıdaki gibi işlenir:

1. IRT ' deki uygun kaydın ReplyExp alanına gönderilen ileti sayısını ekleyin. Bu bilgiler iletide yer alır.
2. IRT ' deki kaydın ReplyRec alanını 1 artırın.
3. IRT ' deki kaydın PropsOut alanını 1 azaltır.
4. İletiyi bekleme kuyruğuna kopyalayın. CAM, `Priority` öğesini 2 olarak ve ileti tanımlayıcısının diğer alanlarını yayılım iletisindeki alanlara ayarlar.

Yanıt iletisi

Yanıt iletisi, çek hesabı programına ya da ajans sorgu programlarından birine yapılan isteklerden birine verilen yanıtı içerir. Yanıt iletileri aşağıdaki gibi işlenir:

1. IRT ' deki kaydın ReplyRec alanını 1 artırın.
2. İletiyi bekleme kuyruğuna `Priority` 1 olarak ayarlanarak ve ileti tanımlayıcısının diğer alanları yanıt iletisinin alanlarına ayarlanarak kopyalayın.
3. ReplyRec = ReplyExpve PropsOut = 0 ise, MsgComplete işaretini ayarlayın.

Diğer iletiler

Uygulama başka ileti beklemiyor. Ancak, uygulama sistem tarafından yayınlanan iletileri alabilir ya da bilinmeyen bir `CorrelId`siçeren iletileri yanıtlayabilir.

CAM, bu iletileri CSQ4SAMP.DEAD.QUEUE, burada incelenebilir. Bu koyma işlemi başarısız olursa, ileti kaybolur ve program devam eder. Programın bu bölümünün tasarımıyla ilgili daha fazla bilgi için bkz. “Örneğin beklenmeyen iletileri nasıl işleyeceğini” sayfa 1153.

z/OS *z/OS üzerinde çek hesabı programı (CSQ4CVB3)*

Hesap denetleme programı, CSQ4SAMP.B3.MESSAGES. Kuyruk açıldıktan sonra, bu program bekleme seçeneğiyle MQGET çağrısıyla ve bekleme aralığı 30 saniye olarak ayarlanmış olarak kuyruktan bir ileti alır.

Program, kredi isteği iletisindeki hesap numarası için CSQ4BAQ VSAM veri kümesini arar. İlgili hesap adını, ortalama bakiyeyi ve kredi değeri endeksini alır ya da hesap numarasının veri kümesinde olmadığını not eder.

Daha sonra program, kredi isteği iletisinde adı belirtilen yanıt kuyruğuna bir yanıt iletisi (MQPUT1 çağrısı kullanılarak) koyar. Bu yanıt iletisi için program:

- Kredi isteği iletisinin `CorrelId` kopyasını kopyalar
- MQPMO_PASS_IDENTITY_CONTEXT seçeneğini kullanır

Program, bekleme süresi doluncaya kadar kuyruktan ileti almaya devam eder.

z/OS *z/OS üzerinde dağıtım programı (CSQ4CVB4)*

Dağıtım programı, CSQ4SAMP.B4.MESSAGES.

Kredi talebinin kredi kartı geçmişi, tasarruf hesapları ve ipotek ödemeleri gibi kayıtlara erişimi olan diğer kurumlara dağıtımının benzetimini yapmak için program, namelist CSQ4SAMP.B4.NAMELIST. CSQ4SAMP.B *n*.MESSAGES, burada *n* 5, 6 ya da 7 'dir. Bir iş uygulamasında, ajanslar ayrı yerlerde olabilir, bu nedenle bu kuyruklar uzak kuyruklar olabilir. Örnek uygulamayı bunu gösterecek şekilde değiştirmek istiyorsanız, bkz. “z/OS üzerinde birden çok kuyruk yöneticisiyle Kredi Denetimi örneği” sayfa 1154.

Dağıtım programı aşağıdaki adımları gerçekleştirir:

1. Ad çizelisinden, programın kullanacağı kuyrukların adlarını alır. Program bunu, ad listesi nesnesinin özniteliklerini sorgulamak için MQINQ çağrısıyla yapar.
2. Bu kuyrukları ve CSQ4SAMP.B4.MESSAGES.
3. CSQ4SAMP.B4.MESSAGES:
   a. Bekleme seçeneğiyle ve bekleme aralığı 30 saniye olarak ayarlanmış MQGET çağrısıyla bir ileti alın.
   b. Ad listesinde yer alan her kuyruğa uygun CSQ4SAMP.B2.REPLY.n kuyruğu. Program, kredi isteği iletisinin *`CorrelId`* kopyasını bu kopyalama iletilerine kopyalar ve MQPUT çağrısında MQPMO_PASS_IDENTITY_CONTEXT seçeneğini kullanır.
   c. CSQ4SAMP.B2.REPLY.n , kaç iletiyi başarıyla yerleştirdiğini gösterir.
   d. Bir eşitleme noktası bildirin.

z/OS *Aracı sorgu programı (CSQ4CVB5/CSQ4CCB5)- z/OS*

Ajans-sorgu programı, hem COBOL programı hem de C programı olarak sağlanır. Her iki program da aynı tasarıma sahiptir. Bu, farklı türdeki programların bir IBM MQ uygulamasında kolayca bir arada bulunabileceğini ve bu tür bir uygulamayı oluşturan program modüllerinin kolayca değiştirilebileceğini gösterir.

Programın bir eşgörünümü, aşağıdaki kuyruklardan herhangi birinde bir tetikleme olayı tarafından başlatılır:

- COBOL programı için (CSQ4CVB5):
  - CSQ4SAMP.B5.MESSAGES
  - CSQ4SAMP.B6.MESSAGES
  - CSQ4SAMP.B7.MESSAGES
- C programı için (CSQ4CCB5), kuyruk CSQ4SAMP.B8.MESSAGES

Not: C programını kullanmak istiyorsanız, CSQ4SAMP.B7.MESSAGES kuyruğunu CSQ4SAMP.B8.MESSAGESile değiştirmek için ad listesi CSQ4SAMP.B4.NAMELIST tanımını değiştirmeniz gerekir. Bunu yapmak için aşağıdakilerden herhangi birini kullanabilirsiniz:

- IBM MQ for z/OS işlemleri ve denetim panoları
- ALTER NAMELIST komutu
- CSQUTIL yardımcı programı

Uygun kuyruk açıldıktan sonra bu program, bekleme seçeneğiyle MQGET çağrısıyla ve bekleme aralığı 30 saniye olarak ayarlanmış olarak kuyruktan bir ileti alır.

Bu program, CSQ4BAQ VSAM veri kümesinde kredi isteği iletisinde iletilen hesap numarasını arayarak bir kurumun veri tabanının aranmasını benzetir. Daha sonra, hizmet vermekte olduğu kuyruğun adını ve bir güvenilirlik dizinini içeren bir yanıt oluşturur. İşlemeyi basitleştirmek için, güvenilirlik endeksi rasgele seçilir.

Yanıt iletisini koyarken, program MQPUT1 çağrıyı kullanır ve:

- Kredi isteği iletisinin `CorrelId` kopyasını kopyalar
- MQPMO_PASS_IDENTITY_CONTEXT seçeneğini kullanır

Program, kredi isteği iletisinde adı belirtilen yanıt kuyruğuna yanıt iletisi gönderir. (Kredi isteği iletisinde, yanıt kuyruğunun iyesi olan kuyruk yöneticisinin adı da belirtilir.)

z/OS *z/OS üzerinde Kredi denetimi örneği için tasarımla ilgili dikkat edilecek noktalar*

Kredi Denetimi örneği için tasarım konuları.

Bu konu aşağıdakilerle ilgili bilgi içerir:

- “CAM ' de ayrı sorgu ve yanıt kuyrukları” sayfa 1152
- “Örneğin hataları nasıl işleyeceğini” sayfa 1152
- “Örneğin beklenmeyen iletileri nasıl işleyeceğini” sayfa 1153
- “Örneğin eşitleme noktalarını kullanma şekli” sayfa 1153
- “Örneğin ileti bağlamı bilgilerini kullanma şekli” sayfa 1153
- “CAM ' de ileti ve ilinti tanıtıcılarının kullanımı” sayfa 1154

CAM ' de ayrı sorgu ve yanıt kuyrukları

Uygulama hem sorgular hem de yanıtlar için tek bir kuyruk kullanabilir, ancak aşağıdaki nedenlerden ötürü ayrı kuyruklar kullanacak şekilde tasarlanmıştır:

- Program sorgu sayısı üst sınırını işlerken, başka sorgular kuyrukta bırakılabilir. Tek bir kuyruk kullanılıyorsa, bunun kuyruktan alınması ve başka bir yerde saklanması gerekir.
- İleti trafiği izin verecek kadar yüksekse, CAM ' nin diğer eşgörünümleri aynı sorgu kuyruğuna hizmet vermek için otomatik olarak başlatılabilir. Ama program devam eden sorguları takip etmeli ve bunu yapmak için başlattığı sorulardaki tüm cevapları geri almalıdır. Yalnızca bir kuyruk kullanılırsa, program, iletilerin bu program için mi, yoksa başka bir kuyruk için mi olduğunu görmek için iletilere göz atmalıdır. Bu, operasyonu çok daha az verimli hale getirir.

  Uygulama, birden çok CAM ' yi destekleyebilir ve eşlenmiş yanıt ve bekleme kuyruklarını kullanarak devam eden sorguları etkili bir şekilde kurtarabilir.
- Program, sinyal kullanarak birden çok kuyruğunda etkin bir şekilde bekleyebilir.

Örneğin hataları nasıl işleyeceğini

Kullanıcı arabirimi programı hataları doğrudan kullanıcıya raporlayarak işler.

Diğer programların kullanıcı arabirimleri yoktur, bu nedenle hataları başka yollarla ele almaları gerekir. Ayrıca, birçok durumda (örneğin, bir MQGET çağrısı başarısız olursa) bu diğer programlar, uygulamanın kullanıcısının kimliğini bilmez.

Diğer programlar hata iletilerini CSQ4SAMPadlı CICS geçici depolama kuyruğuna yerleştirir. CICStarafından sağlanan hareket CEBR 'sini kullanarak bu kuyruğa göz atabilirsiniz. Programlar hata iletilerini CICS CSML günlüğüne de yazar.

Örneğin beklenmeyen iletileri nasıl işleyeceğini

Bir ileti kuyruğa alma uygulaması tasarladığınızda, kuyruğa beklenmedik bir şekilde gelen iletilerin nasıl işleneceğine karar vermeniz gerekir.

İki temel seçenek şunlardır:

- Uygulama, beklenmeyen iletiyi işlemeden artık çalışmaz. Bu, büyük olasılıkla uygulamanın bir işletmene bildirimde bulunması, kendisini sonlandırması ve otomatik olarak yeniden başlatılmamasını sağlaması anlamına gelir (tetikleme ayarını kapatarak bunu yapabilir). Bu seçenek, uygulamaya ilişkin tüm işlemlerin tek bir beklenmeyen iletiyle durdurulabileceği ve uygulamayı yeniden başlatmak için bir işlecin araya girmesi gerektiği anlamına gelir.
- Uygulama, iletiyi hizmet vermekte olduğu kuyruktan kaldırır, iletiyi başka bir konuma yerleştirir ve işlemeye devam eder. Bu iletiyi koymak için en iyi yer, sistem gitmeyen ileti kuyruğudur.

İkinci seçeneği belirlerseniz:

- Bir işletmen ya da başka bir program, iletilerin nereden geldiğini bulmak için gitmeyen iletiler kuyruğuna konan iletileri incelemelidir.
- Gönderilmeyen iletiler kuyruğuna yerleştirilemezse, beklenmeyen bir ileti kaybolur.
- Uzun bir beklenmeyen ileti, gönderilmeyen iletiler kuyruğundaki iletilerin sınırından daha uzun ya da programdaki arabellek büyüklüğünden daha uzun olursa kesilir.

Uygulamanın dış etkinliklerden minimum etkiyle tüm sorguları sorunsuz bir şekilde işlediğinden emin olmak için Kredi Denetimi örnek uygulaması ikinci seçeneği kullanır. Örneği aynı kuyruk yöneticisini kullanan diğer uygulamalardan ayrı tutmanızı sağlamak için Kredi Denetimi örneği, sistemin gitmeyen iletiler kuyruğunu kullanmaz; bunun yerine kendi gitmeyen iletiler kuyruğunu kullanır. Bu kuyruk CSQ4SAMP.DEAD.QUEUE. Örnek, örnek programlar için sağlanan arabellek alanından daha uzun olan iletileri keser. Bu kuyruktaki iletilere göz atmak için Göz At örnek uygulamasını ya da iletileri ileti tanımlayıcılarıyla birlikte yazdırmak için İleti Yazdır örnek uygulamasını kullanabilirsiniz.

Ancak, örneği birden çok kuyruk yöneticisi, beklenmeyen iletiler ya da teslim edilemeyen iletiler üzerinde çalışacak şekilde genişletir ve kuyruk yöneticisi tarafından sistemin kullanılmayan ileti kuyruğuna konabilir.

Örneğin eşitleme noktalarını kullanma şekli

Credit Check örnek uygulamasındaki programlar, aşağıdakileri sağlamak için eşitleme noktalarını bildirir:

- Beklenen her iletiye yanıt olarak yalnızca bir yanıt iletisi gönderilir
- Beklenmeyen iletilerin birden çok kopyası hiçbir zaman örneğin gitmeyen iletiler kuyruğuna konmaz
- CAM, bekleme kuyruğundan kalıcı iletiler alarak kısmen tamamlanmış tüm sorguları kurtarabilir

Bunu başarmak için, bir iletinin alınmasını, bu iletinin işlenmesini ve sonraki herhangi bir koyma işlemini kapsayacak tek bir iş birimi kullanılır.

Örneğin ileti bağlamı bilgilerini kullanma şekli

Kullanıcı arabirimi programı (CSQ4CVB1) ileti gönderdiğinde, MQPMO_DEFAULT_CONTEXT seçeneğini kullanır. Bu, kuyruk yöneticisinin hem kimlik hem de kaynak bağlam bilgileri ürettiği anlamına gelir. Kuyruk yöneticisi bu bilgileri programı başlatan hareketten (MVB1) ve hareketi başlatan kullanıcı kimliğinden alır.

CAM sorgu iletileri gönderdiğinde, MQPMO_PASS_IDENTITY_CONTEXT seçeneğini kullanır. Bu, konmakta olan iletinin kimlik bağlamı bilgilerinin özgün sorgu iletisinin kimlik bağlamından kopyalandığı anlamına gelir. Bu seçenekle, kaynak bağlam bilgileri kuyruk yöneticisi tarafından oluşturulur.

CAM yanıt iletileri gönderdiğinde, MQPMO_ALTERNATE_USER_AUTHORITY seçeneğini kullanır. Bu, CAM bir yanıt kuyruğu açtığında kuyruk yöneticisinin güvenlik denetimi için diğer bir kullanıcı kimliği kullanmasına neden olur. CAM, özgün sorgu iletisinin göndericisinin kullanıcı kimliğini kullanır. Bu, kullanıcıların yalnızca oluşturdukları sorguları görmelerine izin verileceği anlamına gelir. Diğer kullanıcı kimliği, özgün sorgu iletisinin ileti tanımlayıcısındaki kimlik bağlamı bilgilerinden alınır.

Sorgu programları (CSQ4CVB3/4/5) yanıt iletileri gönderdiğinde, MQPMO_PASS_IDENTITY_CONTEXT seçeneğini kullanırlar. Bu, konmakta olan iletinin kimlik bağlamı bilgilerinin özgün sorgu iletisinin kimlik bağlamından kopyalandığı anlamına gelir. Bu seçenekle, kaynak bağlam bilgileri kuyruk yöneticisi tarafından oluşturulur.

Not: MVB3/4/5 işlemleriyle ilişkilendirilen kullanıcı kimliği, B2.REPLY.n kuyrukları. Bu kullanıcı kimlikleri, işlenmekte olan istekle ilişkili kimliklerle aynı olmayabilir. Bu olası güvenlik açığını atlamak için, sorgu programları yanıtlarını koyarken MQPMO_ALTERNATE_USER_AUTHORITY seçeneğini kullanabilir. Bu, MVB1 ' in her bir kullanıcısının B2.REPLY.n kuyrukları.

CAM ' de ileti ve ilinti tanıtıcılarının kullanımı

Uygulama, herhangi bir zamanda işlediği tüm canlı sorguları izlemek zorundadır. Bunu yapmak için, her bir sorguyla ilgili sahip olduğu tüm bilgileri ilişkilendirmek üzere her kredi isteği iletisinin benzersiz ileti tanıtıcısını kullanır.

CAM, sorgu iletisinin `MsgId` öğesini, sorgu için gönderdiği tüm istek iletilerinin `CorrelId` içine kopyalar. Örnekteki diğer programlar (CSQ4CVB3 -5), aldıkları her iletinin `CorrelId` dosyasını yanıt iletilerinin `CorrelId` içine kopyalar.

z/OS *z/OS üzerinde birden çok kuyruk yöneticisiyle Kredi Denetimi örneği*

Örneği iki kuyruk yöneticisine ve CICS sistemlerine (her kuyruk yöneticisi farklı bir CICS sistemine bağlıyken) kurarak dağıtılmış kuyruğa alma işlemini göstermek için Kredi Denetimi örnek uygulamasını kullanabilirsiniz.

Örnek program kurulduğunda ve her sistemde tetikleyici izleme programı (CKTI) çalışıyorsa, aşağıdakileri yapmanız gerekir:

1. İki kuyruk yöneticisi arasındaki iletişim bağlantısını ayarlayın. Bunun nasıl yapılacağını öğrenmek için Dağıtılmış kuyruğa alma özelliğini yapılandırmabaşlıklı konuya bakın.
2. Bir kuyruk yöneticisinde, kullanmak istediğiniz uzak kuyrukların (diğer kuyruk yöneticisinde) her biri için yerel bir tanımlama yaratın. Bu kuyruklar CSQ4SAMP.B *n*.MESSAGES, burada *n* 3, 5, 6 ya da 7 'dir. (Bunlar, çek hesabı programı ve ajans sorgu programı tarafından sunulan kuyruklardır.) Bunun nasıl yapılacağını öğrenmek için DEFINE QREMOTE ve DEFINE kuyruklarıbaşlıklı konuya bakın.
3. Ad listesi tanımını değiştirin (CSQ4SAMP.B4.NAMELIST), kullanmak istediğiniz uzak kuyrukların adlarını içermesini sağlar. Bunun nasıl yapılacağını öğrenmek için DEFINE NAMELISTbaşlıklı konuya bakın.

z/OS *z/OS üzerindeki Kredi Denetimi örneğinin IMS uzantısı*

Çek hesabı programının bir sürümü, bir IMS toplu ileti işleme (BMP) programı olarak sağlanır. C dilinde yazılmıştır.

Program, CICS sürümüyle aynı işlevi gerçekleştirir; ancak, hesap bilgilerini almak için, program bir VSAM dosyası yerine bir IMS veritabanını okur. Çek hesabı programının CICS sürümünü IMS sürümüyle değiştirirseniz, uygulamayı kullanma yönteminde bir fark görmüyorsunuz.

IMS sürümünü hazırlamak ve çalıştırmak için aşağıdakileri yapmanız gerekir:

1. “z/OS üzerinde Kredi Denetimi örneğinin hazırlanması ve çalıştırılması” sayfa 1145içindeki adımları izleyin.
2. “z/OS üzerindeki IMS ortamı için örnek uygulamanın hazırlanması” sayfa 1126içindeki adımları izleyin.
3. CSQ4SAMP.B2.OUTPUT.ALIAS .IMS.MESSAGES (CSQ4SAMP.B3.MESSAGES). Bunu yapmak için aşağıdakilerden birini kullanabilirsiniz:
   - IBM MQ for z/OS işlemleri ve denetim panoları
   - ALTER QALIAS komutu.

IMS denetim hesabı programını kullanmanın başka bir yolu da, dağıtım programından ileti alan kuyruklardan birine hizmet vermektir. Kredi Denetimi örnek uygulamasının teslim edilen formunda, bu kuyruklardan üç tanesi vardır (B5/6/7.MESSAGES), tümü ajans-sorgu programı tarafından sunulur. Bu

program bir VSAM veri kümesini arar. VSAM veri kümesinin ve IMS veritabanının kullanımını karşılaştırmak için IMS denetim hesabı programının bu kuyruklardan birine hizmet etmesini sağlayabilirsiniz. Bunu yapmak için, namelist CSQ4SAMP.B4.NAMELIST tanımlamasını CSQ4SAMP.B *n*ürünlerinden birinin yerine geçecek şekilde değiştirmeniz gerekir.CSQ4SAMP.B3 içeren MESSAGES.IMS.MESSAGES kuyruğu. Aşağıdakilerden birini kullanabilirsiniz:

- IBM MQ for z/OS işlemleri ve denetim panoları
- ALTER NAMELIST komutu.

Daha sonra örneği CICS hareket MVB1içinden çalıştırabilirsiniz. Kullanıcı, işlem ya da yanıtta herhangi bir fark görmez. IMS BMP, bir durdurma iletisi aldıktan sonra ya da beş dakika etkinlik dışı kaldıktan sonra durur.

IMS çek hesabı programının tasarımı (CSQ4ICB3)

Bu program bir BMP olarak çalışır. Herhangi bir IBM MQ iletisi gönderilmeden önce JCL ' yi kullanarak programı başlatın.

Program, kredi isteği iletilerinde hesap numarası için bir IMS veritabanında arama yapar. İlgili hesap adını, ortalama bakiyeyi ve kredi değeri endeksini alır.

Program, veritabanı aramasının sonuçlarını işlenmekte olan IBM MQ iletisinde adı belirtilen yanıt kuyruğuna gönderir. Döndürülen ileti, yanıt oluşturma işleminin doğru sorgunun işlendiğini doğrulayabilmesi için hesap tipini ve arama sonuçlarını alınan iletiye ekler. İleti, aşağıdaki gibi üç 79 karakterlik grup biçimindedir:

```
'Response from CHECKING ACCOUNT for name : JONES J B'
'   Opened 870530, 3-month average balance = 000012.57'
'   Credit worthiness index - BBB'
```

İleti odaklı bir BMP olarak çalışırken, program IMS ileti kuyruğunu boşaltır, ardından iletileri IBM MQ for z/OS kuyruğundan okur ve işler. IMS ileti kuyruğundan bilgi alınmaz. Program, tanıtıcıları kapatıldığından, her denetim noktasından sonra kuyruk yöneticisine yeniden bağlanır.

Toplu iş odaklı bir BMP ' de çalışırken, tanıtıcıları kapatılmadığı için program her denetim noktasından sonra kuyruk yöneticisine bağlanmaya devam eder.

z/OS *z/OS üzerinde İleti İşleyici örneği*

İleti İşleyici örnek TSO uygulaması, bir kuyruktaki iletilere göz atmanızı, iletileri iletmenizi ve silmenizi sağlar. Örnek, C ve COBOL dillerinde kullanılabilir.

Örnek hazırlanıyor ve çalıştırılıyor

Aşağıdaki adımları izleyin:

1. Örneği, “z/OS üzerinde TSO ortamı için örnek uygulamaların hazırlanması” sayfa 1121içinde açıklandığı gibi hazırlayın.
2. Örnek için sağlanan CLIST ' i (CSQ4RCH1), panoların yerini, ileti kütüğünün yerini ve yükleme modüllerinin yerini tanımlayacak şekilde uyarlamanızı sağlar.

Örneğin hem C, hem de COBOL sürümünü çalıştırmak için CLIST CSQ4RCH1 komutunu kullanabilirsiniz. Sağlanan CSQ4RCH1 sürümü C sürümünü çalıştırır ve COBOL sürümü için gerekli uyarlama yönergelerini içerir.

Not:

1. Örnekle birlikte sağlanan örnek kuyruk tanımlaması yok.
2. VS COBOL II, ISPFile çoklu görevi desteklemez, bu nedenle bölünmüş ekranın her iki tarafında Message Handler örnek uygulamasını kullanmayın. Eğer yaparsanız, sonuçlar önceden kestirilemez.

z/OS *z/OS üzerinde Message Handler örneğini kullanma*

Örneği kurup uyarlanmış CLIST CSQ4RCH1' den çağırdıktan sonra, Şekil 146 sayfa 1156 içinde gösterilen ekran görüntülenir.

```
---------------------- IBM MQ for z/OS -- Samples ----------------------
COMMAND ===>
User Id : JOHNJ



Enter information. Press ENTER :


Queue Manager Name  : ________________________________________________ :

Queue Name      : _____________________________________________ :









F1=HELP   F2=SPLIT   F3=END    F4=RETURN  F5=RFIND   F6=RCHANGE
F7=UP    F8=DOWN   F9=SWAP   F10=LEFT   F11=RIGHT  F12=RETRIEVE
```

Şekil 146. Message Handler örneği için ilk ekran

Görüntülenecek kuyruk yöneticisini ve kuyruk adını girin (büyük ve küçük harfe duyarlı) ve ileti listesi ekranı görüntülenir (bkz. Şekil 147 sayfa 1156).

```
---------------------- IBM MQ for z/OS -- Samples ------- Row 1 to 4 of 4
COMMAND ==>

Queue Manager  : VM03                        :
Queue      : MQEI.IMS.BRIDGE.QUEUE              :

Message number  01 of 04

Msg  Put Date  Put Time Format   User   Put Application
No  MM/DD/YYYY HH:MM:SS Name  Identifier  Type   Name
01  10/16/1998 13:51:19 MQIMS  NTSFV02   00000002 NTSFV02A
02  10/16/1998 13:55:45 MQIMS  JOHNJ    00000011 EDIT\CLASSES\BIN\PROGTS
03  10/16/1998 13:54:01 MQIMS  NTSFV02   00000002 NTSFV02B
04  10/16/1998 13:57:22 MQIMS  johnj    00000011 EDIT\CLASSES\BIN\PROGTS
****************************** Bottom of data ********************************
```

Şekil 147. Message Handler örneği için ileti listesi ekranı

Bu ekran, kuyruktaki ilk 99 iletiyi ve her biri için aşağıdaki alanları gösterir:

İlt No.
İleti numarası

Koyma Tarihi: AA/GG/YYYY
İletinin kuyruğa konma tarihi (GMT)

Koyma Süresi SS: DD: SS
İletinin kuyruğa konma zamanı (GMT)

Biçim Adı
MQMD.Format alanını biçimlendir

Kullanıcı Kimliği
MQMD.UserIdentifier alanı

Koyma Uygulaması Tipi
MQMD.PutApplType alanı

Koyma Uygulaması Adı
MQMD.PutApplName alanı

Kuyruktaki iletilerin toplam sayısı da görüntülenir.

Bu ekrandan, imleç konumuna göre değil, sayı olarak bir ileti seçilebilir ve daha sonra görüntülenebilir. Örneğin, bkz. Şekil 148 sayfa 1157.

```
---------------------- IBM MQ for z/OS -- Samples ----- Row 1 to 35 of 35
COMMAND ==>

Queue Manager  : VM03                        :
Queue      : MQEI.IMS.BRIDGE.QUEUE              :
Forward to Q Mgr : VM03                       :
Forward to Queue : QL.TEST.ISCRES1                 :

Action : _ :  (D)elete (F)orward

Message Content :
--------------------------------------------------------------------------------
Message Descriptor
StrucId     : `MD `
Version     : 000000001
Report      : 000000000
MsgType     : 000000001
Expiry      : -00000001
Feedback     : 000000000
Encoding     : 000000785
CodedCharSetId  : 000000500
Format      : `MQIMS  `
Priority     : 000000000
Persistence   : 000000001
MsgId      : `C3E2D840E5D4F0F34040404040404040AF6B30F0A89B7605`X
CorrelId     : `00000000000000000000000000000000000000000000000000000000`X
BackoutCount   : 000000000
ReplyToQ     : `QL.TEST.ISCRES1                `
ReplyToQMgr   : `VM03                        `
UserIdentifier  : `NTSFV02   `
AccountingToken :
`06F2F5F5F3F0F1000000000000000000000000000000000000000000000000000`X
ApplIdentityData : `                        `
PutApplType   : 000000002
PutApplName   : `NTSFV02A          `
PutDate     : `19971016`
PutTime     : `13511903`
ApplOriginData  : `   `

Message Buffer :  108 byte(s)
00000000 : C9C9 C840 0000 0001 0000 0054 0000 0311 `IIH .....T......`
00000010 : 0000 0000 4040 4040 4040 4040 0000 0000 `....    ....`
00000020 : 4040 4040 4040 4040 4040 4040 4040 4040 `                `
00000030 : 4040 4040 4040 4040 4040 4040 4040 4040 `                `
00000040 : 0000 0000 0000 0000 0000 0000 0000 0000 `................`
00000050 : 40F1 C300 0018 0000 C9C1 D7D4 C4C9 F2F8 ` 1C.....IAPMDI28`
00000060 : 40C8 C5D3 D3D6 40E6 D6D9 D3C4      ` HELLO WORLD  `
****************************** Bottom of data ********************************
```

Şekil 148. Seçilen ileti görüntülenir

İleti görüntülendikten sonra silinebilir, kuyrukta bırakılabilir ya da başka bir kuyruğa iletilebilir. `Forward to Q Mgr` ve `Forward to Queue` alanları MQMD ' deki değerlerle kullanıma hazırlanır, ileti iletilmeden önce bu değerler değiştirilebilir.

Örnek tasarım, ileti anahtar olarak `MsgId` ve `CorrelId` kullanılarak alındığından, yalnızca benzersiz `MsgId / CorrelId` birleşimlerine sahip iletilerin seçilmesine ve görüntülenmesine izin verir. Anahtar benzersiz değilse, örnek seçilen iletiyi kesin olarak alamaz.

Not: İletilere göz atmak için SCSQCLST (CSQ4RCH1) örneğini kullandığınızda, her çağırma iletinin geriletme sayısının artmasına neden olur. Bu örneğin davranışını değiştirmek istiyorsanız, örneği kopyalayın ve içeriği gerektiği şekilde değiştirin. Bu geriletme sayısına dayanan diğer uygulamaların bu artan sayıdan etkilenebildiğini bilmelisiniz.

z/OS *z/OS üzerinde örnek Message Handler örneğinin tasarımı*

Bu konuda, Message Handler örnek uygulamasını oluşturan her bir programın tasarımı açıklanmaktadır.

Nesne doğrulama programı

Bu, geçerli bir kuyruk ve kuyruk yöneticisi adı ister.

Bir kuyruk yöneticisi adı belirtmezseniz, varsa, varsayılan kuyruk yöneticisi kullanılır. Yalnızca yerel kuyruklar kullanılabilir; kuyruk tipinin yerel olup olmadığını denetlemek için bir MQINQ yayınlanır ve kuyruk yerel değilse bir hata bildirilir. Kuyruk başarıyla açılmazsa ya da kuyrukta MQGET çağrısı engellenirse, CompCode ve Reason dönüş kodunu gösteren hata iletileri döndürülür.

İleti listesi programı

Kuyruktaki iletilere ilişkin, putdate, puttime ve message format gibi bilgileri içeren bir liste görüntülenir.

Listede saklanan ileti sayısı üst sınırı 99 'dur. Kuyrukta bundan daha fazla ileti varsa, yürürlükteki kuyruk derinliği de görüntülenir. Görüntülemek üzere bir ileti seçmek için, giriş alanına ileti numarasını yazın (varsayılan değer 01 'dir). Girdiniz geçerli değilse, uygun bir hata iletisi alırsınız.

İleti içeriği programı

Bu, ileti içeriğini görüntüler.

İçerik biçimlendirilir ve iki parçaya bölünür:

1. İleti tanımlayıcı
2. İleti arabelleği

İleti tanımlayıcısı, her bir alanın içeriğini ayrı bir satırda gösterir.

İleti arabelleği, içeriğine bağlı olarak biçimlendirilir. Arabellek bir boş harf üstbilgisi (MQDLH) ya da iletim kuyruğu üstbilgisi (MQXQH) tutuyorsa, bunlar biçimlenir ve arabelleğin kendisinden önce görüntülenir.

Arabellek verileri biçimlenmeden önce, başlık satırı iletinin arabellek uzunluğunu bayt cinsinden gösterir. Arabellek büyüklüğü üst sınırı 32768 bayttır ve bundan daha uzun olan iletiler kesilir. Arabelleğin tam boyutu, iletinin yalnızca ilk 32768 baytın görüntülendiğini belirten bir iletiyle birlikte görüntülenir.

Arabellek verileri iki şekilde biçimlendirilir:

1. Arabelleğe göreli konum yazdırıldıktan sonra, arabellek verileri onaltılı olarak görüntülenir.
2. Arabellek verileri, EBCDIC değerleri olarak yeniden görüntülenir. Herhangi bir EBCDIC değeri yazdırılamazsa, bunun yerine bir nokta (.) yazdırır.

Silmek için D ya da işlem alanına iletmek için F girebilirsiniz. İletiyi iletmeyi seçerseniz, `forward-to queue` ve `queue manager name` doğru ayarlanmalıdır. Bu alanlara ilişkin varsayılan değerler, ileti tanımlayıcısı ReplyToQ ve ReplyToQMgr alanlarından okunur.

Bir iletiyi iletirseniz, arabellekte saklanan üstbilgi öbeği soyulur. İleti başarıyla iletildiyse, özgün kuyruktan kaldırılır. Geçersiz işlemler girerseniz, hata iletileri görüntülenir.

CSQ4CHP9 adlı örnek bir yardım panosu da vardır.

z/OS *z/OS üzerinde Zamanuyumsuz Put örneği*

Zamanuyumsuz Put örnek programı, zamanuyumsuz MQPUT çağrılarını kullanarak iletileri kuyruğa koyar. Örnek, MQSTAT çağrısından durum bilgilerini de alır.

Zamanuyumsuz Put uygulamaları şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQPUT

- MQSTAT
- MQCLOSE
- MQDISC

Örnek programlar C programlama dilinde sağlanır.

Zamanuyumsuz Put uygulamaları toplu iş ortamında çalışır. Toplu iş uygulamaları için Diğer örnekler konusuna bakın.

Bu konuda, Zamanuyumsuz Tüketim programının tasarımına ve CSQ4BCS2 örneğinin çalıştırılmasına ilişkin bilgiler de sağlanır.

- "CSQ4BCS2 örneğinin çalıştırılması" sayfa 1159
- "Zamanuyumsuz Put örnek programının tasarımı" sayfa 1159

CSQ4BCS2 örneğinin çalıştırılması

Bu örnek program en çok altı parametre alır:

1. Hedef kuyruğun adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. Açma seçenekleri (isteğe bağlı).
4. Seçenekleri kapat (isteğe bağlı).
5. Hedef kuyruk yöneticisinin adı (isteğe bağlı).
6. Dinamik kuyruğun adı (isteğe bağlı).

Bir kuyruk yöneticisi belirtilmezse, CSQ4BCS2 varsayılan kuyruk yöneticisine bağlanır. İleti içeriği standart giriş yoluyla sağlanır (**SYSIN DD**).

Programı çalıştırmak için örnek bir JCL var; CSQ4BCSP' de bulunuyor.

Zamanuyumsuz Put örnek programının tasarımı

Program, iletilerin konmasına ilişkin hedef kuyruğu açmak için, verilen çıkış seçenekleriyle ya da MQOO_OUTPUT ve MQOO_FAIL_IF_QUIESCING seçenekleriyle MQOPEN çağrısını kullanır.

Program kuyruğu açamazsa, MQOPEN çağrısıyla döndürülen neden kodunu içeren bir hata iletisi yayınlanır. Programı bu ve sonraki MQI çağrılarında basit tutmak için, birçok seçenek için varsayılan değerler kullanılır.

Her giriş satırı için, program metni bir arabelleğe okur ve o satırın metnini içeren bir veri paketi iletisi yaratmak için MQPMO_ASYNC_RESPONSE ile MQPUT çağrıyı kullanır ve iletiyi hedef kuyruğa zamanuyumsuz olarak yerleştirir. Program, girişin sonuna ulaşıncaya ya da MQPUT çağrısı başarısız oluncaya kadar devam eder. Program girişin sonuna ulaşırsa, MQCLOSE çağrıyı kullanarak kuyruğu kapatır.

Daha sonra program, MQSTS yapısı döndüren MQSTAT çağrısını verir ve başarıyla konan iletilerin sayısını, bir uyarıyla konan iletilerin sayısını ve başarısızlıkların sayısını içeren iletileri görüntüler.

Not: MQSTAT çağrısıyla bir MQPUT hatası saptandığında ne olduğunu görmek için, hedef kuyruktaki MAXDEPTH değerini düşük bir değere ayarlayın.

z/OS *z/OS üzerinde Toplu Zamanuyumsuz Tüketim örneği*

CSQ4BCS1 örnek programı C dilinde teslim edilir; birden çok kuyruktan gelen iletileri zamanuyumsuz olarak kullanmak için MQCB ve MQCTL ' nin kullanılmasını gösterir.

Zamanuyumsuz Tüketim örnekleri toplu iş ortamında çalışır. Toplu iş uygulamaları için Diğer örnekler konusuna bakın.

CICS ortamında çalışan bir COBOL örneği de vardır, bkz. "CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örneği z/OS" sayfa 1161.

Uygulamalar şu MQI çağrılarını kullanır:

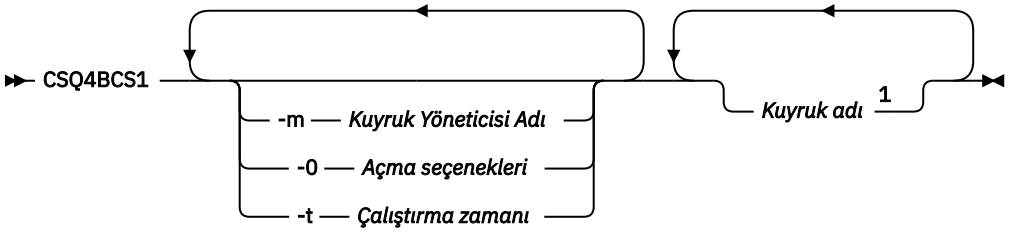
- MQCONN
- MQOPEN
- MQCLOSE
- MQDISC
- MQCB
- MQCTL

Bu konuda ayrıca aşağıdaki başlıklara ilişkin bilgiler de sağlanmıştır:

- "CSQ4BCS1 örneğinin çalıştırılması" sayfa 1160
- "Toplu Zamanuyumsuz Tüketim örnek programının tasarımı" sayfa 1160

CSQ4BCS1 örneğinin çalıştırılması

Bu örnek program aşağıdaki sözdizimini izler:



Notlar:

[1] Bu örnek en çok 10 kuyruğu destekler. Birden çok kuyruktan ileti okumak için birden çok kuyruk adı sağlayın.

Bu programı çalıştıracak örnek bir JCL var; bu JCL, CSQ4BCSC' de bulunur.

Toplu Zamanuyumsuz Tüketim örnek programının tasarımı

Örnek, birden çok kuyruktan gelen iletilerin geliş sırasına göre nasıl okunacağını gösterir. Bu, zamanuyumlu MQGET kullanarak daha fazla kod gerektirir. Zamanuyumsuz tüketim ile yoklama gerekmez ve iş parçacığı ve depolama yönetimi IBM MQtarafından gerçekleştirilir. Örnek programda hatalar konsola yazılır.

Örnek kod aşağıdaki adımları içerir:

1. Tek ileti tüketimi geri çağırma işlevini tanımlayın.

```
void MessageConsumer(MQHCONN   hConn,
MQMD  * pMsgDesc,
MQGMO  * pGetMsgOpts,
MQBYTE * Buffer,
MQCBC  * pContext)
{ ... }
```

2. Kuyruk yöneticisine bağlanın.

```
MQCONN(QMName,&Hcon,&CompCode,&CReason);
```

3. Giriş kuyruklarını açın ve her bir kuyruğu `MessageConsumer` geri çağırma işleviyle ilişkilendirin.

```
MQOPEN(Hcon,&od,O_options,&Hobj,&OpenCode,&Reason);
cbd.CallbackFunction = MessageConsumer;
MQCB(Hcon,MQOP_REGISTER,&cbd,Hobj,&md,&gmo,&CompCode,&Reason);
```

`cbd.CallbackFunction` ' in her kuyruk için ayarlanması gerekmez; bu yalnızca giriş alanıdır. Her bir kuyrukla farklı bir geri çağırma işlevini ilişkilendirebilirsiniz.

4. İletilerin tüketimini başlatın.

```
MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
```

5. Kullanıcının Enter tuşuna basmasını bekleyin ve iletilerin kullanımını durdurun.

```
MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
```

6. Son olarak, kuyruk yöneticisiyle bağlantıyı kesin.

```
MQDISC(&Hcon,&CompCode,&Reason);
```

z/OS *CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örneği z/OS*

Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örnek programları, CICSiçinde zamanuyumsuz tüketim kullanımını ve yayınlama ve abone olma özelliklerini gösterir.

Bir *Kayıt istemcisi* programı, üç Geri Arama işleyicisini (bir olay işleyicisi ve iki ileti tüketicisi) kaydeder ve Zamanuyumsuz Tüketimi başlatır. *İleti alışverişi istemcisi* programı iletileri bir kuyruğa koyar ya da CICS konsolundan, iki ileti tüketicisinin (CSQ4CVCN ve CSQ4CVCT) kullanabilmesi için uygun iletileri yayınlar.

Örneğin işleyişi üzerinde çalıştırma zamanı denetimi sağlamak için, ileti tüketicilerinden biri, aldığı iletiler kullanılarak, Geri Arama işleyicilerinden herhangi biri için SUSPEND, RESUME ya da DEREGISTER komutunu verebilir. Denetim altında Zamanuyumsuz Tüketimi sona erdirmek üzere bir MQCTL STOP yayınlamak için de kullanılabilir. Diğer ileti tüketicisi bir konuya abone olmak için kaydedildi.

Her program, örneğin davranışını görüntülemek için uygun noktalarda COBOL DISPLAY deyimlerini yayınlar.

Uygulamalar şu MQI çağrılarını kullanır:

- MQOPEN
- MQPUT
- MQSUB
- MQGet
- MQCLOSE
- MQCB
- MQCTL

Programlar COBOL dilinde sağlanır. CICS uygulamaları için bkz. CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma Örnekleri .

Bu konuda ayrıca aşağıdaki bilgiler de sağlanır:

- “Kurulum” sayfa 1161
- “Kayıt İstemcisi CSQ4CVRG” sayfa 1162
- “Olay işleyici CSQ4CVEV” sayfa 1162
- “Basit İleti Tüketicisi CSQ4CVCN” sayfa 1162
- “Denetim İletisi Tüketicisi CSQ4CVCT” sayfa 1162
- “İleti Sistemi İstemcisi CSQ4CVPT” sayfa 1162

Kurulum

İleti Tüketicileri tarafından kullanılan Kuyruk ve Konu adları, Kayıt ve İleti Sistemi İstemci programlarında sabit olarak kodlanır.

Kuyruk, **SAMPLE.CONTROL.QUEUE**, örneği çalıştırmadan önce CICS bölgesiyle ilişkilendirilmiş Kuyruk Yöneticisi 'nde tanımlanmalıdır. Gerekirse **Haberler/Ortam/Filmler**konusu tanımlanabilir ya da yürütme sırasında varsayılan Denetim Nesnesi altında yaratılır (yoksa).

CICS programları ve hareket tanımlamaları bir grup kurularak kurulabilir: CSQ4SAMP.

Kayıt İstemcisi CSQ4CVRG

Kayıt İstemcisi programı, CICS işlemi MVRG altında başlatılmalıdır. Giriş gerektirmez.

Kayıt İstemcisi başlatıldığında, MQCB kullanarak aşağıdaki Geri Arama işleyicilerini kaydeder:

- Olay işleyici olarak CSQ4CVEV .
- **Haberler/Medya/Filmler**başlıklı konuda İleti Tüketicisi olarak CSQ4CVCN .
- Kuyruktaki İleti Tüketicisi olarak CSQ4CVCT , **SAMPLE.CONTROL.QUEUE**.

Kayıt İstemcisi, kayıtlı üç Geri Arama işleyicisinin adlarını içeren bir veri yapısını, iki ileti tüketicisiyle ilişkili nesne işleyicileriyle birlikte CSQ4CVCT' ye iletir.

Callback işleyicilerini kaydettirdikten sonra, Kayıt İstemcisi Zamanuyumsuz Tüketimi başlatmak için bir MQCTL START_WAIT yayınlar ve denetim kendisine geri dönünceye kadar askıya alır (örneğin, bir MQCTL STOP yayınlayan Callback işleyicilerinden biri tarafından).

Olay işleyici CSQ4CVEV

Yönlendirildiğinde, Olay İşleyici çağrı tipini gösteren bir ileti görüntüler (örneğin, START). IBM MQ neden kodu CONNECTION_QUIESCING için yönlendirildiğinde, Olay İşleyici Zamanuyumsuz Tüketimi sona erdirmek ve denetimi Kayıt İstemciye döndürmek için bir MQCTL STOP verir.

Basit İleti Tüketicisi CSQ4CVCN

Yönlendirildiğinde, bu İleti Tüketicisi çağrı tipini gösteren bir ileti görüntüler (örneğin, REGISTER). MSG_DRIVEN çağrı tipi için yönlendirildiğinde, İleti Tüketicisi gelen iletiyi alır ve CICS iş günlüğüne çıkarır.

Denetim İletisi Tüketicisi CSQ4CVCT

Yönlendirildiğinde, bu İleti Tüketicisi çağrı tipini gösteren bir ileti görüntüler (örneğin, START). MSG_KALDIRILDI çağrı tipi için yönlendirildiğinde, İleti Tüketicisi, Kayıt İstemcisi tarafından iletilen gelen iletiyi ve veri yapısını alır. İleti içeriğine dayalı olarak, aşağıdakilerden birine uygun MQCB ya da MQCTL komutlarını verir:

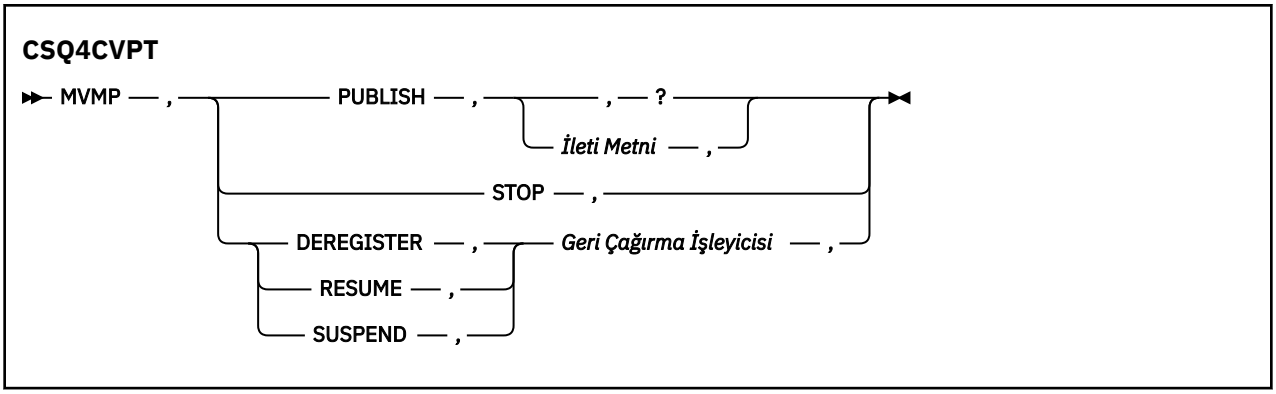
- STOP Zamanuyumsuz Tüketim (denetimi Kayıt İstemciye geri döndürme).
- SUSPEND, RESUME ya da DEREGISTER adlı bir Callback işleyicisi (kendisi dahil).

İleti Sistemi İstemcisi CSQ4CVPT

İleti Sistemi İstemcisi 'nin iki işlevi vardır:

- Bir iletiyi, CSQ4CVCNİleti Tüketicisi tarafından tüketilmek üzere bir konuya yayınlar.
- Bir denetim iletisini CSQ4CVCTDenetim İletisi Tüketicisi tarafından tüketilmek üzere kuyruğa koyar ve bu, örneğin davranışında olası bir değişikliğe neden olur.

Messaging Client programı, CICS hareketi altındaki bir CICS konsolundan başlatılmalı ve komut satırı girişini aşağıdaki sözdizimiyle almalıdır:



YAYINLA
İleti metnini (ya da varsayılan bir iletiyi) Basit İleti Tüketicisi tarafından kullanım için bir Alınmış İleti olarak yayınlayın.

DUR
Zamanuyumsuz Tüketimi Durdur.

DEREGISTER
Adı Belirtilen Geri Çağırma işleyicisinin kaydını kaldır.

Sürdür
Adı Belirtilen Geri Çağırma işleyicisini sürdürün.

Askıya al
Adı Belirtilen Geri Çağırma işleyicisini askıya alın.

Giriş alanları konumlu ve virgülle ayrılmış. Anahtar Sözcükler ve Geri Çağırma İşleyicisi adları büyük ve küçük harfe duyarlı değildir.

Örnekler:

Çizelge 186. Giriş örnekleri

Örnek	**Açıklama**
MVMP, YAYıNLA,,	Varsayılan ileti yayınla
MVMP,publish, `A short message`,	Verili metni yayınla
MVMP, DUR,	Zamanuyumsuz Tüketimi Durdur
MVMP,DEREGISTER,CSQ4CVEV,	Olay İşleyicinin kaydını kaldır
MVMP,resume,csq4cvcn,	Basit İleti Tüketiciyi Sürdür
MVMP,SUSPEND,CSQ4CVEV,	Olay Işleyicinin Askıya Alınması

Burada MVMP, CSQ4CVPTMessaging Client programıyla ilişkilendirilmiş CICS hareketin.

Not:

- Tüm Geri Çağırma işleyicilerinin askıya alınması ya da kayıttan çıkartılması, Kayıt İstemcisi tarafından yayınlanan START_WAIT 'i sonlandırır, denetimi buna geri döndürür ve görevi sona erdirir.
- Denetim Geri Çağırma İşleyicisinin askıya alınması ya da kayıttan kaldırılması kasıtlı olarak önlenmedi, ancak örneğin davranışını daha fazla denetleme yeteneğini kaldırır.

z/OS *z/OS üzerinde Yayınla/Abone ol örneği*

Yayınla/Abone Ol örnek programları, IBM MQiçindeki yayınlama ve abone olma özelliklerinin kullanımını gösterir.

IBM MQ Yayınla/Abone Ol arabirimine nasıl programlanacağını gösteren dört C ve iki COBOL programlama dili örnek programı vardır. Programlar C ve COBOL dilinde sağlanır. Uygulamalar toplu iş ortamında çalışır; toplu iş uygulamaları için Yayınlama/Abone Olma Örnekleri konusuna bakın.

CICS ortamında çalışan COBOL örnekleri de vardır; bkz. “CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örneği z/OS” sayfa 1161.

Bu konuda ayrıca, Yayınlama/Abone Olma örnek programlarının nasıl çalıştırılacağına ilişkin bilgiler de sağlanır. Bu örnek programlar şunlardır:

CSQ4BCP1 örneğinin çalıştırılması

Bu program C dilinde yazılmıştır; iletileri bir konuya yayınlar. Bu programı çalıştırmadan önce abone örneklerinden birini başlatın.

Bu program en çok dört parametreye kadar sürer:

1. Hedef konu dizgisinin adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. Açma seçenekleri (isteğe bağlı).
4. Seçenekleri kapat (isteğe bağlı).

Bir kuyruk yöneticisi belirtilmezse, CSQ4BCP1 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için örnek bir JCL vardır, CSQ4BCPP' de bulunur.

İleti içeriği standart giriş yoluyla sağlanır (**`SYSIN DD`**).

CSQ4BCP2 örneğinin çalıştırılması

Bu program C harfiyle yazılır; bir konuya abone olur ve alınan iletileri yazdırır.

Bu program en çok üç parametre alır:

1. Hedef konu dizgisinin adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. MQSD abonelik seçenekleri (isteğe bağlı).

Bir kuyruk yöneticisi belirtilmezse, CSQ4BCP2 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için örnek bir JCL var, CSQ4BCPS' de bulunuyor.

CSQ4BCP3 örneğinin çalıştırılması

Bu program C dilinde yazılır; kullanıcı tarafından belirlenen bir hedef kuyruk kullanılarak bir konuya abone olur ve alınan iletileri yazdırır.

Bu program en çok dört parametreye kadar sürer:

1. Hedef konu dizgisinin adı (gerekli).
2. Hedefin adı (gerekli).
3. Kuyruk yöneticisinin adı (isteğe bağlı).
4. MQSD abonelik seçenekleri (isteğe bağlı).

Bir kuyruk yöneticisi belirtilmezse, CSQ4BCP3 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için örnek bir JCL vardır, CSQ4BCPD' de bulunur.

CSQ4BCP4 örneğinin çalıştırılması

Bu program C dilinde yazılır; MQSUB çağrısında genişletilmiş seçeneklerin kullanılmasına izin veren bir konuya abone olur ve bu konudan iletileri alır ve daha basit MQSUB örneğinde kullanılabilir olanları genişletir: CSQ4BCP2. İleti bilgi yüküne ek olarak, her iletiye ilişkin ileti özellikleri de alınır ve görüntülenir.

Bu program değişken bir parametre kümesi alır:

- **-t** *Topic string*.
- **-o** *Topic object name*.

  Önemli: -t ya da **-o**ya da her ikisinden biri gereklidir
- **-m** *Queue manager name* (isteğe bağlı).
- **-b** *Connection binding type* (isteğe bağlı); burada *tip* aşağıdaki değerlerden herhangi birine sahip olabilir:
  - *standard* : Varsayılan değer olan MQCNO_STANDARD_BINDING
  - *paylaşılan*: MQCNO_SHARED_BINDING
  - *fastpath*: MQCNO_FASTPATH_BINDING
  - *yalıtılmış*: MQCNO_ISOLATED_BINDING
- **-q** *Destination queue name* (isteğe bağlı).
- **-w** *Wait interval on MQGET in seconds* (isteğe bağlı); burada *saniye* aşağıdaki değerlerden herhangi birine sahip olabilir:
  - unlimited: MQWI_UNLIMITED
  - none: Bekleme yok
  - *n*: Bekleme aralığı (saniye)
  - Değer belirtilmedi: Değer belirtilmediğinde, varsayılan değer 30 saniyedir
- **-d** *Subscription name* (isteğe bağlı). Sürekli abonelik olarak adlandırılan abonelikleri yaratır ya da sürdürür.
- **-k** (isteğe bağlı). MQCLOSE üzerinde sürekli aboneliği korur.

Bir kuyruk yöneticisi belirtilmezse, CSQ4BCP4 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için örnek bir JCL vardır, CSQ4BCPE' de bulunur.

CSQ4BVP1 örneğinin çalıştırılması

Bu program COBOL dilinde yazılmıştır, iletileri bir konuya yayınlar. Bu programı çalıştırmadan önce abone örneklerinden birini başlatın.

Bu program parametre almıyor. **SYSIN DD** , giriş konusu adını, kuyruk yöneticisi adını ve ileti içeriğini sağlar.

Bir kuyruk yöneticisi belirtilmezse, CSQ4BVP1 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için örnek bir JCL var, CSQ4BVPP' de bulunuyor.

CSQ4BVP2 örneğinin çalıştırılması

Bu program COBOL dilinde yazılır, bir konuya abone olur ve alınan iletileri yazdırır.

Bu program parametre almıyor. **SYSIN DD** , konu adı ve kuyruk yöneticisi adı için giriş sağlar.

Bir kuyruk yöneticisi belirtilmezse, CSQ4BVP1 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için örnek bir JCL var, CSQ4BVPP' de bulunuyor.

z/OS *z/OS üzerindeki Set and Inquire message özelliği örneği*

İleti özelliği örnek programları, kullanıcı tanımlı özelliklerin bir ileti tanıtıcısına eklenmesini ve o iletiyle ilişkili özelliklerin engizisyonunu gösterir.

Uygulamalar şu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQPUT
- MQGet
- MQCLOSE
- MQDISC
- MQCRTMH
- MQDLTMH
- MQINQMP
- MQSETMP

Programlar C dilinde teslim edilir. Uygulamalar toplu iş ortamında çalışır. Toplu iş uygulamaları için Diğer örnekler konusuna bakın.

CSQ4BCM1 programı, ileti kuyruğundaki bir ileti tanıtıcısının özelliklerini sorgulamak için kullanılır ve MQINQMP API çağrısının kullanımına bir örnektir. Örnek, kuyruktan bir ileti alır ve tüm ileti tanıtıcısı özelliklerini yazdırır.

CSQ4BCM2 programı, ileti kuyruğundaki bir ileti tanıtıcısının özelliklerini ayarlamak için kullanılır ve MQSETMP API çağrısının kullanımına bir örnektir. Örnek bir ileti tanıtıcısı yaratır ve bunu MQGMO yapısının `MsgHandle` alanına koyar. Daha sonra iletiyi bir kuyruğa koyar.

İleti sorma ve yazdırma özelliklerinin diğer örnekleri, CSQ4BCG1 ve CSQ4BCP4 örnek programlarında yer alır.

Bu konuda, aşağıdaki başlıklar altında Set ve Inquire ileti özelliği örneklerinin çalıştırılmasına ilişkin bilgiler de sağlanır:

- "CSQ4BCM1 örneğinin çalıştırılması" sayfa 1166
- "CSQ4BCM2 örneğinin çalıştırılması" sayfa 1166

CSQ4BCM1 örneğinin çalıştırılması

Bu program en çok dört parametreye kadar sürer:

1. Hedef kuyruğun adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. Açma seçenekleri (isteğe bağlı).
4. Seçenekleri kapat (isteğe bağlı).

CSQ4BCM2 örneğinin çalıştırılması

Bu program en çok altı parametre alır:

1. Hedef kuyruğun adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. Açma seçenekleri (isteğe bağlı).
4. Seçenekleri kapat (isteğe bağlı).
5. Hedef kuyruk yöneticisinin adı (isteğe bağlı).
6. Dinamik kuyruğun adı (isteğe bağlı).

Özellik adları, değerler ve ileti içeriği, standart giriş (**`SYSIN DD`**) aracılığıyla sağlanır. Programı çalıştırmak için örnek bir JCL var, CSQ4BCMP' de bulunuyor.

Managed File Transfer için uygulama geliştirilmesi

Managed File Transferile çalıştırılacak programları belirtin, Managed File Transferile Apache Ant kullanın, kullanıcı çıkışlarıyla Managed File Transfer 'u özelleştirin ve Managed File Transfer ' u aracı komut kuyruğuna iletileri koyarak denetleyin.

MFT ile çalıştırılacak programları belirtme

Programları, Managed File Transfer Agent ' un çalıştığı bir sistemde çalıştırabilirsiniz. Dosya aktarma isteğinin bir parçası olarak, aktarma başlamadan önce ya da sona erdikten sonra çalıştırılacak bir program belirleyebilirsiniz. Ayrıca, yönetilen bir çağrı isteği göndererek, dosya aktarma isteğinin parçası olmayan bir programı da başlatabilirsiniz.

Bu görev hakkında

Çalıştırılacak programı belirleyebileceğiniz beş senaryo vardır:

- Aktarma isteğinin bir parçası olarak, aktarım başlamadan önce kaynak aracıda.
- Aktarma isteğinin bir parçası olarak, aktarım başlamadan önce hedef aracıda.
- Aktarma isteğinin bir parçası olarak, aktarım tamamlandıktan sonra kaynak aracıda.
- Aktarma isteğinin bir parçası olarak, aktarım tamamlandıktan sonra hedef aracıda.
- Transfer isteğinin bir parçası olarak değil. Bir programı çalıştırmak için aracıya bir istek gönderebilirsiniz. Bu senaryoya bazen yönetilen çağrı denir.

Kullanıcı çıkışları ve program çağrıları aşağıdaki sırayla çağrılır:

```
- SourceTransferStartExit(onSourceTransferStart).
- PRE_SOURCE Command.
- DestinationTransferStartExits(onDestinationTransferStart).
- PRE_DESTINATION Command.
- The Transfer request is performed.
- DestinationTransferEndExits(onDestinationTransferENd).
- POST_DESTINATION Command.
- SourceTransferEndExits(onSourceTransferEnd.
- POST_SOURCE Command.
```

Notlar:

1. **`DestinationTransferEndExits`** yalnızca aktarma başarıyla ya da kısmen tamamlandığında çalıştırılır.
2. **`postDestinationCall`** yalnızca aktarma başarıyla ya da kısmen tamamlandığında çalıştırılır.
3. **`SourceTransferEndExits`** başarılı, kısmen başarılı ya da başarısız aktarımlar için çalıştırılır.
4. **`postSourceCall`** yalnızca aşağıdaki durumda çağrılır:
   - Aktarım iptal edilmedi.
   - Başarılı ya da kısmen başarılı bir sonuç vardır.
   - Hedef sonrası aktarım programları başarıyla çalıştırıldı.

Yordam

- Aşağıdaki seçeneklerden birini kullanarak çalıştırmak istediğiniz programı belirleyin:

  Apache Ant görevi kullanma

  Bir programı başlatmak için `fte:filecopy`, `fte:filemove`ve `fte:call` Ant görevlerinden birini kullanın. Bir Ant görevini kullanarak, `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrcve` `fte:command` iç içe geçmiş öğelerini kullanarak beş senaryodan herhangi birinde bir program belirtebilirsiniz. Daha fazla bilgi için bkz. Program çağırma içiçe öğeleri.

Dosya aktarma isteği iletisini düzenle

Bir aktarma isteğiyle oluşturulan XML ' i düzenleyebilirsiniz. Bu yöntemi kullanarak, XML dosyasına **preSourceCall**, **postSourceCall**, **preDestinationCall**, **postDestinationCall**ve **managedCall** öğelerini ekleyerek beş senaryodan herhangi birinde bir program çalıştırabilirsiniz. Daha sonra, bu değiştirilen XML dosyasını yeni bir dosya aktarma isteğine ilişkin aktarma tanımlaması olarak (örneğin, **fteCreateTransfer -td** parametresi) kullanın. Daha fazla bilgi için bkz. MFT aracısı çağrı isteği iletisi örnekleri.

fteCreateTransfer komutunu kullanma

Başlatılacak programları belirlemek için **fteCreateTransfer** komutunu kullanabilirsiniz. Bu komutu, aktarma isteğinin bir parçası olarak, ilk dört senaryoda çalıştırılacak programları belirtmek için kullanabilirsiniz, ancak yönetilen bir çağrı başlatamayabilirsiniz. Kullanılacak parametrelerle ilgili bilgi için bkz. **fteCreateTransfer**: Yeni bir dosya aktarımı başlatma. Bu komutun kullanılmasına ilişkin örnekler için fteCreateTransfer to start programsbaşlıklı konuya bakın.

İlgili başvurular

commandPath MFT özelliği

Yönetilen çağrılar

Managed File Transfer (MFT) aracıları genellikle dosyaları ya da iletileri aktarmak için kullanılır. Bunlar *Yönetilen Aktarmalar*olarak bilinir. Aracılar, dosyaları ya da iletileri aktarmaya gerek kalmadan komutları, komut dosyalarını ya da JCL ' yi çalıştırmak için de kullanılabilir. Bu yetenek *Yönetilen Çağrılar*olarak bilinir.

Yönetilen arama istekleri bir aracıya çeşitli şekillerde gönderilebilir:

- fte: call Ant görevinikullanarak.
- Bir komutu ya da komut dosyasını çalıştıran bir görev XML ' i ile kaynak izleyicisinin yapılandırılması. Ek bilgi için Komutları ve komut dosyalarını başlatmak için izleme görevlerinin yapılandırılması başlıklı konuya bakın.
- Aracının komut kuyruğuna doğrudan bir XML iletisi yerleştiriliyor. Yönetilen Çağrı XML şemasına ilişkin ek bilgi için Dosya aktarma isteği iletisi biçimi konusuna bakın.

Yönetilen çağrılar için, çalıştırılmakta olan komut ya da komut dosyasını içeren dizinin **commandPath**aracı özelliğinde belirtilmesi gerekir.

Yönetilen çağrılar, aracının **commandPath**dizininde belirtilmeyen dizinlerde bulunan komutları ya da komut dosyalarını çalıştıramaz. Bu, aracının herhangi bir kötü niyetli kod çalıştırmadığından emin olmak için.

Önemli: commandPathdeğerini belirttiğinizde varsayılan olarak bunun böyle olduğundan emin olmak için:

- Var olan herhangi bir aracı kum havuzu, başlatılırken aracı tarafından yapılandırılır; böylece tüm **commandPath** dizinleri, bir aktarma için erişim verilmeyen dizinler listesine otomatik olarak eklenir.
- Aracı başlatıldığında var olan tüm kullanıcı kum havuzları güncellenir; böylece, tüm **commandPath** dizinleri (ve bunların alt dizinleri) <read> ve <write> öğelerine <exclude> öğeleri olarak eklenir.
- Aracı, bir aracı kum havuzunu ya da kullanıcı kum havuzlarını kullanacak şekilde yapılandırılmadıysa, aracı **commandPath** dizinlerini reddedilen dizinler olarak belirten yeni bir aracı kum havuzu başlatıldığında yeni bir aracı kum havuzu yaratılır.

Ayrıca, yalnızca yetkili kullanıcıların yönetilen çağrı istekleri göndermesine izin verildiğinden emin olmak için bir aracı üzerinde yetki denetimini de etkinleştirebilirsiniz. Bu konuda daha fazla bilgi için MFT aracı işlemlerinde kullanıcı yetkilerini kısıtlamabaşlıklı konuya bakın.

Yönetilen bir çağrım parçası olarak çağrılan komut, komut dosyası ya da JCL, aracı tarafından izlenen bir dış işlem olarak çalışır. Süreç sona erdiğinde, yönetilen çağrı tamamlanır ve süreçten gelen dönüş kodu, **fte:call** Ant görevini çağıran aracı ya da Ant komut dosyası tarafından kullanılabilir duruma getirilebilir.

Yönetilen çağrı **fte:call** Ant görevi tarafından başlatıldıysa, Ant komut dosyanız, yönetilen aramanın başarılı olup olmadığını belirlemek için dönüş kodunun değerini denetleyebilir.

Yönetilen çağrıların diğer tüm tiplerinde, yönetilen çağrıın başarıyla tamamlandığını belirtmek için hangi dönüş kodu değerlerinin kullanılacağını belirtebilirsiniz. Aracı, dış işlem tamamlandığında işlemdeki dönüş kodunu bu dönüş kodlarıyla karşılaştırır.

Not: Yönetilen çağrılar dış işlemler olarak çalıştığından, başlatıldıktan sonra iptal edilemezler.

Yönetilen çağrılar ve kaynak aktarım yuvaları

Bir aracı, Advanced agent properties: Transfer limitbaşlıklı konuda açıklanan **maxSourceTransfers**aracı özelliğinde belirtildiği gibi bir dizi kaynak aktarım yuvası içerir.

Yönetilen bir çağrı ya da yönetilen bir aktarım çalıştırıldığında, bunlar bir kaynak aktarım yuvasını işgal eder. Yuva, yönetilen çağrı ya da yönetilen aktarım tamamlandığında serbest bırakılır.

Bir aracı yeni bir yönetilen çağrı ya da yönetilen aktarma isteği aldığında tüm kaynak aktarım yuvaları kullanılırsa, istek, bir yuva kullanılabilir oluncaya kadar aracı tarafından kuyruğa alınır.

Yönetilen bir çağrı yönetilen bir aktarımı başlatırsa (örneğin, yönetilen bir çağrı bir Ant komut dosyası çalıştırıyorsa ve Ant komut dosyası bir dosyayı aktarmak için fte: filecopy ya da fte: filetaşıma görevini kullanıyorsa), iki kaynak aktarım yuvası gerekir:

- Yönetilen aktarım için bir tane
- Yönetilen çağrı için bir tane

Bu durumda, yönetilen aktarımının tamamlanması ya da kurtarılması uzun sürerse, yönetilen aktarım tamamlanıncaya, iptal edilinceye ya da bir **transferRecoveryTimeout**nedeniyle zamanaşımına uğrayıncaya kadar iki kaynak aktarım yuvasının dolu olduğunu unutmayın. **transferRecoveryTimeout**ile ilgili ayrıntılar için Aktarma kurtarma zamanaşımı kavramları konusuna bakın. Bu, aracının işleyebileceği diğer yönetilen aktarımların ya da yönetilen çağrıların sayısını sınırlayabilir.

Bu nedenle, kaynak aktarım yuvalarını uzun süre meşgul etmediğinden emin olmak için yönetilen bir çağrı tasarımını göz önünde bulundurmanız gerekir.

REST API olanağının yönetilen çağrılarla kullanılması

V 9.3.0 V 9.3.0

HTTP GET ve HTTP POST fiilleri, yönetilen çağrıları etkinleştirmek ve yalnızca REST APISürüm 3 üzerinde çalışmak için desteklenir.

HTTP DELETE ve HTTP UPDATE gibi diğer fiiller desteklenmez ve bunları kullanmayı denerseniz HTTP 405 hata kodunu döndürür.

Uyarı: Yönetilen bir arama gönderildikten sonra REST APIkullanılarak iptal edilemez.

Apache Ant ' yi MFT ile kullanma

Managed File Transfer , dosya aktarma işlevini Apache Ant aracıyla bütünleştirmek için kullanabileceğiniz görevleri sağlar.

Önceden yapılandırdığınız bir Managed File Transfer ortamında Ant görevlerini çalıştırmak için **fteAnt** komutunu kullanabilirsiniz. Yorumlanan bir komut dosyası dilinden karmaşık dosya aktarma işlemlerini koordine etmek için Ant komut dosyalarınızdan dosya aktarma Ant görevlerini kullanabilirsiniz.

Apache Anthakkında daha fazla bilgi için Apache Ant proje web sayfasına bakın: https://ant.apache.org/

İlgili kavramlar

“MFT ile Ant komut dosyalarını kullanmaya başlama” sayfa 1170
Ant komut dosyalarının Managed File Transfer ile kullanılması, yorumlanan bir komut dosyası oluşturma dilinden karmaşık dosya aktarma işlemlerini koordine etmenizi sağlar.

fteAnt: Ant görevlerini MFT içinde çalıştırma

İlgili başvurular

“MFT için örnek Ant görevleri” sayfa 1171

Managed File Transferkurulumunuzla birlikte sağlanan bir dizi örnek Ant komut dosyası vardır. Bu örnekler *MQ_INSTALLATION_PATH*/mqft/samples/fteantdizininde bulunur. Her örnek komut dosyası bir `init` hedefi içerir, bu komut dosyalarını yapılandırmanızla çalıştırmak için `init` hedefinde ayarlanan özellikleri düzenleyin.

MFT ile Ant komut dosyalarını kullanmaya başlama

Ant komut dosyalarının Managed File Transfer ile kullanılması, yorumlanan bir komut dosyası oluşturma dilinden karmaşık dosya aktarma işlemlerini koordine etmenizi sağlar.

Ant komut dosyaları

Ant komut dosyaları (ya da oluşturma dosyaları), bir ya da daha fazla hedefi tanımlayan XML belgeleridir. Bu hedefler, çalıştırılacak görev öğelerini içerir. Managed File Transfer , dosya aktarma işlevini Apache Antile bütünleştirmek için kullanabileceğiniz görevleri sağlar. Ant komut dosyaları hakkında bilgi edinmek için Apache Ant proje web sayfasına bakın: https://ant.apache.org/

Managed File Transfer görevlerini kullanan Ant komut dosyaları örnekleri, *MQ_INSTALLATION_PATH*/mqft/samples/fteant dizininde ürün kuruluşunuzla birlikte sağlanır.

Protokol köprüsü aracılarında, Ant komut dosyaları protokol köprüsü aracısı sisteminde çalıştırılır. Bu Ant komut dosyalarının FTP ya da SFTP sunucusundaki dosyalara doğrudan erişimi yoktur.

Ad alanı

Ad alanı, dosya aktarımı Ant görevlerini, aynı adı paylaşabilecek diğer Ant görevlerinden ayırmak için kullanılır. Ad alanını Ant komut dosyanızın proje etiketinde tanımlarsınız.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="do_ping">

  <target name="do_ping">
     <fte:ping cmdqm="qm@localhost@1414@SYSTEM.DEF.SVRCONN" agent="agent1@qm1"
      rcproperty="ping.rc" timeout="15"/>
  </target>

</project>
```

`xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs"` özniteliği, Ant'e `com.ibm.wmqfte.ant.taskdefs`kitaplığında `fte` öneki olan görevlerin tanımlarını aramasını söyler.

Ad alanı öneki olarak `fte` kullanmanıza gerek yoktur; herhangi bir değeri kullanabilirsiniz. Ad alanı öneki `fte` tüm örneklerde ve örnek Ant komut dosyalarında kullanılır.

Ant komut dosyalarını çalıştırma

Dosya aktarma Ant görevlerini içeren Ant komut dosyalarını çalıştırmak için **fteAnt** komutunu kullanın. Örneğin:

```
fteAnt -file ant_script_location/ant_script_name
```

Daha fazla bilgi için bkz. **fteAnt**: Ant görevlerini MFTiçinde çalıştırma.

Dönüş kodları

Dosya aktarma Ant görevleri, Managed File Transfer komutlarıyla aynı dönüş kodlarını döndürür. Daha fazla bilgi için bkz. MFTiçin dönüş kodları.

İlgili başvurular

fteAnt: Ant görevlerini MFT içinde çalıştırma

“MFT için örnek Ant görevleri” sayfa 1171
Managed File Transferkurulumunuzla birlikte sağlanan bir dizi örnek Ant komut dosyası vardır. Bu örnekler *MQ_INSTALLATION_PATH*/mqft/samples/fteantdizininde bulunur. Her örnek komut dosyası bir `init` hedefi içerir, bu komut dosyalarını yapılandırmanızla çalıştırmak için `init` hedefinde ayarlanan özellikleri düzenleyin.

MFT için örnek Ant görevleri

Managed File Transferkurulumunuzla birlikte sağlanan bir dizi örnek Ant komut dosyası vardır. Bu örnekler *MQ_INSTALLATION_PATH*/mqft/samples/fteantdizininde bulunur. Her örnek komut dosyası bir `init` hedefi içerir, bu komut dosyalarını yapılandırmanızla çalıştırmak için `init` hedefinde ayarlanan özellikleri düzenleyin.

e-posta

E-posta örneği, bir dosyayı aktarmak ve aktarım başarısız olursa belirli bir e-posta adresine e-posta göndermek için Ant görevlerinin nasıl kullanılacağını gösterir. Komut dosyası, kaynak ve hedef aracıların etkin olup olmadığını ve Managed File Transfer ping görevini kullanarak aktarımları işleyebilip işleyemediğini denetler. Her iki aracı da etkinse, komut dosyası özgün dosyayı silmeden kaynak ve hedef aracılar arasında bir dosyayı aktarmak için Managed File Transfer fte: filecopy görevini kullanır. Aktarım başarısız olursa, komut dosyası standart Ant e-posta görevini kullanarak hatayla ilgili bilgi içeren bir e-posta gönderir.

hub

hub örneği iki komut dosyasından oluşmuştur: `hubcopy.xml` ve `hubprocess.xml.` `hubcopy.xml` komut dosyası, 'hub and spoke' stil topolojilerini oluşturmak için Ant komut dosyasını nasıl kullanabileceğinizi gösterir. Bu örnekte, iki dosya, konuşlu makinelerde çalışan aracılardan merkez makinesinde çalışan bir aracıya aktarılır. Her iki dosya da aynı anda aktarılır ve aktarımlar tamamlandığında dosyaları işlemek için hub makinesinde `hubprocess.xml` Ant komut dosyası çalıştırılır. Her iki dosya da doğru aktarılırsa, Ant komut dosyası dosyaların içeriğini birleştirir. Dosyalar doğru aktarılmazsa, Ant komut dosyası aktarılan dosya verilerini silerek temizlenir. Bu örneğin doğru çalışması için, `hubprocess.xml` komut dosyasını Hub aracısının komut yoluna koymanız gerekir. Bir aracının komut yolunu ayarlama hakkında daha fazla bilgi için bkz. commandPath MFT özelliği.

librarytransfer (yalnızcaIBM i platformu)

IBM i

IBM i Librarytransfer örneği, bir IBM i sisteminde bir IBM i kitaplığını ikinci IBM i sistemine aktarmak için Ant görevlerinin nasıl kullanılacağını gösterir.

IBM i Librarytransfer örneği, iki IBM i sistemi arasında yerel kitaplık nesnelerini aktarmak için Managed File Transfer içinde önceden tanımlanmış Ant görevlerle IBM i üzerinde yerel saklama dosyası desteğini kullanır. Örnek, kaynak aracı sisteminde istenen kitaplığı geçici bir saklama dosyasına kaydeden bir yürütülür komut dosyasını çağırmak için Managed File Transfer filecopy görevinde `librarysave.sh` < presrc> içiçe yerleşimli bir öğe kullanır. Saklama dosyası, dosya nesnesi ant görevi tarafından hedef aracı sistemine taşınır; burada < postdst> içiçe yerleşimli bir öğe, `libraryrestore.sh` yürütülür komut dosyasını çağırmak için kullanılır. Bu komut dosyası, saklama dosyasına kaydedilen kitaplığı hedef sisteme geri yüklemek için kullanılır.

IBM i Bu örneği çalıştırmadan önce, `librarytransfer.xml` dosyasında açıklandığı gibi bazı yapılandırmayı tamamlamanız gerekir. İki IBM i makinesinde çalışan bir Managed File Transfer ortamınız da olmalıdır. Kurulum, birinci IBM i makinesinde çalışan bir kaynak aracıdan ve ikinci IBM i makinesinde çalışan bir hedef aracıdan oluşmalıdır. İki ajan birbirleriyle iletişim kurabilmelidir.

IBM i Librarytransfer örneği aşağıdaki üç dosyadan oluşur:

- `librarytransfer.xml`

- `librarysave.sh` (< presrc> yürütülebilir komut dosyası)
- `libraryrestore.sh` (< postdst> yürütülür komut dosyası)

Örnek dosyalar şu dizinde bulunur: `/QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/librarytransfer`

IBM i Bu örneği çalıştırmak için kullanıcı aşağıdaki adımları tamamlamalıdır:

1. Bir Qshell oturumu başlatın. IBM i komut penceresinde şunu yazın: STRQSH
2. Dizini `bin` dizinine aşağıdaki gibi değiştirin:

```
cd  /QIBM/ProdData/WMQFTE/V7/bin
```

3. Gerekli yapılandırmayı tamamladıktan sonra, aşağıdaki komutu kullanarak örneği çalıştırın:

```
fteant -f /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/librarytransfer/librarytransfer.xml
```

physicalfiletransfer (yalnızcaIBM i platformu)

IBM i Physicalfiletransfer örneği, bir Kaynak Fiziksel ya da Veritabanı dosyasını tek bir IBM i sistemindeki bir kitaplıktan ikinci bir IBM i sistemindeki bir kitaplığa aktarmak için Ant görevlerinin nasıl kullanılacağını gösterir.

IBM i Physicalfiletransfer örneği, iki IBM i sistemi arasında tam Kaynak Fiziksel ve Veritabanı dosyalarını aktarmak için Managed File Transfer içinde önceden tanımlanmış Ant Görevleriyle IBM i üzerinde yerel saklama dosyası desteğini kullanır. Örnek, kaynak aracı sistemindeki bir kitaplıktan istenen Kaynak Fiziksel ya da Veritabanı dosyasını geçici bir saklama dosyasına kaydetmek için Managed File Transfer yürütülebilir bir komut dosyasını çağırmak için dosya dosyası görevi içindeki bir < presrc> içiçe yerleştirilmiş öğesini `physicalfilesave.sh` kullanır. Saklama dosyası, filecopy ant görevi tarafından, yürütülür komut dosyasını çağırmak için < postdst> içiçe yerleştirilmiş bir öğenin kullanıldığı hedef aracı sistemine taşınır `physicalfilerestore.sh` daha sonra, saklama dosyasının içindeki dosya nesnesini hedef sistemde belirtilen bir kitaplığa geri yükler.

IBM i Bu örneği çalıştırmadan önce, `physicalfiletransfer.xml` dosyasında açıklandığı gibi bazı yapılandırmayı tamamlamanız gerekir. İki IBM i sisteminde de çalışan bir Managed File Transfer ortamınız olmalıdır. Kurulum, birinci IBM i sisteminde çalışan bir kaynak aracıdan ve ikinci IBM i sisteminde çalışan bir hedef aracıdan oluşmalıdır. İki ajan birbirleriyle iletişim kurabilmelidir.

IBM i Physicalfiletransfer örneği aşağıdaki üç dosyadan oluşur:

- `physicalfiletransfer.xml`
- `physicalfilesave.sh` (< presrc> yürütülebilir komut dosyası)
- `physicalfilerestore.sh` (< postdst> yürütülür komut dosyası)

Örnek dosyalar şu dizinde bulunur: `/QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/physicalfiletransfer`

IBM i Bu örneği çalıştırmak için kullanıcı aşağıdaki adımları tamamlamalıdır:

1. Bir Qshell oturumu başlatın. IBM i komut penceresinde şunu yazın: STRQSH
2. Dizini `bin` dizinine aşağıdaki gibi değiştirin:

```
cd  /QIBM/ProdData/WMQFTE/V7/bin
```

3. Gerekli yapılandırmayı tamamladıktan sonra, aşağıdaki komutu kullanarak örneği çalıştırın:

```
fteant -f /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/physicalfiletransfer/
physicalfiletransfer.xml
```

zaman aşımı

Zamanaşımı örneği, bir dosya aktarma girişiminde bulunmak için Ant görevlerinin nasıl kullanılacağını ve belirtilen bir zamanaşımı değerinden uzun sürerse, aktarma işleminin iptal edileceğini gösterir. Komut dosyası, Managed File Transfer fte: filecopy görevini kullanarak bir dosya aktarımı başlatır. Bu aktarımın sonucu ertelendi. Komut dosyası, aktarımın tamamlanması için belirli bir saniye beklemek üzere Managed File Transfer fte: awaitoutcome Ant task görevini kullanır. Aktarma işlemi belirli bir süre içinde tamamlanmazsa, dosya aktarımını iptal etmek için Managed File Transfer fte: cancel Ant task (fte: İptal) görevi kullanılır.

vsamtransfer

z/OS

z/OS vsamtransfer örneği, Managed File Transferkomutunu kullanarak bir VSAM veri kümesinden başka bir VSAM veri kümesine aktarmak için Ant görevlerinin nasıl kullanılacağını gösterir. Managed File Transfer şu anda VSAM veri kümelerinin aktarılmasını desteklemez. Örnek komut dosyası, yürütülür dosyayı (`datasetcopy.sh`) çağırmak için `presrc` Program çağırma içiçe yerleştirilmiş öğeleri kullanarak VSAM veri kayıtlarını sıralı bir veri kümesine yükler. Komut dosyası, sıralı veri kümesini kaynak aracıdan hedef aracıya aktarmak için Managed File Transfer fte: filetaşıma görevini kullanır. Daha sonra komut dosyası, `loadvsam.jcl` komut dosyasını çağırmak için `postdst` Program çağırma içiçe yerleştirilmiş öğelerini kullanır. Bu JCL komut dosyası, aktarılan veri kümesi kayıtlarını hedef VSAM veri kümesine yükler. Bu örnek, bu dil seçeneğini göstermek için hedef çağrı için JCL kullanır. Aynı sonuç, bunun yerine ikinci bir kabuk komut dosyası kullanılarak da elde edilebilir.

z/OS Bu örnek, kaynak ve hedef veri kümelerinin VSAM olmasını gerektirmez. Kaynak ve hedef veri kümeleri aynı tipdeyse, örnek herhangi bir veri kümesi için çalışır.

z/OS Bu örneğin doğru çalışması için, `datasetcopy.sh` komut dosyasını kaynak aracının komut yoluna ve `loadvsam.jcl` komut dosyasını hedef aracının komut yoluna koymanız gerekir. Bir aracının komut yolunu ayarlama hakkında daha fazla bilgi için bkz. commandPath MFT özelliği.

PostaKodu

zip örneği, şu iki komut dosyalarından oluşmuştur: `zip.xml` ve `zipfiles.xml`. Örnek, bir dosya aktarma taşıma işlemini gerçekleştirmeden önce bir Ant komut dosyasını çalıştırmak için Managed File Transfer fte: filetaşıma görevinin içindeki `presrc` içiçe yerleşimli öğenin nasıl kullanılacağını gösterir. `zip.xml` komut dosyasında `presrc` içiçe yerleşimli öğesi tarafından çağrılan `zipfiles.xml` komut dosyası, bir dizinin içeriğini sıkıştırır. `zip.xml` komut dosyası sıkıştırılmış dosyayı aktarır. Bu örnek, kaynak aracının komut yolunda `zipfiles.xml` Ant komut dosyasının bulunmasını gerektirir. Bunun nedeni, `zipfiles.xml` Ant komut dosyasının kaynak aracıdaki dizinin içeriğini sıkıştırmak için kullanılan hedefi içermesi olabilir. Bir aracının komut yolunu ayarlama hakkında daha fazla bilgi için bkz. commandPath MFT özelliği.

İlgili kavramlar

“MFT ile Ant komut dosyalarını kullanmaya başlama” sayfa 1170
Ant komut dosyalarının Managed File Transfer ile kullanılması, yorumlanan bir komut dosyası oluşturma dilinden karmaşık dosya aktarma işlemlerini koordine etmenizi sağlar.

İlgili başvurular

fteAnt: Ant görevlerini MFT içinde çalıştırma

MFT ürününü kullanıcı çıkışlarıyla özelleştirme

Kullanıcı çıkışı yordamları olarak bilinen kendi programlarınızı kullanarak Managed File Transfer özelliklerini özelleştirebilirsiniz.

Önemli: Bir kullanıcı çıkışı içindeki herhangi bir kod IBMtarafından desteklenmez ve bu kodla ilgili herhangi bir sorunun başlangıçta şirketiniz ya da çıkışı sağlayan satıcı firma tarafından araştırılması gerekir.

Managed File Transfer , kod içinde Managed File Transfer ' in denetimi yazdığınız bir programa (bir kullanıcı çıkışı yordamı) iletebileceği noktaları sağlar. Bu noktalar kullanıcı çıkış noktaları olarak bilinir. Daha sonra Managed File Transfer , programınız çalışmasını bitirdiğinde denetimi sürdürebilir. Kullanıcı çıkışlarından herhangi birini kullanmanız gerekmez, ancak belirli gereksinimlerinizi karşılayacak şekilde Managed File Transfer sisteminizin işlevini genişletmek ve özelleştirmek istiyorsanız bunlar kullanışlıdır.

Dosya aktarma işlemi sırasında, kaynak sistemde bir kullanıcı çıkışını ve dosya aktarma işlemi sırasında hedef sistemde bir kullanıcı çıkışını çağırabileceğiniz iki noktayı çağırabileceğiniz iki nokta vardır. Aşağıdaki çizelge, çıkış noktalarını kullanmak için gerçekleştirmeniz gereken bu kullanıcı çıkış noktalarının ve Java arabiriminin her birini özetler.

Çizelge 187. Kaynak tarafı ve hedef tarafı çıkış noktalarının ve Java arabirimlerinin özeti

Çıkış noktası	**Gerçekleştirilecek Java arabirimi**
Kaynak tarafı çıkış noktaları:	
Tüm dosya aktarımı başlamadan önce	SourceTransferStartExit.java arabirimi
Tüm dosya aktarımı tamamlandıktan sonra	SourceTransferEndExit.java arabirimi
Hedef tarafı çıkış noktaları:	
Tüm dosya aktarımı başlamadan önce	DestinationTransferStartExit.java arabirimi
Tüm dosya aktarımı tamamlandıktan sonra	DestinationTransferEndExit.java arabirimi

Kullanıcı çıkışları aşağıdaki sırayla çağrılır:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

SourceTransferStartExit ve DestinationTransferStartExit çıkışları tarafından yapılan değişiklikler, sonraki çıkışlara giriş olarak yayılır. Örneğin, SourceTransferStartExit çıkışı aktarma meta verilerini değiştirirse, değişiklikler giriş aktarma meta verilerinde diğer çıkışlara yansıtılır.

Kullanıcı çıkışları ve program çağrıları aşağıdaki sırayla çağrılır:

```
- SourceTransferStartExit(onSourceTransferStart).
- PRE_SOURCE Command.
- DestinationTransferStartExits(onDestinationTransferStart).
- PRE_DESTINATION Command.
- The Transfer request is performed.
- DestinationTransferEndExits(onDestinationTransferENd).
- POST_DESTINATION Command.
- SourceTransferEndExits(onSourceTransferEnd.
- POST_SOURCE Command.
```

Notlar:

1. **`DestinationTransferEndExits`** yalnızca aktarma başarıyla ya da kısmen tamamlandığında çalıştırılır.
2. **`postDestinationCall`** yalnızca aktarma başarıyla ya da kısmen tamamlandığında çalıştırılır.
3. **`SourceTransferEndExits`** başarılı, kısmen başarılı ya da başarısız aktarımlar için çalıştırılır.

4. **postSourceCall** yalnızca aşağıdaki durumda çağrılır:
   - Aktarım iptal edilmedi.
   - Başarılı ya da kısmen başarılı bir sonuç vardır.
   - Hedef sonrası aktarım programları başarıyla çalıştırıldı.

Kullanıcı çıkışınızı oluşturma

Kullanıcı çıkışı oluşturmak için arabirimler *MQ_INSTALL_DIRECTORY*/mqft/lib/com.ibm.wmqfte.exitroutines.api.jariçinde bulunur. Çıkışınızı oluştururken bu .jar dosyasını sınıf yoluna eklemeniz gerekir. Çıkışı çalıştırmak için, çıkışı .jar dosyası olarak açın ve bu .jar dosyasını aşağıdaki bölümde açıklandığı gibi bir dizine yerleştirin.

Kullanıcı çıkış yerleri

Kullanıcı çıkış yordamlarınızı iki olası yerde saklayabilirsiniz:

- exits dizini. Her aracı dizininin altında bir çıkış dizini vardır. Örneğin: var\mqm\mqft\config\QM_JUPITER\agents\AGENT1\exits
- Alternatif bir konum belirtmek için exitClassYol özelliğini ayarlayabilirsiniz. Hem exits dizininde hem de exitClassYoluyla ayarlanan sınıf yolunda çıkış sınıfları varsa, exits dizinindeki sınıflar önceliklidir; bu, her iki konumda da aynı ada sahip sınıflar varsa, exits dizinindeki sınıfların öncelikli olduğu anlamına gelir.

Bir aracıyı kullanıcı çıkışlarını kullanacak şekilde yapılandırma

Bir aracının çağırdığı kullanıcı çıkışlarını belirtmek için ayarlanabilen dört aracı özelliği vardır. Bu aracı özellikleri şunlardır: sourceTransferStartExitClasses, sourceTransferEndExitClasses, destinationTransferStartExitClassesve destinationTransferEndExitClasses. Bu özelliklerin nasıl kullanılacağına ilişkin bilgi için bkz. MFT Kullanıcı çıkışları için aracı özellikleri.

Protokol köprüsü aracılarında kullanıcı çıkışlarının çalıştırılması

Kaynak aracı çıkışı çağırdığında, aktarım için kaynak öğelerin listesini iletir. Normal aracılar için bu, tam olarak nitelenmiş dosya adlarının bir listesidir. Dosyaların yerel olması (ya da bağlama yoluyla erişilebilir olması) gerektiğinden, çıkış dosyaya erişebilir ve dosyayı şifreleyebilir.

Ancak, bir Protokol Köprüsü Aracısı için, listedeki girişler aşağıdaki biçimdedir:

```
"<file server identifier>:<fully-qualified file name of the file on the remote file server>"
```

Listedeki her giriş için, çıkışın önce dosya sunucusuna bağlanması gerekir (FTP kullanılarak). FTPS ya da SFTP iletişim kuralları), dosyayı karşıdan yükleyin, yerel olarak şifreleyin ve şifrelenmiş dosyayı dosya sunucusuna geri yükleyin.

Connect:Direct köprü aracılarında kullanıcı çıkışlarının çalıştırılması

Connect:Direct köprü aracılarında kullanıcı çıkışlarını çalıştıramazsınız.

İlgili kavramlar

“MFT kaynak ve hedef kullanıcı çıkışları” sayfa 1176

MFT kullanıcı çıkışları için meta veriler

MFT kullanıcı çıkışları için Java arabirimleri

İlgili başvurular

“MFT kullanıcı çıkışları için uzaktan hata ayıklamanın etkinleştirilmesi” sayfa 1180

Kullanıcı çıkışlarınızı geliştirirken, kodunuzda sorunların bulunmasına yardımcı olmak için bir hata ayıklayıcı kullanmak isteyebilirsiniz.

“Örnek MFT kaynak aktarma kullanıcı çıkışı” sayfa 1181

“Örnek iletişim kuralı köprüsü kimlik bilgileri kullanıcı çıkışı” sayfa 1182

MFT kaynak izleyicisi kullanıcı çıkışları
MFT Kullanıcı çıkışları için aracı özellikleri

MFT kaynak ve hedef kullanıcı çıkışları

Dizin ayırıcılar

Kaynak dosya belirtimlerindeki dizin ayırıcılar, **fteCreateTransfer** komutunda ya da IBM MQ Explorerkomutunda dizin ayırıcıları nasıl belirttiğinizden bağımsız olarak, her zaman eğik çizgi (/) karakterleri kullanılarak gösterilir. Bir çıkış yazarken bunu dikkate almalısınız. Örneğin, şu kaynak dosyanın var olup olmadığını denetlemek istiyorsanız: `c:\a\b.txt` ve bu kaynak dosyayı **fteCreateTransfer** komutunu ya da IBM MQ Explorerkomutunu kullanarak belirttiyseniz, dosya adının gerçekte şu şekilde saklandığını unutmayın: `c:/a/b.txt` Bu nedenle, özgün `c:\a\b.txt`dizgisini ararken eşleşme bulamazsınız.

Kaynak tarafı çıkış noktaları

Tüm dosya aktarımı başlamadan önce

Bu çıkış, bekleyen aktarımlar listesinde bir sonraki aktarma isteği olduğunda ve aktarma başlamak üzereyken kaynak aracı tarafından çağrılır.

Bu çıkış noktasının kullanımları örnek olarak, aracının bir dış komutu kullanarak okuma/yazma erişimine sahip olduğu bir dizine aşamalı olarak dosya göndermek ya da hedef sistemdeki dosyaları yeniden adlandırmak verilebilir.

Aşağıdaki bağımsız değişkenleri bu çıkışa geçir:

- Kaynak aracı adı
- Hedef aracı adı
- Ortam meta verileri
- Meta verileri aktar
- Dosya belirtimleri (dosya meta verileri dahil)

Bu çıkıştan döndürülen veriler aşağıdaki gibidir:

- Aktarma meta verileri güncellendi. Girdiler eklenebilir, değiştirilebilir ve silinebilir.
- Kaynak dosya adı ve hedef dosya adı çiftlerinden oluşan dosya belirtimlerinin güncellenmiş listesi. Girdiler eklenebilir, değiştirilebilir ve silinebilir
- Aktarmaya devam edilip edilmeyeceğini belirten gösterge
- Aktarma Günlüğüne eklenecek dizgi.

Bu çıkış noktasında kullanıcı çıkış kodunu çağırmak için SourceTransferStartExit.java arabirimini gerçekleştirin.

Tüm dosya aktarımı tamamlandıktan sonra

Bu çıkış, tüm dosya aktarımı tamamlandıktan sonra kaynak aracı tarafından çağrılır.

Bu çıkış noktasının kullanımı, aktarımın tamamlandığını işaretlemek için e-posta ya da IBM MQ iletisi gönderilmesi gibi bazı tamamlanma görevlerini gerçekleştirmektir.

Aşağıdaki bağımsız değişkenleri bu çıkışa geçir:

- Aktarma çıkışı sonucu
- Kaynak aracı adı
- Hedef aracı adı
- Ortam meta verileri
- Meta verileri aktar
- Dosya sonuçları

Bu çıkıştan döndürülen veriler aşağıdaki gibidir:

- Aktarma günlüğüne eklenecek dizgi güncellendi.

Bu çıkış noktasında kullanıcı çıkış kodunu çağırmak için SourceTransferEndExit.java arabirimini gerçekleştirin.

Hedef yan çıkış noktaları

Tüm dosya aktarımı başlamadan önce

Bu çıkış noktasının kullanımına örnek olarak, hedefteki izinlerin doğrulanması verilebilir.

Aşağıdaki bağımsız değişkenleri bu çıkışa geçir:

- Kaynak aracı adı
- Hedef aracı adı
- Ortam meta verileri
- Meta verileri aktar
- Dosya Belirtimleri

Bu çıkıştan döndürülen veriler aşağıdaki gibidir:

- Hedef dosya adları kümesi güncellendi. Girdiler değiştirilebilir, ancak eklenemez ya da silinemez.
- Aktarmaya devam edilip edilmeyeceğini belirten gösterge
- Aktarma Günlüğüne eklenecek dizgi.

Bu çıkış noktasında kullanıcı çıkış kodunu çağırmak için DestinationTransferStartExit.java arabirimini gerçekleştirin.

Tüm dosya aktarımı tamamlandıktan sonra

Bu kullanıcı çıkışının kullanımına örnek olarak, aktarılan dosyaları kullanan bir toplu iş başlatılabilir ya da aktarma başarısız olursa e-posta gönderilebilir.

Aşağıdaki bağımsız değişkenleri bu çıkışa geçir:

- Aktarma çıkışı sonucu
- Kaynak aracı adı
- Hedef aracı adı
- Ortam meta verileri
- Meta verileri aktar
- Dosya sonuçları

Bu çıkıştan döndürülen veriler aşağıdaki gibidir:

- Aktarma günlüğüne eklenecek dizgi güncellendi.

Bu çıkış noktasında kullanıcı çıkış kodunu çağırmak için DestinationTransferEndExit.java arabirimini gerçekleştirin.

İlgili kavramlar

MFT kullanıcı çıkışları için Java arabirimleri

İlgili başvurular

“MFT kullanıcı çıkışları için uzaktan hata ayıklamanın etkinleştirilmesi” sayfa 1180
Kullanıcı çıkışlarınızı geliştirirken, kodunuzda sorunların bulunmasına yardımcı olmak için bir hata ayıklayıcı kullanmak isteyebilirsiniz.

“Örnek MFT kaynak aktarma kullanıcı çıkışı” sayfa 1181

MFT kaynak izleyicisi kullanıcı çıkışları

MFT aktarım G/Ç kullanıcı çıkışlarının kullanılması

Managed File Transfer aktarımlarına ilişkin temel dosya sistemi G/Ç işini gerçekleştirmek üzere özel kod yapılandırmak için Managed File Transfer aktarım G/Ç kullanıcı çıkışlarını kullanabilirsiniz.

Genellikle MFT aktarımları için bir aracı, aktarım için uygun dosya sistemleriyle etkileşimde bulunmak üzere yerleşik G/Ç sağlayıcılarından birini seçer. Yerleşik G/Ç sağlayıcıları aşağıdaki dosya sistemi tiplerini destekler:

- Normal UNIX-type ve Windows-type dosya sistemleri
- z/OS z/OS sıralı ve bölümlenmiş veri kümeleri (yalnızca z/OS üzerinde)
- IBM i IBM i yerel saklama dosyaları (yalnızca IBM i üzerinde)
- IBM MQ Kuyruklar
- Uzak FTP ve SFTP protokol sunucuları (yalnızca protokol köprüsü aracıları için)
- Uzak Connect:Direct düğümleri (yalnızca Connect:Direct köprü aracıları için)

Desteklenmeyen ya da özel G/Ç davranışı gerektiren dosya sistemleri için bir aktarım G/Ç kullanıcı çıkışı yazabilirsiniz.

Aktarım G/Ç kullanıcı çıkışları, kullanıcı çıkışları için var olan altyapıyı kullanır. Ancak, bu aktarma G/Ç kullanıcı çıkışları, işlevlerine her dosya için aktarma sırasında birden çok kez erişildiği için diğer kullanıcı çıkışlarından farklıdır.

Yüklenecek G/Ç çıkış sınıflarını belirtmek için IOExitClasses (`agent.properties` dosyasında) aracı özelliğini kullanın. Her çıkış sınıfını virgülle ayırın, örneğin:

```
IOExitClasses=testExits.TestExit1,testExits.testExit2
```

Aktarım G/Ç kullanıcı çıkışlarına ilişkin Java arabirimleri aşağıdaki gibidir:

IOExit
G/Ç çıkışının kullanılıp kullanılmadığını belirlemek için kullanılan ana giriş noktası. Bu eşgörünüm, IOExitPath yönetim ortamlarının oluşmasından sorumludur.

IOExitClassesaracı özelliği için yalnızca IOExit G/Ç çıkış arabirimini belirtmeniz gerekir.

IOExitPath
Soyut bir arabirimi gösterir; örneğin, bir veri kapsayıcısı kümesini gösteren bir veri kapsayıcısı ya da genel arama karakteri. Bu arabirimi gerçekleştiren bir sınıf eşgörünümü yaratamazsınız. Arabirim, yolun incelenmesini ve türetilen yolların listelenmesini sağlar. IOExitResourceYolu ve IOExitWildcardYol arabirimleri IOExitPath' ı genişletir.

IOExitChannel
Verilerin bir IOExitPath kaynağından okunmasını ya da bir kaynağa yazılmasını sağlar.

IOExitRecordKanalı
Verilerin birden çok kayıt içinde IOExitPath kaynağından okunmasını ya da bu kaynağa yazılmasını sağlayan, kayıt odaklı IOExitPath kaynakları için IOExitChannel arabirimini genişletir.

IOExitLock
Paylaşılan ya da dışlayıcı erişim için IOExitPath kaynağındaki bir kilidi gösterir.

z/OS **IOExitRecordResourcePath**
IOExitResourceYol arabirimini, kayıt odaklı bir dosya için veri kapsayıcısını (örneğin, bir z/OS veri kümesi) gösterecek şekilde genişletir. Verileri bulmak ve okuma ya da yazma işlemleri için IOExitRecordKanal örnekleri oluşturmak için arabirimi kullanabilirsiniz.

IOExitResourceYolu
IOExitPath arabirimini, bir veri kapsayıcısını (örneğin, bir dosya ya da dizin) gösterecek şekilde genişletir. Verileri bulmak için arabirimi kullanabilirsiniz. Arabirim bir dizini gösteriyorsa, yolların listesini döndürmek için listPaths yöntemini kullanabilirsiniz.

IOExitWildcardYolu
IOExitPath arabirimini, genel arama karakterini gösteren bir yolu gösterecek şekilde genişletir. Bu arabirimi birden çok IOExitResourceYoluyla eşleştirmek için kullanabilirsiniz.

IOExitProperties
Managed File Transfer 'in G/Ç' nin belirli yönleri için IOExitPath ' ı nasıl işleyeceğini belirleyen özellikleri belirtir. Örneğin, bir aktarma yeniden başlatılırsa, ara dosyaların kullanılıp kullanılmayacağı ya da bir kaynağın baştan yeniden okuyup okumayacağı.

İlgili kavramlar
"MFT ürününü kullanıcı çıkışlarıyla özelleştirme" sayfa 1174
Kullanıcı çıkışı yordamları olarak bilinen kendi programlarınızı kullanarak Managed File Transfer özelliklerini özelleştirebilirsiniz.

İlgili başvurular
IOExit.java arabirimi
IOExitChannel.java arabirimi
IOExitLock.java arabirimi
IOExitPath.java arabirimi
IOExitProperties.java arabirimi
IOExitRecordChannel.java arabirimi
z/OS IOExitRecordResourcePath.java arabirimi
IOExitResourcePath.java arabirimi
IOExitWildcardPath.java arabirimi
MFT`agent.properties` dosyası

IBM i IBM i kullanıcı çıkışlarında örnek MFT

Managed File Transfer , kuruluşunuzla IBM i ürününe özgü örnek kullanıcı çıkışları sağlar. Örnekler, *MQMFT_install_dir*`/samples/ioexit-IBMi` ve *MQMFT_install_dir*`/samples/userexit-IBMi`dizinlerinde bulunur.

com.ibm.wmqfte.exit.io.ibmi.qdls.FTEQDLSExit
com.ibm.wmqfte.exit.io.ibmi.qdls.FTEQDLSExit örnek kullanıcı çıkışı, IBM iüzerinde QDLS dosya sistemindeki dosyaları aktarır. Çıkış kurulduktan sonra, `/QDLS` ile başlayan dosyalara yapılan aktarımlar çıkışı otomatik olarak kullanır.

Bu çıkışı kurmak için aşağıdaki adımları tamamlayın:

1. `com.ibm.wmqfte.samples.ibmi.ioexits.jar` dosyasını *WMQFTE_install_dir*`/samples/ioexit-IBMi` dizininden aracının `exits` dizinine kopyalayın.
2. com.ibm.wmqfte.exit.io.ibmi.qdls.FTEQDLSExit öğesini IOExitClasses özelliğine ekleyin.
3. Aracıyı yeniden başlatın.

com.ibm.wmqfte.exit.user.ibmi.FileMemberMonitorExit
com.ibm.wmqfte.exit.user.ibmi.FileMemberMonitorExit örnek kullanıcı çıkışı, MFT dosya izleyicisi gibi davranır ve fiziksel dosya üyelerini bir IBM i kitaplığından otomatik olarak aktarır.

Bu çıkışı çalıştırmak için, "library.qsys.monitor" meta veri alanı için bir değer belirtin (örneğin, **-md** parametresini kullanarak). Bu parametre, bir dosya üyesinin IFS stili yolunu alır ve dosya ve üye genel arama karakterleri içerebilir. Örneğin, /QSYS.LIB/FOO.LIB/BAR.FILE/*.MBR, /QSYS.LIB/FOO.LIB/*.FILE/BAR.MBR, /QSYS.LIB/FOO.LIB/*.FILE/*.-MBR.

Bu örnek çıkışta, aktarma sırasında kullanılan adlandırma şemasını belirlemek için kullanabileceğiniz isteğe bağlı bir meta veri alanı "naming.scheme.qsys.monitor" de bulunur. Varsayılan olarak bu alan "unix" olarak ayarlanır ve hedef dosyanın F00.MBRolarak adlandırılmasına neden olur. IBM i FTP FILE.MEMBER şeması, örneğin, /QSYS.LIB/FOO.LIB/BAR.FILE/BAZ.MBR , BAR.BAZ.

Bu çıkışı kurmak için aşağıdaki adımları tamamlayın:

1. `com.ibm.wmqfte.samples.ibmi.userexits.jar` dosyasını *`WMQFTE_install_dir`*`/samples/userexit-IBMi` dizininden aracının `exits` dizinine kopyalayın.
2. `agent.properties` dosyasındaki sourceTransferStartExitClasses özelliğine com.ibm.wmqfte.exit.user.ibmi.FileMemberMonitorExit öğesini ekleyin.
3. Aracıyı yeniden başlatın.

com.ibm.wmqfte.exit.user.ibmi.EmptyFileDeleteExit

com.ibm.wmqfte.exit.user.ibmi.EmptyFileDeleteExit örnek kullanıcı çıkışı, kaynak dosya üyesi aktarımın bir parçası olarak silindiğinde boş bir dosya nesnesini siler. IBM i dosya nesneleri birçok üyeyi tutabileceğinden, dosya nesneleri MFTtarafından dizinler gibi işlenir. Bu nedenle, MFTkomutunu kullanarak bir dosya nesnesi üzerinde taşıma işlemi gerçekleştiremezsiniz; taşıma işlemleri yalnızca üye düzeyinde desteklenir. Sonuç olarak, bir üye üzerinde taşıma işlemi gerçekleştirdiğinizde, şimdi boş olan dosya geride kalır. Bu boş dosyaları aktarma isteğinin bir parçası olarak silmek istiyorsanız bu örnek çıkışı kullanın.

"empty.file.delete" meta verileri için "true" değerini belirtirseniz ve bir FTEFileMemberaktarirseniz, dosya boşsa örnek çıkış üst dosyayı siler.

Bu çıkışı kurmak için aşağıdaki adımları tamamlayın:

1. com.ibm.wmqfte.samples.ibmi.userexits.jar dosyasını *`WMQFTE_install_dir`*`/samples/userexit-IBMi` dizininden aracının `exits` dizinine kopyalayın.
2. `agent.properties` dosyasındaki sourceTransferStartExitClasses özelliğine com.ibm.wmqfte.exit.user.ibmi.EmptyFileDeleteExit ekleyin.
3. Aracıyı yeniden başlatın.

İlgili başvurular

“MFT aktarım G/Ç kullanıcı çıkışlarının kullanılması” sayfa 1178
Managed File Transfer aktarımlarına ilişkin temel dosya sistemi G/Ç işini gerçekleştirmek üzere özel kod yapılandırmak için Managed File Transfer aktarım G/Ç kullanıcı çıkışlarını kullanabilirsiniz.

MFT Kullanıcı çıkışları için aracı özellikleri

MFT kullanıcı çıkışları için uzaktan hata ayıklamanın etkinleştirilmesi

Kullanıcı çıkışlarınızı geliştirirken, kodunuzda sorunların bulunmasına yardımcı olmak için bir hata ayıklayıcı kullanmak isteyebilirsiniz.

Aracıyı çalıştıran Java sanal makinesinde çıkışlar çalıştığından, genellikle tümleşik bir geliştirme ortamında bulunan doğrudan hata ayıklama desteğini kullanamazsınız. Ancak, JVM ' nin uzaktan hata ayıklamasını etkinleştirebilir ve daha sonra, uygun bir uzak hata ayıklayıcıyı bağlayabilirsiniz.

Uzaktan hata ayıklamayı etkinleştirmek için standart JVM parametrelerini **`-Xdebug`** ve **`-Xrunjdwp`**kullanın. Bu özellikler, aracıyı çalıştıran JVM ' ye **BFG_JVM_PROPERTIES** ortam değişkeni tarafından iletilir. Örneğin, AIX and Linux üzerinde aşağıdaki komutlar aracıyı başlatır ve JVM ' nin 8765 TCP kapısındaki hata ayıklayıcı bağlantılarını dinlemesine neden olur.

```
export BFG_JVM_PROPERTIES="-Xdebug -Xrunjdwp:transport=dt_socket,server=y,address=8765"
fteStartAgent -F TEST_AGENT
```

Hata ayıklayıcı bağlanıncaya kadar aracı başlatılamaz. **`export`** komutu yerine Windows üzerinde **`set`** komutunu kullanın.

Hata ayıklayıcı ile JVM arasında başka iletişim yöntemleri de kullanabilirsiniz. Örneğin, JVM hata ayıklayıcıya bağlantıyı açabilir ya da TCP yerine paylaşılan bellek kullanabilirsiniz. Daha fazla ayrıntı için Java Platform Debugger Architecture belgesine bakın.

Aracıyı uzak hata ayıklama kipinde başlattığınızda **-F** (önalan) parametresini kullanmanız gerekir.

Eclipse hata ayıklayıcısını kullanma

Aşağıdaki adımlar, Eclipse geliştirme ortamındaki uzaktan hata ayıklama yeteneği için geçerlidir. JPDA uyumlu diğer uzak hata ayıklayıcılar da kullanabilirsiniz.

1. Eclipsesürümünüze bağlı olarak **Çalıştır** > **Hata Ayıklama İletişim Kutusunu Aç** (ya da **Çalıştır** > **Hata Ayıklama Yapılandırmaları** ya da **Çalıştır** > **Hata Ayıklama İletişim Kutusu**) seçeneğini tıklatın.
2. Hata ayıklama yapılanışı yaratmak için yapılanış tipleri listesinde **Uzak Java Uygulaması** öğesini çift tıklatın.
3. Yapılandırma alanlarını doldurun ve hata ayıklama yapılandırmasını kaydedin. Aracı JVM 'yi hata ayıklama kipinde önceden başlatdıysanız, JVM' ye şimdi bağlanabilirsiniz.

Örnek MFT kaynak aktarma kullanıcı çıkışı

```
/*
 * A Sample Source Transfer End Exit that prints information about a transfer to standard
output.
 * If the agent is run in the background the output will be sent to the agent's event log file.
If
 * the agent is started in the foreground by specifying the -F parameter on the fteStartAgent
 * command the output will be sent to the console.
 *
 * To run the exit execute the following steps:
 *
 * Compile and build the exit into a jar file. You need the following in the class path:
 * {MQ_INSTALLATION_PATH}\mqft\lib\com.ibm.wmqfte.exitroutines.api.jar
 *
 * Put the jar in your agent's exits directory:
 * {MQ_DATA_PATH}\config\coordQmgrName\agents\agentName\exits\
 *
 * Update the agent's properties file:
 * {MQ_DATA_PATH}\config\coordQmgrName\agents\agentName\agent.properties
 * to include the following property:
 * sourceTransferEndExitClasses=[packageName.]SampleEndExit
 *
 * Restart agent to pick up the exit
 *
 * Send the agent a transfer request:
 * For example: fteCreateTransfer -sa myAgent -da YourAgent -df output.txt input.txt
 */

import java.util.List;
import java.util.Map;
import java.util.Iterator;

import com.ibm.wmqfte.exitroutine.api.SourceTransferEndExit;
import com.ibm.wmqfte.exitroutine.api.TransferExitResult;
import com.ibm.wmqfte.exitroutine.api.FileTransferResult;

public class SampleEndExit implements SourceTransferEndExit {

    public String onSourceTransferEnd(TransferExitResult transferExitResult,
            String sourceAgentName,
            String destinationAgentName,
            Map<String, String>environmentMetaData,
            Map<String, String>transferMetaData,
            List<FileTransferResult>fileResults) {

        System.out.println("Environment Meta Data: " + environmentMetaData);
        System.out.println("Transfer Meta Data:    " + transferMetaData);

        System.out.println("Source agent:          " +
                sourceAgentName);
        System.out.println("Destination agent:     " +
                destinationAgentName);
```

```
        if (fileResults.isEmpty()) {
            System.out.println("No files in the list");
            return "No files";
        }
        else {

            System.out.println( "File list: ");

            final Iterator<FileTransferResult> iterator = fileResults.iterator();

            while (iterator.hasNext()){
                final FileTransferResult thisFileSpec = iterator.next();
                System.out.println("Source file spec: "  +
                        thisFileSpec.getSourceFileSpecification() +
                        ", Destination file spec: " +
                        thisFileSpec.getDestinationFileSpecification());
            }
        }
        return "Done";
    }
}
```

Örnek iletişim kuralı köprüsü kimlik bilgileri kullanıcı çıkışı

Bu örnek kullanıcı çıkışını kullanma hakkında bilgi için Çıkış sınıflarını kullanarak dosya sunucusuna ilişkin kimlik bilgilerini eşlemebaşlıklı konuya bakın.

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import java.util.StringTokenizer;

import com.ibm.wmqfte.exitroutine.api.CredentialExitResult;
import com.ibm.wmqfte.exitroutine.api.CredentialExitResultCode;
import com.ibm.wmqfte.exitroutine.api.CredentialPassword;
import com.ibm.wmqfte.exitroutine.api.CredentialUserId;
import com.ibm.wmqfte.exitroutine.api.Credentials;
import com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit;

/**
 * A sample protocol bridge credential exit
 *
 * This exit reads a properties file that maps mq user ids to server user ids
 * and server passwords. The format of each entry in the properties file is:
 *
 * mqUserId=serverUserId,serverPassword
 *
 * The location of the properties file is taken from the protocol bridge agent
 * property protocolBridgeCredentialConfiguration.
 *
 * To install the sample exit compile the class and export to a jar file.
 * Place the jar file in the exits subdirectory of the agent data directory
 * of the protocol bridge agent on which the exit is to be installed.
 * In the agent.properties file of the protocol bridge agent set the
 * protocolBridgeCredentialExitClasses to SampleCredentialExit
 * Create a properties file that contains the mqUserId to serverUserId and
 * serverPassword mappings applicable to the agent. In the agent.properties
 * file of the protocol bridge agent set the protocolBridgeCredentialConfiguration
 * property to the absolute path name of this properties file.
 * To activate the changes stop and restart the protocol bridge agent.
 *
 * For further information on protocol bridge credential exits refer to
 * the WebSphere MQ Managed File Transfer documentation online at:
 * https://www.ibm.com/docs/SSEP7X_7.0.4/welcome/WelcomePagev7r0.html
 */
public class SampleCredentialExit implements ProtocolBridgeCredentialExit {

    // The map that holds mq user ID to serverUserId and serverPassword mappings
    final private Map<String,Credentials> credentialsMap = new HashMap<String, Credentials>();
```

```
    /* (non-Javadoc)
      * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#initialize(java.util.Map)
      */
     public synchronized boolean initialize(Map<String, String> bridgeProperties) {

        // Flag to indicate whether the exit has been successfully initialized or not
        boolean initialisationResult = true;

        // Get the path of the mq user ID mapping properties file
         final String propertiesFilePath = bridgeProperties.get("protocolBridgeCredentialConfiguration");

         if (propertiesFilePath == null || propertiesFilePath.length() == 0) {
             // The properties file path has not been specified. Output an error and return false
             System.err.println("Error initializing SampleCredentialExit.");
              System.err.println("The location of the mqUserID mapping properties file has not been
specified in the
       protocolBridgeCredentialConfiguration property");
             initialisationResult = false;
         }

        if (initialisationResult) {

            // The Properties object that holds mq user ID to serverUserId and serverPassword
             // mappings from the properties file
             final Properties mappingProperties = new Properties();

             // Open and load the properties from the properties file
             final File propertiesFile = new File (propertiesFilePath);
             FileInputStream inputStream = null;
             try {
                 // Create a file input stream to the file
                 inputStream = new FileInputStream(propertiesFile);

                 // Load the properties from the file
                 mappingProperties.load(inputStream);
             }
             catch (FileNotFoundException ex) {
                 System.err.println("Error initializing SampleCredentialExit.");
                  System.err.println("Unable to find the mqUserId mapping properties file: " +
propertiesFilePath);
                 initialisationResult = false;
             }
             catch (IOException ex) {
                 System.err.println("Error initializing SampleCredentialExit.");
                  System.err.println("Error loading the properties from the mqUserId mapping properties
file: " + propertiesFilePath);
                 initialisationResult = false;
             }
             finally {
                 // Close the inputStream
                 if (inputStream != null) {
                     try {
                         inputStream.close();
                     }
                     catch (IOException ex) {
                         System.err.println("Error initializing SampleCredentialExit.");
                          System.err.println("Error closing the mqUserId mapping properties file: " +
propertiesFilePath);
                         initialisationResult = false;
                     }
                 }
             }

             if (initialisationResult) {
                 // Populate the map of mqUserId to server credentials from the properties
                 final Enumeration<?> propertyNames = mappingProperties.propertyNames();
                  while ( propertyNames.hasMoreElements()) {
                     final Object name = propertyNames.nextElement();
                     if (name instanceof String ) {
                         final String mqUserId = ((String)name).trim();
                          // Get the value and split into serverUserId and serverPassword
                        final String value = mappingProperties.getProperty(mqUserId);
                        final StringTokenizer valueTokenizer = new StringTokenizer(value, ",");
                         String serverUserId = "";
                         String serverPassword = "";
                         if (valueTokenizer.hasMoreTokens()) {
                             serverUserId = valueTokenizer.nextToken().trim();
                         }
                         if (valueTokenizer.hasMoreTokens()) {
                             serverPassword = valueTokenizer.nextToken().trim();
                         }
```

```
                        // Create a Credential object from the serverUserId and serverPassword
final Credentials credentials = new Credentials(new CredentialUserId(serverUserId), new
CredentialPassword(serverPassword));
                        // Insert the credentials into the map
                        credentialsMap.put(mqUserId, credentials);
                    }
                }

            }
        }

        return initialisationResult;
    }
     /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#mapMQUserId(java.lang.String)
     */
    public synchronized CredentialExitResult mapMQUserId(String mqUserId) {
        CredentialExitResult result = null;
        // Attempt to get the server credentials for the given mq user id
        final Credentials credentials = credentialsMap.get(mqUserId.trim());
        if ( credentials == null) {
            // No entry has been found so return no mapping found with no credentials
            result = new CredentialExitResult(CredentialExitResultCode.NO_MAPPING_FOUND, null);
        }
        else {
            // Some credentials have been found so return success to the user along with the credentials
            result = new CredentialExitResult(CredentialExitResultCode.USER_SUCCESSFULLY_MAPPED,
credentials);
        }
        return result;
    }
     /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#shutdown(java.util.Map)
     */
    public void shutdown(Map<String, String> bridgeProperties) {
        // Nothing to do in this method because there are no resources that need to be released
    }
}
```

Örnek protokol köprüsü özellikleri kullanıcı çıkışı

Bu örnek kullanıcı çıkışının nasıl kullanılacağına ilişkin bilgi için bkz. ProtocolBridgePropertiesExit2: Protokol dosya sunucusu özelliklerinin araştırılması

SamplePropertiesExit2.java

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Properties;

import com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2;
import com.ibm.wmqfte.exitroutine.api.ProtocolServerPropertyConstants;

/**
 * A sample protocol bridge properties exit. This exit reads a properties file
 * that contains properties for protocol servers.
 * <p>
 * The format of each entry in the properties file is:
 * {@literal serverName=type://host:port}
 * Ensure there is a default entry such as
 * {@literal default=type://host:port}
 * otherwise the agent will fail to start with a BFGBR0168 as it must have a
 * default server.
 * <p>
 * The location of the properties file is taken from the protocol bridge agent
 * property {@code protocolBridgePropertiesConfiguration}.
 * <p>
 * The methods {@code getCredentialLocation} returns the location of the associated
 * ProtocolBridgeCredentials.xml, this sample it is defined to be stored in a directory
 * defined by the environment variable CREDENTIALSHOME
 * <p>
```

```
 * To install the sample exit:
 * <ol>
 * <li>Compile the class and export to a jar file.
 * <li>Place the jar file in the {@code exits} subdirectory of the agent data directory
 * of the protocol bridge agent on which the exit is to be installed.
 * <li>In the {@code agent.properties} file of the protocol bridge agent
 * set the {@code protocolBridgePropertiesExitClasses} to
 * {@code SamplePropertiesExit2}.
 * <li>Create a properties file that contains the appropriate properties to specify the
 * required servers.
 * <li>In the {@code agent.properties} file of the protocol bridge agent
 * set the <code>protocolBridgePropertiesConfiguration</code> property to the
 * absolute path name of this properties file.
 * <li>To activate the changes stop and restart the protocol bridge agent.
 * </ol>
 * <p>
 * For further information on protocol bridge properties exits refer to the
 * WebSphere MQ Managed File Transfer documentation online at:
 * <p>
 * {@link https://www.ibm.com/docs/SSEP7X_7.0.4/welcome/WelcomePagev7r0.html}
 */
public class SamplePropertiesExit2 implements ProtocolBridgePropertiesExit2 {

    /**
     * Helper class to encapsulate protocol server information.
     */
    private static class ServerInformation {
        private final String type;
        private final String host;
        private final int port;

        public ServerInformation(String url) {
            int index = url.indexOf("://");
            if (index == -1) throw new IllegalArgumentException("Invalid server URL: "+url);
            type = url.substring(0, index);

            int portIndex = url.indexOf(":", index+3);
            if (portIndex == -1) {
                host = url.substring(index+3);
                port = -1;
            } else {
                host = url.substring(index+3,portIndex);
                port = Integer.parseInt(url.substring(portIndex+1));
            }
        }

        public String getType() {
            return type;
        }

        public String getHost() {
            return host;
        }

        public int getPort() {
            return port;
        }
    }

    /** A {@code Map} that holds information for each configured protocol server */
    final private Map<String, ServerInformation> servers = new HashMap<String, ServerInformation>();


    /* (non-Javadoc)
     * @see
com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#getProtocolServerProperties(java.lang.String)
     */
    public Properties getProtocolServerProperties(String protocolServerName) {
        // Attempt to get the protocol server information for the given protocol server name
        // If no name has been supplied then this implies the default.
        final ServerInformation info;
        if (protocolServerName == null || protocolServerName.length() == 0) {
            protocolServerName = "default";
        }
        info = servers.get(protocolServerName);

        // Build the return set of properties from the collected protocol server information, when
available.
        // The properties set here is the minimal set of properties to be a valid set.
        final Properties result;
        if (info != null) {
            result = new Properties();
```

```
            result.setProperty(ProtocolServerPropertyConstants.SERVER_NAME, protocolServerName);
            result.setProperty(ProtocolServerPropertyConstants.SERVER_TYPE, info.getType());
            result.setProperty(ProtocolServerPropertyConstants.SERVER_HOST_NAME, info.getHost());
            if (info.getPort() != -1)
result.setProperty(ProtocolServerPropertyConstants.SERVER_PORT_VALUE, ""+info.getPort());
            result.setProperty(ProtocolServerPropertyConstants.SERVER_PLATFORM, "UNIX");
            if (info.getType().toUpperCase().startsWith("FTP")) {        // FTP & FTPS
                result.setProperty(ProtocolServerPropertyConstants.SERVER_TIMEZONE, "Europe/London");
                result.setProperty(ProtocolServerPropertyConstants.SERVER_LOCALE, "en-GB");
            }
            result.setProperty(ProtocolServerPropertyConstants.SERVER_FILE_ENCODING, "UTF-8");
        } else {
            System.err.println("Error no default protocol file server entry has been supplied");
            result = null;
        }

        return result;
    }

    /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#initialize(java.util.Map)
     */
    public boolean initialize(Map<String, String> bridgeProperties) {
        // Flag to indicate whether the exit has been successfully initialized or not
        boolean initialisationResult = true;

        // Get the path of the properties file
        final String propertiesFilePath = bridgeProperties.get("protocolBridgePropertiesConfiguration");
        if (propertiesFilePath == null || propertiesFilePath.length() == 0) {
            // The protocol server properties file path has not been specified. Output an error and
return false
            System.err.println("Error initializing SamplePropertiesExit.");
            System.err.println("The location of the protocol server properties file has not been
specified in the
      protocolBridgePropertiesConfiguration property");
            initialisationResult = false;
        }

        if (initialisationResult) {
            // The Properties object that holds protocol server information
            final Properties mappingProperties = new Properties();

            // Open and load the properties from the properties file
            final File propertiesFile = new File (propertiesFilePath);
            FileInputStream inputStream = null;
            try {
                // Create a file input stream to the file
                inputStream = new FileInputStream(propertiesFile);

                // Load the properties from the file
                mappingProperties.load(inputStream);
            } catch (final FileNotFoundException ex) {
                System.err.println("Error initializing SamplePropertiesExit.");
                System.err.println("Unable to find the protocol server properties file: " +
propertiesFilePath);
                initialisationResult = false;
            } catch (final IOException ex) {
                System.err.println("Error initializing SamplePropertiesExit.");
                System.err.println("Error loading the properties from the protocol server properties
file: " + propertiesFilePath);
                initialisationResult = false;
            } finally {
                // Close the inputStream
                if (inputStream != null) {
                    try {
                        inputStream.close();
                    } catch (final IOException ex) {
                        System.err.println("Error initializing SamplePropertiesExit.");
                        System.err.println("Error closing the protocol server properties file: " +
propertiesFilePath);
                        initialisationResult = false;
                    }
                }
            }

            if (initialisationResult) {
                // Populate the map of protocol servers from the properties
                for (Entry<Object, Object> entry : mappingProperties.entrySet()) {
                    final String serverName = (String)entry.getKey();
                    final ServerInformation info = new ServerInformation((String)entry.getValue());
                    servers.put(serverName, info);
                }
```

```
            }
        }

        return initialisationResult;
    }

    /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#shutdown(java.util.Map)
     */
    public void shutdown(Map<String, String> bridgeProperties) {
        // Nothing to do in this method because there are no resources that need to be released
    }

    /* (non-Javadoc)
     * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2#getCredentialLocation()
     */
    public String getCredentialLocation() {
        String envLocationPath;
        if (System.getProperty("os.name").toLowerCase().contains("win")) {
            // Windows style
            envLocationPath = "%CREDENTIALSHOME%\\ProtocolBridgeCredentials.xml";
        }
        else {
            // Unix style
            envLocationPath = "$CREDENTIALSHOME/ProtocolBridgeCredentials.xml";
        }
        return envLocationPath;
    }

}
```

Aracı komut kuyruğuna ileti koyarak MFT ' u denetleme

Aracı komut kuyruklarına ileti koyarak Managed File Transfer ' u denetleyen bir uygulama yazabilirsiniz.

Aracının aşağıdaki işlemlerden birini gerçekleştirmesini istemek için bir aracının komut kuyruğuna bir ileti koyabilirsiniz:

- Dosya aktarımı oluştur
- Zamanlanmış dosya aktarımı yaratır
- Dosya aktarımını iptal et
- Zamanlanmış dosya aktarımını iptal et
- Komut çağırma
- İzleme Programı Yaratılması
- İzleme Programının Silinmesi
- Aracının etkin olduğunu belirtmek için bir ping döndürme

Aracının bu işlemlerden birini gerçekleştirmesini istemek için, iletinin aşağıdaki şemadan biriyle uyumlu bir XML biçiminde olması gerekir:

FileTransfer.xsd
Bu biçimdeki iletiler, dosya aktarımı ya da zamanlanmış dosya aktarımı yaratmak, bir komutu çağırmak ya da dosya aktarımını ya da zamanlanmış dosya aktarımını iptal etmek için kullanılabilir. Daha fazla bilgi için bkz. Dosya aktarma isteği iletisi biçimi.

Monitor.xsd
Bu biçimdeki iletiler, kaynak izleme programı yaratmak ya da silmek için kullanılabilir. Daha fazla bilgi için bkz. MFT izleme programı istek iletisi biçimleri.

PingAgent.xsd
Bu biçimdeki iletiler, etkin olup olmadığını denetlemek üzere bir aracıya ping komutu göndermek için kullanılabilir. Daha fazla bilgi için bkz. Ping MFT aracısı istek iletisi biçimi.

Aracı, istek iletilerine bir yanıt döndürür. Yanıt iletisi, istek iletisinde tanımlanan bir yanıt kuyruğuna yerleştirilir. Yanıt iletisi, aşağıdaki şema tarafından tanımlanan bir XML biçiminde:

Reply.xsd
Daha fazla bilgi için bkz. MFT aracısı yanıt iletisi biçimi.

MQ Telemetry için uygulama geliştirilmesi

Telemetri uygulamaları, algılama ve kontrol aygıtlarını internet ve kuruluşlarda bulunan diğer bilgi kaynaklarıyla bütünleştirir.

Tasarım kalıplarını, örnek örnekleri, örnek programları, programlama kavramlarını ve başvuru bilgilerini kullanarak MQ Telemetry için uygulamalar geliştirin.

İlgili kavramlar

MQ Telemetry

Telemetri kullanım senaryoları

İlgili görevler

kurmaMQ Telemetry

YönetmeMQ Telemetry

MQ Telemetry sorunlarının giderilmesi

İlgili başvurular

MQ Telemetry Başvuru

IBM MQ Telemetry Transport Örnek programlar

Örnek komut dosyaları, örnek bir IBM MQ Telemetry Transport v3 istemci uygulamasıyla (`mqttv3app.jar`) çalışır. IBM MQ 8.0.0 ve sonraki sürümler için örnek istemci uygulaması artık MQ Telemetryiçinde bulunmaz. Bu, IBM Messaging Telemetry Clients SupportPacürününün bir parçasıydı (artık kullanılamıyor). Benzer örnek uygulamalar Eclipse Paho ve MQTT.orgtarafından ücretsiz olarak kullanıma sunulmaya devam eder.

En son bilgiler ve yüklemeler için aşağıdaki kaynaklara bakın:

- Eclipse Paho projesi ve MQTT.org, çeşitli programlama dilleri için en son telemetri istemcilerini ve örneklerini ücretsiz olarak karşıdan yüklüyor. IBM MQ Telemetry Transport' yi yayınlamak ve abone olmak ve güvenlik özellikleri eklemek için örnek programlar geliştirmenize yardımcı olması amacıyla bu siteleri kullanın.
- IBM Messaging Telemetry Clients SupportPac artık karşıdan yüklenebilir durumda değil. Önceden karşıdan yüklenen bir kopyanızı varsa, bu kopyanın içeriği şöyledir:
  - IBM Messaging Telemetry Clients SupportPac MA9B sürümü, derlenmiş bir örnek uygulama (`mqttv3app.jar`) ve ilişkili istemci kitaplığı (`mqttv3.jar`) içeriyordu. Bunlar aşağıdaki dizinlerde sağlanmıştır:
    - `ma9b/SDK/clients/java/org.eclipse.paho.sample.mqttv3app.jar`
    - `ma9b/SDK/clients/java/org.eclipse.paho.client.mqttv3.jar`
  - Bu SupportPac' ın MA9C sürümünde `/SDK/` dizini ve içeriği kaldırılmıştır:
    - Yalnızca örnek uygulama (`mqttv3app.jar`) için kaynak sağlandı. Şu ldı ki var var:

      ```
      ma9c/clients/java/samples/org/eclipse/paho/sample/mqttv3app/*.java
      ```

    - Derlenmiş istemci kitaplığı sağlandı. Şu ldı ki var var:

      ```
      ma9c/clients/java/org.eclipse.paho.client.mqttv3-1.0.2.jar
      ```

IBM Messaging Telemetry Clients SupportPacürününün bir kopyası hala varsa (artık kullanılamıyorsa), örnek uygulamanın kurulmasına ve çalıştırılmasına ilişkin bilgiler Komut satırı kullanılarak MQ Telemetry kuruluşunun doğrulanmasıbaşlıklı konuda sağlanır.

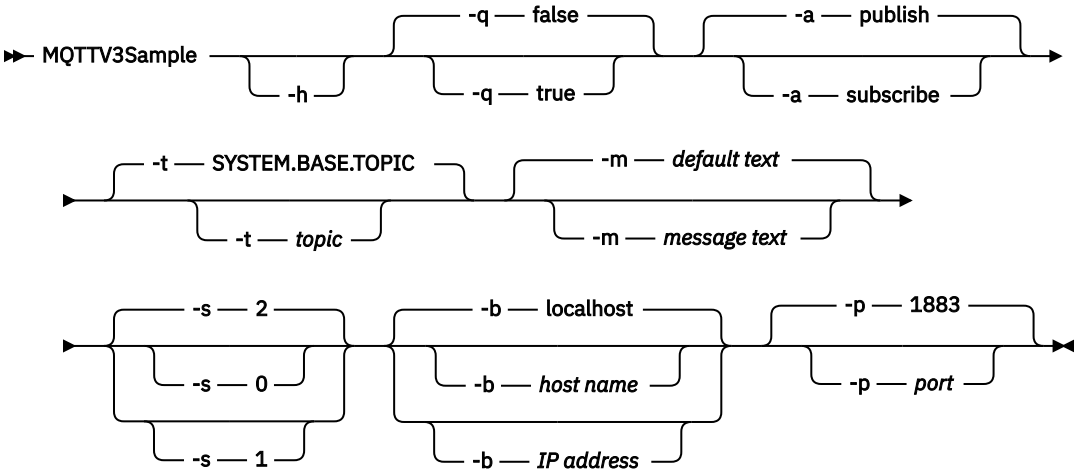
MQTTV3Sample programı

`MQTTV3Sample` programına ilişkin örnek sözdizimi ve parametrelerle ilgili başvuru bilgileri.

Amaç

`MQTTV3Sample` programı, bir iletiyi yayınlamak ve bir konuya abone olmak için kullanılabilir. Bu örnek programın nasıl edineceğiyle ilgili bilgi için bkz. “IBM MQ Telemetry Transport Örnek programlar” sayfa 1188.

MQTTV3Sample syntax



Parametreler

-h
Bu yardım metnini yazdır ve çık

-q
False varsayılan kipini kullanmak yerine sessiz kipi ayarlayın.

-a
Varsayılan yayınlama işlemini varsaymak yerine yayınlama ya da abone olma ayarını belirleyin.

-t
Varsayılan konuyu yayınlamak ya da abone olmak yerine konuyu yayınlayın ya da konuya abone olun

-m
Varsayılan yayın metnini ("Merhaba, bir MQTT v3 uygulamasından") göndermek yerine ileti metnini yayınlayın.

-s
Varsayılan QoS, 2 değerini kullanmak yerine QoS değerini belirleyin.

-b
Varsayılan anasistem adına (localhost) bağlanmak yerine bu anasistem adına ya da IP adresine bağlanın.

-p
Varsayılan değeri (1883) kullanmak yerine bu kapıyı kullanın.

MQTTV3Sample programının çalıştırılması

Windowsile ilgili bir konuya abone olmak için şu komutu kullanın:

```
run MQTTV3Sample -a subscribe
```

Windowsüzerinde bir ileti yayınlamak için şu komutu kullanın:

```
run MQTTV3Sample
```

MQTT istemci programlama kavramları

Bu bölümde açıklanan kavramlar, MQTT protocolile ilgili istemci kitaplıklarını anlamanıza yardımcı olur. Kavramlar, istemci kitaplıklarına eşlik eden API belgelerini tamamlar.

En son bilgiler ve yüklemeler için aşağıdaki kaynaklara bakın:

- Eclipse Paho projesi ve MQTT.org, çeşitli programlama dilleri için en son telemetri istemcilerini ve örneklerini ücretsiz olarak karşıdan yüklüyor. IBM MQ Telemetry Transport' yi yayınlamak ve abone olmak ve güvenlik özellikleri eklemek için örnek programlar geliştirmenize yardımcı olması amacıyla bu siteleri kullanın.
- IBM Messaging Telemetry Clients SupportPac artık karşıdan yüklenebilir durumda değil. Önceden karşıdan yüklenen bir kopyanızı varsa, bu kopyanın içeriği şöyledir:
  - IBM Messaging Telemetry Clients SupportPac MA9B sürümü, derlenmiş bir örnek uygulama (`mqttv3app.jar`) ve ilişkili istemci kitaplığı (`mqttv3.jar`) içeriyordu. Bunlar aşağıdaki dizinlerde sağlanmıştır:
    - `ma9b/SDK/clients/java/org.eclipse.paho.sample.mqttv3app.jar`
    - `ma9b/SDK/clients/java/org.eclipse.paho.client.mqttv3.jar`
  - Bu SupportPac' ın MA9C sürümünde `/SDK/` dizini ve içeriği kaldırılmıştır:
    - Yalnızca örnek uygulama (`mqttv3app.jar`) için kaynak sağlandı. Şu ldı ki var var:

      ```
      ma9c/clients/java/samples/org/eclipse/paho/sample/mqttv3app/*.java
      ```

    - Derlenmiş istemci kitaplığı sağlandı. Şu ldı ki var var:

      ```
      ma9c/clients/java/org.eclipse.paho.client.mqttv3-1.0.2.jar
      ```

Bir MQTT istemcisi geliştirmek ve çalıştırmak için bu kaynakları istemci aygıtına kopyalamanız ya da kurmanız gerekir. Ayrı bir istemci yürütme ortamı kurmanız gerekmez.

İstemcilere ilişkin lisanslama koşulları, istemcileri bağladığınız sunucuyla ilişkilendirilir.

MQTT istemci kitaplıkları, MQTT protocol' in başvuru somutlamalarıdır. Kendi istemcilerinizi farklı aygıt platformlarına uygun farklı dillerde uygulayabilirsiniz. Bkz. IBM MQ Telemetry Transport biçimi ve iletişim kuralı.

API belgeleri, istemcinin bağlı olduğu MQTT sunucusu hakkında herhangi bir varsayımda bulunmaz. Farklı sunuculara bağlanıldığında istemcinin davranışı biraz farklı olabilir. Aşağıdaki tanımlar, IBM MQ telemetri hizmetine bağlandığında istemcinin davranışını açıklar.

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback`' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Geri Çağırmalar

Not: `MqttCallback`üzerinde yapılan en son değişiklikler için Eclipse Paho web sitesine bakın. Örneğin `MqttCallback` , istemcinin Paho sürümünde bir Arabirim olarak tanımlanır ve zamanuyumsuz yöntemler Paho `MqttAsyncClient` sınıfı tarafından sağlanır.

`MqttCallback` arabiriminin üç geri çağırma yöntemi vardır:

`connectionLost(java.lang.Throwable cause)`

Bir iletişim hatası bağlantının kesilmesine neden olduğunda `connectionLost` çağrılır. Bağlantı kurulduktan sonra sunucudaki bir hatanın sonucu olarak sunucu bağlantıyı keserse de çağrılır. Sunucu hataları kuyruk yöneticisi hata günlüğüne kaydedilir. Sunucu istemciyle bağlantıyı keser ve istemci `MqttCallback.connectionLost`' i çağırır.

İstemci uygulamasıyla aynı iş parçacığında kural dışı durum olarak yayınlanan tek uzak hatalar `MqttClient.connect`kural dışı durumlarıdır. Bağlantı kurulduktan sonra sunucu tarafından saptanan hatalar, `MqttCallback.connectionLost` geri çağırma yöntemine `throwable`olarak geri bildirilir.

`connectionLost` ile sonuçlanan tipik sunucu hataları yetkilendirme hatalarıdır. Örneğin, telemetri sunucusu, konu üzerinde yayınlama yetkisi olmayan bir istemci adına bir konu üzerinde yayınlama girişiminde bulunur. `MQCC_FAIL` durum kodunun telemetri sunucusuna döndürülmesiyle sonuçlanan herhangi bir şey, bağlantının atılması ile sonuçlanabilir.

`deliveryComplete(IMqttDeliveryToken token)`

`deliveryComplete` , bir teslim simgesini istemci uygulamasına geri aktarmak için MQTT istemcisi tarafından çağrılır; bkz. “Teslim belirteçleri” sayfa 1197. Geri çağırma, teslim belirtecini kullanarak yayınlanan iletiye `token.getMessage`yöntemiyle erişebilir.

Uygulama geri çağırma, `deliveryComplete` yöntemi tarafından çağrıldıktan sonra denetimi MQTT istemcisine döndürdüğünde teslim tamamlanır. Teslim tamamlanıncaya kadar, `QoS` 1 ya da 2 içeren iletiler kalıcılık sınıfı tarafından korunur.

`deliveryComplete` çağrısı, uygulama ile kalıcılık sınıfı arasında bir eşitleme noktasıdır. `deliveryComplete` yöntemi aynı ileti için hiçbir zaman iki kez çağrılmaz.

Uygulama `deliveryComplete` 'dan MQTT istemcisine geri döndüğünde, istemci `MqttClientPersistence.remove` ' i `QoS` 1 ya da 2içeren iletiler için çağırır. `MqttClientPersistence.remove` , yayınlanan iletinin yerel olarak saklanan kopyasını siler.

Bir hareket işleme perspektifinden `deliveryComplete` çağrısı, teslimi kesinleştirmeye ilişkin tek aşamalı bir harekettir. Geri çağırma sırasında işlem başarısız olursa, yayınlanan iletinin yerel kopyasını silmek için istemci yeniden başlatıldığında `MqttClientPersistence.remove` yeniden çağrılır. Geri çağırma yeniden çağrılmadı. Teslim edilen iletilerin günlüğünü saklamak için geri çağırmayı kullanıyorsanız, günlüğü MQTT istemcisiyle eşitleyemezsiniz. Bir günlüğü güvenilir bir şekilde saklamak istiyorsanız, `MqttClientPersistence` sınıfında günlüğü güncelleyin.

Teslim belirteci ve iletiye ana uygulama iş parçacığı ve MQTT istemcisi tarafından başvurulur. MQTT istemcisi, teslim tamamlandığında `MqttMessage` nesnesini ve istemci bağlantısını kestiğinde teslim simgesi nesnesini çıkartır. İstemci uygulaması başvuruyu silirse, teslim tamamlandıktan sonra `MqttMessage` nesnesi atık olarak toplanabilir. Teslim simgesi, oturum bağlantısı kesildikten sonra atık toplanabilir.

Bir ileti yayınlandıktan sonra `IMqttDeliveryToken` ve `MqttMessage` özniteliklerini alabilirsiniz. İleti yayınlandıktan sonra herhangi bir `MqttMessage` özniteliğini ayarlama girişiminde bulunursanız, sonuç tanımlanmaz.

İstemci önceki oturuma aynı `ClientIdentifier` ile yeniden bağlanırsa, MQTT istemcisi teslim onaylarını işlemeye devam eder; bkz. “Oturumları temizle” sayfa 1194. MQTT istemci uygulaması önceki oturum için `MqttClient.CleanSession` değerini `false` olarak ayarlamalı ve yeni oturumda `false` olarak ayarlamalıdır. MQTT istemcisi, bekleyen teslimatlar için yeni oturumda yeni teslim belirteçleri ve ileti nesneleri oluşturur. `MqttClientPersistence` sınıfını kullanarak nesneleri kurtarır. Uygulama istemcisinin hala eski teslim belirteçlerine ve iletilerine başvuruları varsa, bunlara ilişkin başvuruyu kaldırma. Önceki oturumda başlatılan ve bu oturumda tamamlanan teslimatlar için yeni oturumda uygulama geri çağırma çağrılır.

Bekleyen bir teslim tamamlandığında, uygulama istemcisi bağlandıktan sonra uygulama geri çağrısı çağrılır. Uygulama istemcisi bağlanmadan önce, `MqttClient.getPendingDeliveryTokens` yöntemini kullanarak bekleyen teslimatları alabilir.

İstemci uygulamasının yayınlanan ileti nesnesini ve bilgi yükü byte dizisini yarattığını fark edin. MQTT istemcisi bu nesnelere gönderme yapar. `token.getMessage` yönteminde teslim simgesi tarafından döndürülen ileti nesnesinin, istemci tarafından yaratılan ileti nesnesiyle aynı olması gerekmez. Yeni bir MQTT istemcisi eşgörünümü teslim simgesini yeniden oluştururs a, `MqttClientPersistence` sınıfı `MqttMessage` nesnesini yeniden oluşturur. Tutarlılık için `token.getMessage` , `token.isCompleted` `true`eise, ileti nesnesinin uygulama istemcisi ya da `MqttClientPersistence` sınıfı tarafından oluşturulmasından bağımsız olarak `null` değerini döndürür.

messageArrived(String topic, MqttMessage message)

`messageArrived` , bir abonelik konusuyla eşleşen istemci için bir yayın geldiğinde çağrılır. `konu` , abonelik süzgeci değil, yayın konusudur. Süzgeç genel arama karakterleri içeriyorsa, ikisi farklı olabilir.

Konu, istemci tarafından yaratılan birden çok abonelikle eşleşirse, istemci yayının birden çok kopyasını alır. Bir istemci aynı zamanda abone olduğu bir konuyu yayınlarsa, kendi yayınının bir kopyasını alır.

Bir ileti 1 ya da 2 QoS ile gönderilirse, ileti MQTT istemcisinden önce `messageArrived` `MqttClientPersistence` sınıfı tarafından saklanır. `messageArrived` , `deliveryCompletegibi` davranır: Yayın için yalnızca bir kez çağrılır ve `messageArrived` , MQTT istemcisine döndüğünde yayının yerel kopyası `MqttClientPersistence.remove` tarafından kaldırılır. MQTT istemcisi, `messageArrived` MQTT istemcisine döndüğünde konuya ve iletiye başvurularını bırakır. Uygulama istemcisi nesnelere ilişkin bir başvuruyu tutmamışsa, konu ve ileti nesneleri atık toplanır.

Geri çağırmalar, geri çağırmalar ve istemci uygulaması eşitlemesi

MQTT istemcisi, ana uygulama iş parçacığına ayrı bir iş parçacığında geri çağırma yöntemini çağırır. İstemci uygulaması geri çağırma için bir iş parçacığı yaratmaz, MQTT istemcisi tarafından yaratılır.

MQTT istemcisi, geri çağırma yöntemlerini uyumlulaştırır. Bir kerede tek bir geri çağırma yöntemi çalıştırılır. Eşitleme, hangi yayınların teslim edildiğini gösteren bir nesnenin güncellenmesini kolaylaştırır. `MqttCallback.deliveryComplete` ' in bir eşgörünümü aynı anda çalışır ve böylece daha fazla eşitleme yapmadan sayımı güncellemeniz güvenlidir. Aynı anda yalnızca bir yayın da gelir. `messageArrived` yöntemindeki kodunuz, bir nesneyi eşitlemeden güncelleyebilir. Sayımı ya da güncellenmekte olan nesneyi başka bir iş parçacığında kastediyorsanız, çeteleyi ya da nesneyi eşitleyin.

Teslim belirteci, ana uygulama iş parçacığı ile bir yayının teslimi arasında bir eşitleme mekanizması sağlar. `token.waitForCompletion` yöntemi, belirli bir yayının teslimi tamamlanıncaya ya da isteğe bağlı bir zamanaşımı süresi doluncaya kadar bekler. `token.waitForCompletion` ' i bir kerede bir yayını işlemek için aşağıdaki şekilde kullanabilirsiniz.

`MqttCallback.deliveryComplete` yöntemiyle eşitlemek için. Yalnızca `MqttCallback.deliveryComplete` MQTT Client 'a geri döndüğünde `token.waitForCompletion` devam eder. Bu mekanizmayı kullanarak, kod ana uygulama iş parçacığında çalıştırılmadan önce `MqttCallback.deliveryComplete` içinde çalışan kodu eşitleyebilirsiniz.

Her yayının teslim edilmesini beklemeden yayınlamak isteseniz, ancak tüm yayınlar teslim edildiğinde onay isteseniz ne olurdu? Tek bir iş parçacığında yayınlarsanız, gönderilecek son yayın da teslim edilecek son yayın olur.

Sunucuya gönderilen isteklerin eşitlenmesi

Çizelge 188 sayfa 1193 , MQTT Java istemcisinde sunucuya istek gönderen yöntemleri açıklar. Uygulama istemcisi belirsiz bir zamanaşımı belirlemezse, istemci sunucu için hiçbir zaman süresiz olarak beklemez. İstemci askıda kaldıysa, bu bir uygulama programlama sorunu ya da MQTT istemcisindeki bir hata olabilir.

Çizelge 188. Sunucuya istekle sonuçlanan yöntemlerin eşitleme davranışı		
Yöntem	**Eşitleme**	**Zamanaşımı aralığı**
`MqttClient.Connect`	Sunucuyla bağlantı kurulmasını bekler.	Varsayılan değer olarak 30 saniye ya da bir parametre tarafından ayarlandıktan sonra bir özel durum ortaya çıktı.
`MqttClient.Disconnect`	MQTT istemcisinin yapması gereken işleri bitirmesini ve TCP/IP oturumunun bağlantısını kesmesini bekler.	
`MqttClient.Subscribe`	Abonenin ya da UnSubscribe yönteminin tamamlanmasını bekler.	
`MqttClient.UnSubscribe`		
`MqttClient.Publish`	İsteği MQTT istemcisine geçirdikten sonra hemen uygulama iş parçacığına döner.	Yok.
`IMqttDeliveryToken.waitFor Completion`	Teslim simgesinin döndürülmesini bekler.	Belirsiz ya da parametre olarak ayarlanmış.

İlgili kavramlar

Oturumları temizle

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını

Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı

Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar

Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri

Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma

Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

Oturumları temizle

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

Bir MQTT istemci uygulamasını `MqttClient.connect` yöntemini kullanarak bağladığınızda, istemci bağlantıyı istemci tanıtıcısını ve sunucunun adresini kullanarak tanımlar. Sunucu, oturum bilgilerinin önceki bir sunucu bağlantısından saklanıp saklanmadığını denetler. Önceki bir oturum hala varsa ve `cleanSession=true`varsa, istemcideki ve sunucudaki önceki oturum bilgileri temizlenir. `cleanSession=false` ise, önceki oturum sürdürülür. Önceki bir oturum yoksa, yeni bir oturum başlatılır.

Not: IBM MQ Administrator, açık bir oturumu zorla kapatabilir ve tüm oturum bilgilerini silebilir. İstemci oturumu `cleanSession=false`ile yeniden açarsa, yeni bir oturum başlatılır.

Yayınlar

İstemciyi bağlamadan önce varsayılan `MqttConnectOptions`değerini kullanır ya da `MqttConnectOptions.cleanSession` değerini `true` olarak ayarlarsanız, istemci bağlandığında istemciye ilişkin bekleyen tüm yayın teslimatları kaldırılır.

Temiz oturum ayarı, `QoS=0`ile gönderilen yayınları etkilemez. `QoS=1` ve `QoS=2`için, `cleanSession=true` ' un kullanılması bir yayının kaybedilmesiyle sonuçlanabilir.

Abonelikler

İstemciyi bağlamadan önce varsayılan `MqttConnectOptions`değerini kullanır ya da `MqttConnectOptions.cleanSession` ayarını `true` olarak ayarlarsanız, istemci bağlandığında istemciye ilişkin eski abonelikler kaldırılır. Oturum sırasında istemcinin yaptığı yeni abonelikler, bağlantısı kesildiğinde kaldırılır.

Bağlanmadan önce `MqttConnectOptions.cleanSession` değerini `false` olarak ayarlarsanız, istemcinin yarattığı abonelikler, bağlanmadan önce istemci için var olan tüm aboneliklere eklenir. İstemci bağlantıyı kestiğinde tüm abonelikler etkin kalır.

`cleanSession` özniteliğinin abonelikleri etkileme şeklini anlamanın başka bir yolu da bunu kalıcı bir öznitelik olarak düşünmektir. İstemci, varsayılan kipinde (`cleanSession=true`) abonelikler yaratır ve yalnızca oturum kapsamında yayınlar alır. `cleanSession=false`alternatif kipinde abonelikler dayanıklıdır. İstemci bağlanabilir ve bağlantısını kesebilir ve abonelikleri etkin olarak kalır. İstemci yeniden bağlandığında, teslim edilmemiş yayınlar alır. Bağlıyken, kendi adına etkin olan abonelikler kümesini değiştirebilir.

Bağlanmadan önce `cleanSession` kipini ayarlamanız gerekir; kip tüm oturum için geçerli olur. Ayarını değiştirmek için istemcinin bağlantısını kesmeniz ve yeniden bağlanmanız gerekir. Kipleri `cleanSession=false` kullanarak `cleanSession=true`kipine çevirirseniz, istemciye ilişkin önceki tüm abonelikler ve alınmamış yayınlar atılır.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback`' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

İstemci tanıtıcısı
İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını
Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı
Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar
Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma
Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

İstemci tanıtıcısı, MQTT sisteminin yönetiminde kullanılır. Potansiyel olarak yüz binlerce müşteriyi yönetmek için belirli bir müşteriyi hızla tanımlayabilmeniz gerekir. Örneğin, bir aygıtın arızalı olduğunu ve yardım masasını çalan bir müşteri tarafından size bildirim gönderildiğini varsayın. Müşterinin aygıtı tanımlayabilmesi gerekir ve bu tanımlamayı genellikle istemciye bağlı olan sunucuyla ilişkilendirebilmeniz gerekir.

MQTT istemci bağlantılarına göz attığınızda, her bağlantı istemci tanıtıcısıyla etiketlenir. Bu tanıtıcıyı aygıt ve sunucuyla en iyi şekilde nasıl eşleyeceğinize karar vermenize yardımcı olmak için kendinize aşağıdaki soruları sorun:

- Her bir aygıtı bir istemci tanıtıcısı ve bir sunucuyla eşleyen bir veritabanını korumak ve kullanmak uygun olur mu?
- Aygıtın adı, bağlı olduğu sunucuyu tanımlayabilir mi?
- Bir istemci tanıtıcısını fiziksel bir aygıtla eşleyen bir arama çizelgesine mi ihtiyacınız var?
- İstemci tanıtıcısı, istemcide çalışan belirli bir aygıtı, kullanıcıyı ya da uygulamayı tanımlıyor mu?
- Bir müşteri arızalı bir aygıtı yenisiyle değiştirirse, yeni aygıt eski aygıtla aynı tanıtıcıya sahip mi, yoksa yeni bir tanıtıcı mı ayırıyorsunuz? (Bir fiziksel aygıtı değiştirir ve aynı tanıtıcıyı tutarsanız, bekleyen yayınlar ve etkin abonelikler otomatik olarak yeni aygıta aktarılır.)

İstemci tanıtıcılarının benzersiz olduğundan emin olmak için bir sisteme ve istemcideki tanıtıcıyı ayarlamak için güvenilir bir işleme sahip olmanız gerekir. İstemci aygıtı, kullanıcı arabirimi olmayan bir "kara kutu" ise, aygıtı bir istemci tanıtıcısıyla üretebilir ya da etkinleştirilmeden önce aygıtı yapılandıran bir yazılım kuruluşu ve yapılandırma işlemine sahip olabilirdiniz.

Tanıtıcıyı kısa ve benzersiz tutmak için 48 bitlik aygıt MAC adresinden bir istemci tanıtıcısı oluşturabilirsiniz. İletim büyüklüğü önemli bir sorun değilse, adresin daha kolay denetlenmesi için kalan 17 baytı kullanabilirsiniz.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback`' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle
MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını
Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı
Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar
Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma
Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

Teslim belirteçleri

Bir istemci bir konuda yayınlandığında yeni bir teslim simgesi oluşturulur. Bir yayının teslimini izlemek ya da teslim tamamlanıncaya kadar istemci uygulamasını engellemek için teslim simgesini kullanın.

Simge bir `MqttDeliveryToken` nesnesidir. MqttTopic.publish () yöntemi çağrılarak yaratılır ve istemci oturumunun bağlantısı kesilinceye ve teslim tamamlanıncaya kadar MQTT istemcisi tarafından alıkonur.

Simgenin normal kullanımı, teslimatın tamamlanıp tamamlanmadığını kontrol etmektir. `token.waitForCompletion`çağrısı için döndürülen belirteci kullanarak teslim tamamlanıncaya kadar istemci uygulamasını engelleyin. Alternatif olarak, bir `MqttCallBack` işleyicisi sağlayın. MQTT istemcisi, yayının teslim edilmesinin bir parçası olarak beklediği tüm onayları aldığında, parametre olarak teslim simgesini geçirme `MqttCallBack.deliveryComplete` öğesini çağırır.

Teslimat tamamlanıncaya kadar, `token.getMessage`' i arayarak iade edilen teslim simgesini kullanarak yayını inceleyebilirsiniz.

Tamamlanan teslimatlar
Teslimatların tamamlanması zamanuyumsuzdur ve yayınla ilişkili hizmet kalitesine bağlıdır.

En çok bir kez

`QoS=0`
Teslim, `MqttTopic.publish`' den dönüşte hemen tamamlanır. `MqttCallback.deliveryComplete` hemen çağrılır.

En az bir kez

`QoS=1`
Kuyruk yöneticisinden yayına bir alındı bildirimi alındığında teslim tamamlanır. Alındı bildirimi alındığında `MqttCallback.deliveryComplete` çağrılır. İletişim yavaşsa ya da güvenilmezse, ileti `MqttCallback.deliveryComplete` çağrılmadan önce bir kereden fazla teslim edilebilir.

Tam olarak bir kez

`QoS=2`
Müşteri, yayının abonelere yayınlandığını belirten bir tamamlanma iletisi aldığında teslim tamamlanır. Yayın iletisi alınır alınmaz `MqttCallback.deliveryComplete` çağrılır. Tamamlanma iletisini beklemez.

Sık rastlanmayan durumlarda, istemci uygulamanız `MqttCallback.deliveryComplete` ' den MQTT istemcisine geri dönmeyebilir. `MqttCallback.deliveryComplete` çağrıldığından teslimatın tamamlandığını biliyorsunuz. İstemci aynı oturumu yeniden başlatırsa, `MqttCallback.deliveryComplete` yeniden çağrılamaz.

Tamamlanmamış teslimatlar

Müşteri oturumunun bağlantısı kesildikten sonra teslim tamamlanmazsa, istemciyi yeniden bağlayabilir ve teslimatı tamamlayabilirsiniz. Bir iletinin teslimini, ileti `MqttConnectionOptions` özniteliği `false`olarak ayarlanmış bir oturumda yayınlandıysa tamamlayabilirsiniz.

Aynı istemci tanıtıcısını ve sunucu adresini kullanarak istemciyi oluşturun ve `cleanSession` `MqttConnectionOptions` özniteliğini yeniden `false` olarak ayarlayarak bağlanın. `cleanSession` ayarını `true`olarak ayarlarsanız, bekleyen teslim belirteçleri atılır.

`MqttClient.getPendingDeliveryTokens`' i arayarak bekleyen teslim olup olmadığını denetleyebilirsiniz. İstemciyi bağlamadan önce `MqttClient.getPendingDeliveryTokens` ' i arayabilirsiniz.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback`' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle
MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı
İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Son vasiyet ve vasiyet yayını
Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı
Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar
Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma
Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

Son vasiyet ve vasiyet yayını

Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

Son vasiyet ve vasiyet için bir konu oluşturun. `MQTTManagement/Connections/`*`server URI`*`/`*`client identifer`*`/Lost`gibi bir konu oluşturabilirsiniz.

`MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)` yöntemini kullanarak bir "son vasiyet ve vasiyet" ayarlayın.

`lastWillPayload` iletisinde bir zaman damgası oluşturmayı düşünün. İstemcinin ve bağlantı koşullarının belirlenmesine yardımcı olan diğer istemci bilgilerini ekleyin. `MqttConnectionOptions` nesnesini `MqttClient` oluşturucusuna geçirin.

İletiyi IBM MQiçinde kalıcı yapmak ve teslimatı garanti etmek için `lastWillQos` değerini 1 ya da 2olarak ayarlayın. Son kayıp bağlantı bilgilerini korumak için `lastWillRetained` değerini `true`olarak ayarlayın.

Bağlantı beklenmedik bir şekilde sona ererse, "son vasiyet ve vasiyet" yayını abonelere gönderilir. Bağlantı, istemci `MqttClient.disconnect` yöntemini çağırmadan sona ererse gönderilir.

Bağlantıları izlemek için, bağlantıları ve programlanmış bağlantıları kaydetmek için diğer yayınlarla birlikte "son irade ve vasiyet" yayınını tamamlayın.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback`' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle
MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı
İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

MQTT istemcilerinde ileti kalıcılığı
Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar
Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma

Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

MQTT istemcilerinde ileti kalıcılığı

Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

MQTT' de ileti kalıcılığını iki yönü vardır: İletinin nasıl aktarıldığı ve IBM MQ içinde kalıcı ileti olarak kuyruğa alınıp alınmadığı.

1. MQTT istemcisi, ileti kalıcılığını hizmet kalitesiyle eşler. Bir ileti için seçtiğiniz hizmet kalitesine bağlı olarak, ileti kalıcı olarak yapılır. Gerekli hizmet kalitesini uygulamak için ileti kalıcılığı gereklidir.

   "En çok bir kez"(`QoS=0`) belirtirseniz, istemci ileti yayınlanır yayınlanmaz iletiyi atar. İletinin yukarı akımda işlenmesinde herhangi bir hata oluşursa, ileti yeniden gönderilmez. İstemci etkin durumda kalsa bile, ileti yeniden gönderilmez. `QoS=0` iletilerinin davranışı, IBM MQ hızlı, kalıcı olmayan iletileriyle aynıdır.

   Bir ileti, `QoS` değeri 1 ya da 2 olan bir istemci tarafından yayınlandıysa, kalıcı hale getirilir. İleti yerel olarak saklanır ve artık "en az bir kez", `QoS=1`ya da "tam olarak bir kez", `QoS=2`teslimatı garanti etmek gerekmediği zaman istemciden atılır.
2. Bir ileti `QoS` 1 ya da 2 olarak işaretliyse, IBM MQ içinde kalıcı ileti olarak kuyruğa alınır. `QoS=0`olarak işaretliyse, IBM MQ içinde kalıcı olmayan bir ileti olarak kuyruğa alınır. İleti kanalında `NPMSPEED` özniteliği `FAST`olarak ayarlanmadıysa, IBM MQ kalıcı olmayan iletiler kuyruk yöneticileri arasında "tam olarak bir kez"aktarılır.

Kalıcı bir yayın, bir istemci uygulaması tarafından alınıncaya kadar istemcide saklanır. `QoS=2`için, uygulama geri çağırma denetimi döndürdüğünde yayın istemciden atılır. `QoS=1` için, bir hata oluşursa uygulama yayını yeniden alabilir. `QoS=0`için, geri çağırma yayını bir kereden fazla almaz. Bir hata oluşursa ya da istemcinin yayın sırasında bağlantısı kesilirse yayını almayabilir.

Bir konuya abone olduğunuzda, abonenin kalıcı saklama yetenekleriyle eşleşecek iletileri aldığı `QoS` ' i azaltabilirsiniz. Daha yüksek bir `QoS` ürününde oluşturulan yayınlar, abonenin istediği en yüksek `QoS` ile gönderilir.

İletilerin saklanması

Küçük aygıtlarda veri depolamanın uygulanması büyük farklılıklar gösterir. MQTT istemcisi tarafından yönetilen depolama alanındaki kalıcı iletilerin geçici olarak kaydedilmesi modeli çok yavaş olabilir ya da çok fazla depolama alanı istiyor olabilir. Mobil aygıtlarda, mobil işletim sistemi MQTT iletileri için ideal bir depolama hizmeti sağlayabilir.

Küçük aygıtların kısıtlamalarını karşılamada esneklik sağlamak için MQTT istemcisinin iki kalıcı arabirimi vardır. Arabirimler, kalıcı iletilerin saklanmasında yer alan işlemleri tanımlar. Arabirimler, MQTT client for Javaiçin API belgelerinde açıklanır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu. Arabirimleri bir aygıta uyacak şekilde uygulayabilirsiniz. Java SE üzerinde çalışan MQTT istemcisi, dosya sisteminde kalıcı iletileri saklayan arabirimlerin varsayılan bir uygulamasına sahiptir. `java.io` paketini kullanır.

Kalıcılık sınıfları

`MqttClientPersistence`
`MqttClientPersistence` somutlamasının bir eşgörünümünü, `MqttClient` oluşturucusunun bir değiştirgesi olarak MQTT istemcisine geçirin. `MqttClientPersistence` değiştirgesini `MqttClient` oluşturucusundan çıkarırsanız, MQTT istemcisi kalıcı iletileri `MqttDefaultFilePersistence`sınıfını kullanarak saklar.

`MqttPersistable`
`MqttClientPersistence` , `MqttPersistable` nesnelerini bir depolama anahtarı kullanarak alır ve yerleştirir. `MqttDefaultFilePersistence`kullanmıyorsanız, `MqttClientPersistence` uygulamasının yanı sıra bir `MqttPersistable` somutlaması da sağlamanız gerekir.

`MqttDefaultFilePersistence`

MQTT istemcisi `MqttDefaultFilePersistence` sınıfını sağlar. `MqttDefaultFilePersistence` ' i istemci uygulamanızda somutlaştırırsanız, kalıcı iletilerin saklanacağı dizini `MqttDefaultFilePersistence` oluşturucusunun bir değiştirgesi olarak sağlayabilirsiniz.

Diğer bir seçenek olarak, MQTT istemcisi `MqttDefaultFilePersistence` dosyasını somutlaştırabilir ve dosyaları aşağıdaki varsayılan dizine yerleştirebilir:

`client identifier` `-tcp` *`hostname portnumber`*

Dizin adı dizgisinden aşağıdaki karakterler kaldırılır:

`"\", "\\", "/", ":"` ve `" "`

Dizin yolu, `rcp.data`sistem özelliğinin değeridir; `rcp.data` ayarlanmazsa, yol, `usr.data`sistem özelliğinin değeridir; burada

- `rcp.data` , bir OSGi ya da Eclipse Rich Client Platform (RCP) kuruluşuyla ilişkili bir özelliktir.
- `usr.data` , uygulamayı başlatan Java komutunun başlatıldığı dizindir.

İlgili kavramlar
MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback` ' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle
MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı
İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını
Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

Yayınlar
Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma
Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

Yayınlar

Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

`MqttMessage` , bilgi yükü olarak bir bayt dizisine sahiptir. Mesajları mümkün olduğunca küçük tutmayı hedefle. MQTT protocol tarafından izin verilen ileti uzunluğu üst sınırı 250 MB 'dir.

Genellikle, bir MQTT istemci programı ileti içeriğini değiştirmek için `java.lang.String` ya da `java.lang.StringBuffer` kullanır. Kolaylık olması için, `MqttMessage` sınıfının bilgi yükünü dizgiye dönüştürmek için bir `toString` yöntemi vardır. Bir `java.lang.String` ya da `java.lang.StringBuffer`içinden bayt dizisi bilgi yükünü oluşturmak için `getBytes` yöntemini kullanın.

`getBytes` yöntemi, bir dizgiyi platform için varsayılan karakter kümesine dönüştürür. Varsayılan karakter kümesi genellikle `UTF-8`' dir. Yalnızca metin içeren MQTT yayınları genellikle `UTF-8`içinde kodlanır. Varsayılan karakter kümesini geçersiz kılmak için `getBytes("UTF8")` yöntemini kullanın.

IBM MQiçinde bir MQTT yayını, `jms-byte` iletisi olarak alınır. İleti, `<mqtt>`ve `<mqps>` klasörü içeren bir `MQRFH2` klasörünü içerir. `<mqtt>` klasörü, `clientId`, `msgId`ve `qos`öğelerini içerir, ancak bu içerik gelecekte değişebilir.

`MqttMessage` , üç ek özniteliğe sahiptir: hizmet kalitesi, alıkonup saklanmadığı ve yinelenen olup olmadığı. Yinelenen işaret yalnızca hizmet kalitesi "en az bir kez" ya da "tam olarak bir kez" olduğunda ayarlanır. İleti daha önce gönderildiyse ve MQTT istemcisi tarafından yeterince hızlı onaylanmamışsa, yinelenen öznitelik `true`olarak ayarlanmış olarak ileti yeniden gönderilir.

Yayıncılık

MQTT istemci uygulamasında bir yayın oluşturmak için bir `MqttMessage`oluşturun. Bilgi yükünü, hizmet kalitesini ve alıkonup saklanmadığını ayarlayın ve `MqttTopic.publish(MqttMessage message)` yöntemini çağırın; `MqttDeliveryToken` döndürülür ve yayının tamamlanması zamanuyumsuz olur.

Diğer bir seçenek olarak, MQTT istemcisi bir yayın yaratırken `MqttTopic.publish(byte [] payload, int qos, boolean retained)` yöntemindeki değiştirgelerden sizin için geçici bir ileti nesnesi yaratabilir.

Yayında "en az bir kez" ya da "tam olarak bir kez" hizmet kalitesi varsa, `QoS=1` ya da `QoS=2`, MQTT istemcisi `MqttClientPersistence` arabirimini çağırır. Uygulamaya bir teslim simgesi döndürmeden önce iletiyi saklamak için `MqttClientPersistence` ' i çağırır.

Uygulama, `MqttDeliveryToken.waitForCompletion` yöntemini kullanarak ileti sunucuya teslim edilinceye kadar engellemeyi seçebilir. Diğer bir seçenek olarak, uygulama engellemeden devam

edebilir. Yayınların engellemeden teslim edilip edilmediğini denetlemek istiyorsanız, MQTT istemcisine `MqttCallback` uygulayan bir geri çağırma sınıfının somut örneğini kaydettirin. MQTT istemcisi, yayın teslim edilir edilmez `MqttCallback.deliveryComplete` yöntemini çağırır. Hizmet kalitesine bağlı olarak, `QoS=0`için teslimat neredeyse anında olabilir ya da `QoS=2`için biraz zaman alabilir.

Teslimatın tamamlandığını yoklamak için `MqttDeliveryToken.isComplete` yöntemini kullanın. `MqttDeliveryToken.isComplete` değeri `false`ise, ileti içeriğini almak için `MqttDeliveryToken.getMessage` ' i arayabilirsiniz. `MqttDeliveryToken.isComplete` çağrısının sonucu `true`ise, ileti atılır ve `MqttDeliveryToken.getMessage` çağrılırsa boş değerli gösterge kural dışı durumu oluşur. `MqttDeliveryToken.getMessage` ile `MqttDeliveryToken.isComplete`arasında yerleşik eşitleme yoktur.

İstemci, bekleyen teslim belirteçlerini almadan önce bağlantısını keserse, istemcinin yeni bir eşgörünümü bağlanmadan önce bekleyen teslim belirteçlerini sorgulayabilir. İstemci bağlanıncaya kadar, yeni teslimatlar tamamlanmaz ve `MqttDeliveryToken.getMessage`' i aramanız güvenlidir. Hangi yayınların teslim edilmediğini öğrenmek için `MqttDeliveryToken.getMessage` yöntemini kullanın. `MqttConnectOptions.cleanSession` varsayılan değeri olan `true`ile bağlantı kurarsanız, bekleyen teslim belirteçleri atılır.

abone olunması

Kuyruk yöneticisi, MQTT abonesine gönderilecek yayınlar yaratmaktan sorumludur. Kuyruk yöneticisi, MQTT istemcisi tarafından oluşturulan bir abonelikte konu süzgecinin bir yayındaki konu dizgisiyle eşleşip eşleşmediğini denetler. Eşleşme tam eşleşme olabilir ya da eşleşme genel arama karakterleri içerebilir. Yayın, kuyruk yöneticisi tarafından aboneye iletilmeden önce kuyruk yöneticisi, yayınla ilişkili konu özniteliklerini denetler. Bir denetim konusu nesnesinin kullanıcıya abone olma yetkisi verip vermediğini tanımlamak için Genel arama karakterleri içeren bir konu dizgisi kullanılarak abone olunması başlıklı konuda açıklanan arama yordamını izler.

MQTT istemcisi "en az bir kez" hizmet kalitesine sahip bir yayın aldığında, yayını işlemek için `MqttCallback.messageArrived` yöntemini çağırır. Yayının hizmet kalitesi "tam olarak bir kez" ise `QoS=2`, MQTT istemcisi iletiyi alındığında saklamak için `MqttClientPersistence` arabirimini çağırır. Daha sonra `MqttCallback.messageArrived`adını verdi.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback`' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle
MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı
İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını
Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı

Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma
Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri

Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Bir yayının hizmet kalitesi, `MqttMessage`özniteliğinin bir özniteliğidir. `MqttMessage.setQos`yöntemiyle ayarlanır.

`MqttClient.subscribe` yöntemi, bir konuda istemciye gönderilen yayınlara uygulanan hizmet kalitesini azaltabilir. Aboneye iletilen bir yayının hizmet kalitesi, yayının hizmet kalitesinden farklı olabilir. İki değerin altı bir yayını iletmek için kullanılır.

En çok bir kez
`QoS=0`

İleti en fazla bir kez teslim edilir ya da hiç teslim edilmez. Ağ üzerinden teslimatı kabul edilmez.
İleti saklanmadı. İstemcinin bağlantısı kesilirse ya da sunucu başarısız olursa ileti kaybolabilir.
`QoS=0` , en hızlı aktarma kipidir. Buna bazen "ateş ve unutmak" denir.
MQTT protocol , sunucuların `QoS=0` adresindeki yayınları bir istemciye iletmesini gerektirmez. Sunucunun yayını aldığı sırada istemcinin bağlantısı kesilirse, yayın sunucuya bağlı olarak atılabilir. Telemetry (MQXR) hizmeti, `QoS=0`ile gönderilen iletileri atmaz. Bunlar kalıcı olmayan iletiler olarak saklanır ve yalnızca kuyruk yöneticisi durduğunda atılır.

En az bir kez
`QoS=1`

`QoS=1` , varsayılan aktarma kipidir.
İleti her zaman en az bir kez teslim edilir. Gönderen bir alındı bildirimi almazsa, ileti, bir alındı bildirimi alınıncaya kadar DUP işareti ayarlanmış olarak yeniden gönderilir. Sonuç olarak, alıcı aynı iletiyi birden çok kez gönderebilir ve birden çok kez işleyebilir.
İleti, işleninceye kadar gönderen ve alıcıda yerel olarak saklanmalıdır.
İleti, iletiyi işledikten sonra alıcıdan silinir. Alıcı bir aracıysa, ileti abonelerine yayınlanır. Alıcı bir istemciyse, ileti abone uygulamasına teslim edilir. İleti silindikten sonra alıcı gönderene bir alındı bildirimi gönderir.
İleti, alıcıdan bir alındı bildirimi aldıktan sonra gönderenden silinir.

Tam olarak bir kez

QoS=2

Mesaj her zaman tam olarak bir kez teslim edilir.

İleti, işleninceye kadar gönderen ve alıcıda yerel olarak saklanmalıdır.

QoS=2 , en güvenli, ancak en yavaş aktarım kipidir. İleti gönderenden silinmeden önce gönderen ve alıcı arasında en az iki çift iletim gerektirir. İleti, ilk iletiden sonra alıcıda işlenebilir.

İlk iletim çiftinde, gönderen iletiyi iletir ve alıcıdan iletiyi sakladığına dair onay alır. Gönderen bir alındı bildirimi almazsa, ileti, bir alındı bildirimi alınıncaya kadar DUP işareti ayarlanmış olarak yeniden gönderilir.

İkinci iletim çiftinde gönderen, alıcıya "PUBREL"iletisini işlemeyi tamamlayabileceğini bildirir. Gönderen "PUBREL" iletisinin bir onayını almazsa, bir alındı bildirimi alınıncaya kadar "PUBREL" iletisi yeniden gönderilir. Gönderen, "PUBREL" iletisinin onayını aldığında kaydettiği iletiyi siler

Alıcı, iletiyi yeniden işlememesi koşuluyla, iletiyi birinci ya da ikinci aşamalarda işleyebilir. Alıcı bir aracıysa, iletiyi abonelere yayınlar. Alıcı bir istemciyse, iletiyi abone uygulamasına teslim eder. Alıcı, gönderene iletiyi işlemeyi tamamladığını bildiren bir tamamlanma iletisi gönderir.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback`' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını

Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı

Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar

Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Yayınları ve MQTT istemcilerini alıkoyma

Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler

Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir.

Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

Yayınları ve MQTT istemcilerini alıkoyma

Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Bir konudaki bir yayının korunup korunmayacağını belirtmek için `MqttMessage.setRetained` yöntemini kullanın.

Alıkonan bir yayın yarattığınızda ya da güncellediğinizde, yayını 1 ya da 2 QoS ile gönderin. Bunu 0 'lık bir QoS ile gönderirseniz, IBM MQ kalıcı olmayan bir alıkonan yayın yaratır. Kuyruk yöneticisi durursa yayın alıkonmaz.

Alıkonmayan bir yayını, korunan bir yayını olan bir konuda yayınlarsanız, alıkonan yayın bundan etkilenmez. Geçerli aboneler yeni yayını alır. Yeni aboneler önce alıkonan yayını alır, daha sonra yeni yayınları alır.

Bir ölçümün en son değerini kaydetmek için alıkonan bir yayını kullanabilirsiniz. Bir konuya ilişkin yeni aboneler hemen ölçümün en son değerini alır. Abonenin yayın konusuna son abone olduğundan bu yana yeni bir ölçüm alınmazsa ve abone yeniden abone olursa, abonenin konuyla ilgili en son tutulan yayını yeniden alır.

Alıkonan bir yayını silmek için iki seçeneğiniz vardır:

- **`CLEAR TOPICSTR`** MQSC komutunu çalıştırın.
- Sıfır uzunluklu bir alıkonan yayın yaratır. MQTT 3.1.1 belirtiminde belirtildiği gibi, bir konuya sıfır uzunluklu alıkonan ileti yayınlandıysa, o konuya ilişkin alıkonan ileti temizlenir.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback` ' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle
MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı
İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını
Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı
Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık

mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar
Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri
Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

Abonelikler

Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

`MqttClient.subscribe` yöntemlerini kullanarak bir ya da daha fazla konu süzgeci ve hizmet kalitesi parametresi geçirerek abonelikler oluşturun. Hizmet kalitesi parametresi, abonenin ileti almak için kullanmaya hazır olduğu hizmet kalitesi üst sınırını belirler. Bu istemciye gönderilen iletiler daha yüksek hizmet kalitesiyle teslim edilemez. Hizmet kalitesi, ileti yayınlandığında özgün değerin altına ve abonelik için belirtilen düzeye ayarlanır. İleti almak için varsayılan hizmet kalitesi en az bir kez olmak üzere `QoS=1`' dir.

Abonelik isteğinin kendisi `QoS=1`ile gönderilir.

Yayınlar, MQTT istemcisi `MqttCallback.messageArrived` yöntemini çağırdığında bir abone tarafından alınır. `messageArrived` yöntemi, iletinin yayınlandığı konu dizgisini de aboneye iletir.

`MqttClient.unsubscribe` yöntemlerini kullanarak bir aboneliği ya da bir küme ya da abonelikleri kaldırabilirsiniz.

IBM MQ komutu bir aboneliği kaldırabilir. Abonelikleri IBM MQ Explorerkomutunu kullanarak ya da **runmqsc** ya da PCF komutlarını kullanarak listeleyin. Tüm MQTT istemci abonelikleri adlandırılır. Onlara formun adı verilir: *`ClientIdentifier`*`:`*`Topic name`*

İstemciyi bağlamadan önce varsayılan `MqttConnectOptions`değerini kullanır ya da `MqttConnectOptions.cleanSession` ayarını `true` olarak ayarlarsanız, istemci bağlandığında istemciye ilişkin eski abonelikler kaldırılır. Oturum sırasında istemcinin yaptığı yeni abonelikler, bağlantısı kesildiğinde kaldırılır.

Bağlanmadan önce `MqttConnectOptions.cleanSession` değerini `false` olarak ayarlarsanız, istemcinin yarattığı abonelikler, bağlanmadan önce istemci için var olan tüm aboneliklere eklenir. İstemci bağlantıyı kestiğinde tüm abonelikler etkin kalır.

`cleanSession` özniteliğinin abonelikleri etkileme şeklini anlamanın başka bir yolu da bunu kalıcı bir öznitelik olarak düşünmektir. İstemci, varsayılan kipinde (`cleanSession=true`) abonelikler yaratır ve yalnızca oturum kapsamında yayınlar alır. `cleanSession=false`alternatif kipinde abonelikler dayanıklıdır. İstemci bağlanabilir ve bağlantısını kesebilir ve abonelikleri etkin olarak kalır. İstemci yeniden

bağlandığında, teslim edilmemiş yayınlar alır. Bağlıyken, kendi adına etkin olan abonelikler kümesini değiştirebilir.

Bağlanmadan önce `cleanSession` kipini ayarlamanız gerekir; kip tüm oturum için geçerli olur. Ayarını değiştirmek için istemcinin bağlantısını kesmeniz ve yeniden bağlanmanız gerekir. Kipleri `cleanSession=false` kullanarak `cleanSession=true`ekipine çevirirseniz, istemciye ilişkin önceki tüm abonelikler ve alınmamış yayınlar atılır.

Etkin aboneliklerle eşleşen yayınlar, yayınlanır yayınlanmaz istemciye gönderilir. İstemcinin bağlantısı kesilirse, istemci aynı istemci tanıtıcısıyla aynı sunucuya yeniden bağlanırsa ve `MqttConnectOptions.cleanSession` değeri `false`olarak ayarlanırsa istemciye gönderilir.

Belirli bir istemciye ilişkin abonelikler istemci tanıtıcısı tarafından tanımlanır. İstemciyi farklı bir istemci aygıtından aynı sunucuya yeniden bağlayabilir ve aynı aboneliklerle devam edebilir ve teslim edilmemiş yayınları alabilirsiniz.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback`' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle
MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı
İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını
Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı
Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar
Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma
Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri

Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

MQTT istemcilerinde konu dizgileri ve konu süzgeçleri

Konu dizgileri ve konu süzgeçleri, yayınlamak ve abone olmak için kullanılır. MQTT istemcilerindeki konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQiçindeki konu dizgileriyle aynıdır.

Konu dizeleri, abonelere yayın göndermek için kullanılır. `MqttClient.getTopic(java.lang.String topicString)` yöntemini kullanarak bir konu dizgisi oluşturun.

Konu süzgeçleri, konulara abone olmak ve yayınları almak için kullanılır. Konu süzgeçleri genel arama karakterleri içerebilir. Joker karakterlerle birden çok konuya abone olabilirsiniz. Bir abonelik yöntemini kullanarak bir konu süzgeci oluşturun; örneğin, `MqttClient.subscribe(java.lang.String topicFilter)`.

Konu dizgileri

IBM MQ konu dizgisinin sözdizimi Konu Dizgileriiçinde açıklanmıştır. MQTT konu dizgilerinin sözdizimi, MQTT client for Javaile ilgili API belgelerinde `MqttClient` sınıfında açıklanır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu.

Konu dizgisinin her tipinin sözdizimi hemen hemen aynıdır. Dört küçük fark vardır:

1. MQTT istemcileri tarafından IBM MQ ' e gönderilen konu dizgileri, kuyruk yöneticisi adlarına ilişkin kuralı izlemelidir.
2. Uzunluk üst sınırı farklıdır. IBM MQ konu dizgileri 10.240 karakterle sınırlıdır. Bir MQTT istemcisi 65535 bayta kadar konu dizgileri yaratabilir.
3. MQTT istemcisi tarafından yaratılan bir konu dizgisi boş karakter içeremez.
4. IBM Integration Busiçinde boş bir konu düzeyi, `'...//...'` geçersiz. Boş konu düzeyleri IBM MQtarafından desteklenir.

IBM MQ yayınlama/abone olma seçeneğinden farklı olarak, `mqttv3` iletişim kuralı bir yönetim konusu nesnesi kavramına sahip değildir. Bir konu nesnesinden ve konu dizgisinden konu dizgisi oluşturamazsınız. Ancak, bir konu dizgisi IBM MQiçindeki bir yönetim konusuyla eşlenir. Yönetim konusuyla ilişkilendirilen erişim denetimi, bir yayının konuya yayınlanacağını mı, yoksa atılacağını mı belirler. Abonelere iletildiğinde bir yayına uygulanan öznitelikler, yönetim konusunun özniteliklerinden etkilenir.

Konu süzgeçleri

IBM MQ konu süzgecinin sözdizimi Konu tabanlı joker karakter şemasıiçinde açıklanmıştır. MQTT istemcisiyle oluşturabileceğiniz konu süzgeçlerinin sözdizimi, MQTT client for Javaiçin API belgelerinde `MqttClient` sınıfında açıklanır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. MQTT istemci programlama başvurusu.

İlgili kavramlar

MQTT istemci uygulamalarında geri çağırmalar ve eşitleme
MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, sunucuya ve sunucudan ileti iletme gecikmelerinden olabildiğince ayırabilir. Yayınlar, teslim belirteçleri ve bağlantı kaybı olayları, `MqttCallback` ' i uygulayan bir geri çağırma sınıfındaki yöntemlere teslim edilir.

Oturumları temizle
MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. Durum bilgileri, yayınların "en az bir kez" ve "tam olarak bir kez" teslim edilmesini ve "tam olarak bir kez" alınmasını sağlamak için kullanılır. Oturum durumu, MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Bir MQTT istemcisini oturumlar arasında durum bilgilerini koruyarak ya da korumadan çalıştırmayı seçebilirsiniz. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını tanımlayarak temiz oturum kipini değiştirin.

İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanıtan 23 baytlık bir dizedir. Her tanıtıcı, bir kerede tek bir bağlı istemci için benzersiz olmalıdır. Tanıtıcı yalnızca kuyruk yöneticisi adında geçerli karakterler içermelidir. Bu kısıtlar içinde, herhangi bir tanıtıcı dizgisini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordamın ve seçilen tanıtıcıya sahip bir istemcinin yapılandırılmasına ilişkin bir yöntemin olması önemlidir.

Teslim belirteçleri

Son vasiyet ve vasiyet yayını
Bir MQTT istemci bağlantısı beklenmedik bir şekilde sona ererse, MQ Telemetry ' u "son vasiyet ve vasiyet" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderileceği konuyu önceden tanımlayın. "Son vasiyet ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce yaratın.

MQTT istemcilerinde ileti kalıcılığı
Yayın iletileri "en az bir kez"ya da "tam olarak bir kez"hizmet kalitesiyle gönderilirse kalıcı olarak yapılır. İstemcide kendi kalıcılık düzeneğini uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. İstemciye gönderilen ya da istemciden gönderilen yayınlar için kalıcılık her iki yönde de çalışır.

Yayınlar
Yayınlar, bir konu dizgisiyle ilişkilendirilmiş `MqttMessage` yönetim ortamlarıdır. MQTT istemcileri, IBM MQ' e göndermek üzere yayınlar oluşturabilir ve yayınları almak için IBM MQ ile ilgili konulara abone olabilirler.

Bir MQTT istemcisi tarafından sağlanan hizmet nitelikleri
Bir MQTT istemcisi, yayınların IBM MQ ' e ve MQTT istemcisine teslim edilmesi için üç hizmet niteliği sağlar: "en çok bir kez", "en az bir kez" ve "tam olarak bir kez". Bir MQTT istemcisi, abonelik oluşturmak için IBM MQ ' e bir istek gönderdiğinde, istek "en az bir kez" hizmet kalitesiyle gönderilir.

Yayınları ve MQTT istemcilerini alıkoyma
Bir konu, bir ve yalnızca bir korunan yayına sahip olabilir. Alıkonan bir yayını olan bir konuya abonelik oluşturursanız, yayın hemen size iletilir.

Abonelikler
Bir konu süzgecini kullanarak yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci, birden çok konuya ilgi göstermek için birden çok abonelik ya da joker karakter kullanan bir konu süzgeci içeren bir abonelik yaratabilir. Süzgeçlerle eşleşen konulara ilişkin yayınlar istemciye gönderilir. Bir istemcinin bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar, yeniden bağlandığında istemciye gönderilir.

IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi

IBM MQ için Microsoft Windows Communication Foundation (WCF) özel kanalı, WCF istemcileri ve hizmetleri arasında ileti gönderir ve alır.

İlgili kavramlar

“.NET ile WCF için IBM MQ özel kanalına giriş” sayfa 1211
IBM MQ özel kanalı, Microsoft Windows Communication Foundation (WCF) birleşik programlama modelini kullanan bir aktarım kanalıdır.

“WCF için IBM MQ özel kanallarının kullanılması” sayfa 1215
Windows Communication Foundation (WCF) için IBM MQ özel kanallarını kullanan programcılar için sağlanan bilgilere genel bakış.

“WCF örneklerinin kullanılması” sayfa 1234
Windows Communication Foundation (WCF) örnekleri, IBM MQ özel kanalının nasıl kullanılabileceğine ilişkin bazı basit örnekler sağlar.

FFST: WCF XMS İlk Hata Destek Teknolojisi

İlgili görevler

IBM MQ için WCF özel kanalının izlenmesi

IBM MQ sorunları için WCF özel kanalında sorun giderme

.NET ile WCF için IBM MQ özel kanalına giriş

IBM MQ özel kanalı, Microsoft Windows Communication Foundation (WCF) birleşik programlama modelini kullanan bir aktarım kanalıdır.

Microsoft.NET 3 'te tanıtılan Microsoft Windows Communication Foundation çerçevesi, .NET uygulamalarının ve hizmetlerinin, bunları bağlamak için kullanılan aktarım ve iletişim kurallarından bağımsız olarak geliştirilmesini sağlayarak, hizmet ya da uygulamanın konuşlandırıldığı ortama göre alternatif iletimlerin ya da yapılandırmaların kullanılmasını sağlar.

Bağlantılar çalıştırma zamanında WCF tarafından, gerekli birleşimleri içeren bir kanal yığını oluşturularak yönetilir:

- Protokol öğeleri: WS-* standartları gibi protokolleri desteklemek için hiçbir, bir ya da daha fazla öğe eklenemeyeceğini belirten isteğe bağlı bir öğe kümesi.
- İleti kodlayıcı: İletinin iletme biçiminde diziselleştirilmesini denetleyen yığın içindeki zorunlu bir öğe.
- İletim kanalı: Diziselleştirilmiş iletinin uç noktasına taşınmasından sorumlu yığın içindeki zorunlu bir öğe.

IBM MQ için özel kanal bir iletim kanalıdır ve bu nedenle, WCF özel bağ tanımı kullanan uygulamanın gerektirdiği şekilde, bir ileti kodlayıcı ve isteğe bağlı protokollerle eşleştirilmelidir. Bu şekilde, WCF 'yi kullanmak üzere geliştirilen uygulamalar, verileri Microsofttarafından sağlanan yerleşik iletimleri kullandıkları şekilde göndermek ve almak için IBM MQ özel kanalını kullanabilir ve IBM MQ' in zamanuyumsuz, ölçeklenebilir ve güvenilir ileti sistemi işlevleriyle basit bütünleştirme sağlar. Desteklenen işlevlerin tam listesi için bkz. “WCF Özel kanal özellikleri ve yetenekleri” sayfa 1215.

WCF için IBM MQ özel kanalını ne zaman ve neden kullanıyorum?

WCF istemcileri ve hizmetleri arasında, Microsofttarafından sağlanan yerleşik iletilerle aynı şekilde ileti göndermek ve almak için IBM MQ özel kanalını kullanabilirsiniz; böylece, uygulamalar WCF birleşik programlama modeli içindeki IBM MQ özelliklerine erişebilir.

WCF için IBM MQ özel kanalına ilişkin tipik bir kullanım örüntüsü senaryoları, yerel IBM MQ iletilerinin iletilmesi için SOAP dışı bir arabirim olarak kullanılır.

JMS dışı ileti (Pure MQMessage) biçimi kullanılarak taşınan iletiler

Yerel IBM MQ iletilerinin iletimi için SOAP olmayan bir arabirim olarak WCF için IBM MQ özel kanalını kullandığınızda, iletiler IBM MQöğesinin Non-SOAP/Non-JMS (Pure MQMessage) biçimi kullanılarak taşınır.

WCF kullanıcıları hizmeti başlatabilir ya da başka bir deyişle, hizmet kullanıcıları MQMessages kullanarak bir IBM MQ kuyruğuna ileti gönderebilirler. Uygulamalar MQMD alanlarını ve bilgi yükünü alabilir ve ayarlayabilir. İleti IBM MQ kuyruklarında varsa, bu ileti AIX, Linux, Windowsya da z/OSüzerinde çalışan C ya da Java uygulamaları gibi WCF olmayan uygulamalar tarafından işlenebilir.

WCF için IBM MQ özel kanalına ilişkin yazılım gereksinmeleri

Bu konuda, WCF için IBM MQ özel kanalına ilişkin yazılım gereksinimleri açıklanmaktadır. WCF için IBM MQ özel kanalı yalnızca IBM WebSphere MQ 7.0 ya da daha yüksek kuyruk yöneticilerine bağlanabilir.

Çalıştırma zamanı ortamı gereksinimleri

- Microsoft.NET Framework v4.7.2 ya da üstü anasistem makinesine kurulmalıdır.
- *Java ve .NET Messaging and Web Services* varsayılan olarak IBM MQ kuruluş programının bir parçası olarak kurulur. Bu bileşen, özel kanal için gereken .NET düzeneklerini Genel Montaj Önbelleğine kurar.

Not: IBM MQkurulmadan önce Microsoft .NET Framework V4.7.2 ya da üstü kurulmazsa, IBM MQ ürün kuruluşu hatasız devam eder, ancak IBM MQ classes for .NET kullanılamaz. .NET Framework , IBM MQkurulduktan sonra kuruluysa, IBM MQ.NET yapıbirimleri *WMQInstallDir*\bin\amqiRegisterdotNet.cmd komut dosyası çalıştırılarak kaydedilmelidir; burada

WMQInstallDir , IBM MQ programının kurulu olduğu dizindir. Bu komut dosyası, Global Assembly Cache (GAC) içindeki gerekli düzenekleri kurar. Yapılan işlemleri kaydeden bir `amqi*.log` dosyaları kümesi, %TEMP% dizininde oluşturulur. .NET , daha önceki bir sürümden (örneğin, .NET V3.5) V4.7.2 ya da daha sonraki bir sürüme yükseltildiyse, `amqiRegisterdotNet.cmd` komut dosyasını yeniden çalıştırmanız gerekmez.

Geliştirme ortamı gereksinimleri

- Microsoft Visual Studio 2015 ya da Windows Software Development Kit for .NET 4.7.2 ya da üstü.
- Örnek çözüm dosyalarını oluşturmak için anasistem makinesine Microsoft.NET Framework V4.7.2 ya da üstü kurulmalıdır.

WCF için IBM MQ özel kanal: Kurulu olan nedir?

IBM MQ özel kanalı, Microsoft Windows Communication Foundation (WCF) birleşik programlama modelini kullanan bir aktarım kanalıdır. Özel kanal, kuruluşun bir parçası olarak varsayılan olarak kurulur.

WCF için IBM MQ özel kanalı

Özel kanal ve bağımlılıkları, varsayılan olarak kurulan `Java and .NET Messaging and Web Services` bileşeninde bulunur. IBM MQ ürününüIBM MQ 8.0sürümünden daha önceki bir sürümden yükseltirken, `Java and .NET Messaging and Web Services` bileşeni daha önceki bir kuruluşta kurulduysa, güncelleme varsayılan olarak WCF için IBM MQ özel kanalını kurar.

`.NET Messaging and Web Services` bileşeni `IBM.XMS.WCF.dll` dosyasını ve `IBM.WMQ.WCF.dll` dosyasını içerir ve bu dosyalar, WCF arabirim sınıflarını içeren ana özel kanal düzeneğidir. Bu dosyalar Global Assembly Cache (GAC) içine kurulur ve şu dizinde de bulunur: *MQ_INSTALLATION_PATH* `\bin` ; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizindir.

Aşağıdaki çizelge, özel kanalı kullanmak için gereken anahtar sınıflarını özetler.

Çizelge 189. Özel kanalı kullanmak için gerekli anahtar sınıfları

	SOAP/JMS arabirimi (Var olan)	**SOAP/Non-JMS arabirimi (IBM MQ 8.0 içinden)**
Özel Kanal Düzeneği	IBM.XMS.WCF.dll	IBM.WMQ.WCF.dll
İletim Bağ Tanımı Adı	IBM.XMS.WCF.SoapJmsIbmTransportBindingElement	IBM.WMQ.WCF.WmqIbmTransportBindingElement
İletim Bağ Tanımı İçe Aktarıcısı	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementImporter	IBM.WMQ.WCF.WmqIbmTransportBindingElementImporter
İletim Bağ Tanımı Yapılanışı	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig	IBM.WMQ.WCF.WmqIbmTransportBindingElementConfig
Örnekler (Tek yönlü)	SimpleOneWay_Client, SimpleOneWay_Service	MQMessaging_OneWay_Client, MQMessaging_OneWay_Service
Örnekler (RequestReply)	SimpleRequestReply_Client, SimpleRequestReply_Service	MQMessaging_RequestReply_Client, MQMessaging_RequestReply_Service

IBM.WMQ.WCF.dll , SOAP/JMS ve Non-SOAP/Non-JMS arabirimlerini destekler. Geliştirilen yeni uygulamaların IBM.WMQ.WCF düzeneği.

MQSTR biçimli iletiler gönderiliyor

İstek iletisi MQSTR tipindeyse, yanıt iletisini MQSTR biçiminde göndermeyi seçebilirsiniz.

Yanıt iletisinin biçimini değiştirmek için ek bir URI parametresi **`replyMessageFormat`** kullanmalısınız. Desteklenen değerler şunlardır:

""

"" varsayılan değerdir.

Yanıt iletisi byte (MQMFT_NONE) biçiminde. Örneğin:

```
"jms:/queue?
destination=SampleQ@QM1&connectionFactory=binding(server)connectQueueManager(QM1)
&initialContextFactory=com.ibm.mq.jms.Nojndi&replyDestination=SampleReplyQ&replyMessageForma
t= "
```

MQSTR

Yanıt iletisi MQSTR (MQMFT_STRING) biçiminde. Örneğin:

```
"jms:/queue?
destination=SampleQ@QM1&connectionFactory=binding(server)connectQueueManager(QM1)
&initialContextFactory=com.ibm.mq.jms.Nojndi&replyDestination=SampleReplyQ&replyMessageForma
t=MQSTR"
```

Notlar:

1. **`replyMessageFormat`** değeri büyük ve küçük harfe duyarlı değildir.
2. "" ya da *MQSTR*dışında bir değer kullanılması geçersiz bir değiştirge değeri kural dışı durumuna neden olur.

IBM MQ özel kanal örnekleri

Örnekler, WCF için IBM MQ özel kanalının nasıl kullanılabileceğine ilişkin bazı basit örnekler sağlar. Örnekler ve ilişkili dosyaları *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf` dizininde bulunur; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir. IBM MQ özel kanal örnekleri hakkında daha fazla bilgi için bkz. "WCF örneklerinin kullanılması" sayfa 1234.

svcutil.exe.config

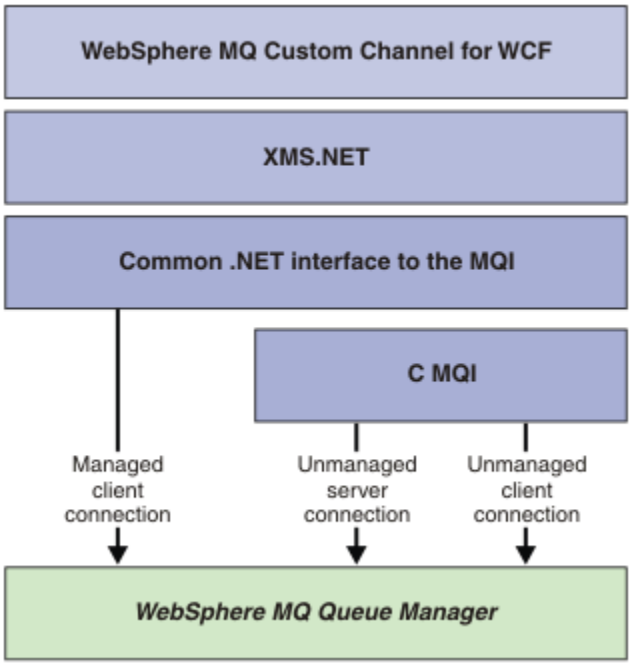
`svcutil.exe.config` , Microsoft WCF `svcutil` istemcisi yetkili sunucu oluşturma aracının özel kanalı tanımasını sağlamak için gereken yapılandırma ayarlarına bir örnektir. `svcutil.exe.config` dosyası, *MQ_INSTALLATION_PATH* `\tools\wcf\docs\examples\` dizininde bulunur; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir. `svcutil.exe.config`kullanımı hakkında daha fazla bilgi için bkz. "Çalışmakta olan bir hizmetten meta verilerle svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılanış dosyaları oluşturulması" sayfa 1231.

WCF mimarisi

WCF için IBM MQ özel kanalı, `IBM Message Service Client for .NET (XMS .NET)` API'nın üstünde bütünleştirilmiştir.

SOAP/JMS arabirimi

WCF mimarisi aşağıdaki şemada gösterilmiştir:



Şekil 149. SOAP/JMS arabirimi için WCF mimarisi

Gerekli tüm bileşenler varsayılan olarak ürün kuruluşuyla birlikte kurulur.

Üç bağlantı şunlardır:

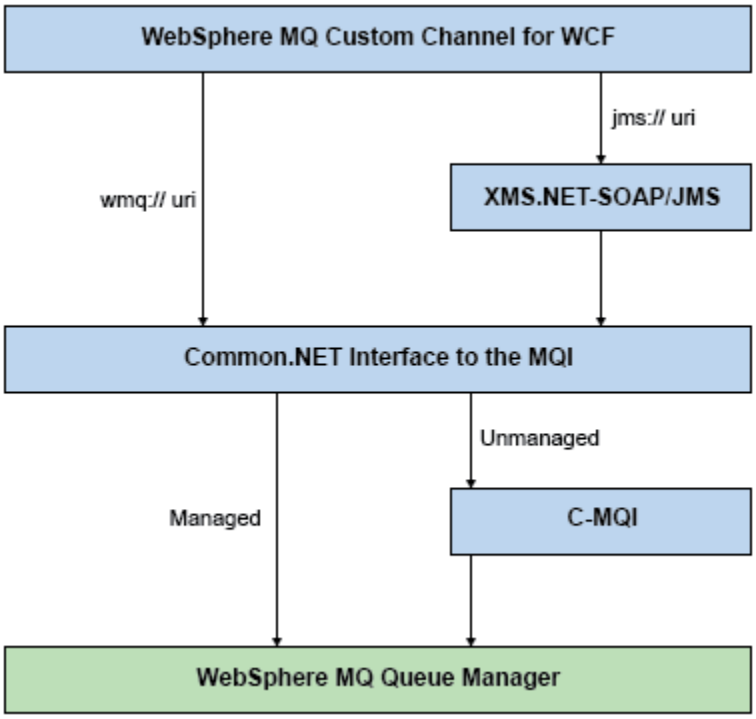
- Yönetilen istemci bağlantıları
- Yönetilmeyen sunucu bağlantıları
- Yönetilmeyen istemci bağlantıları

Bu bağlantılar hakkında daha fazla bilgi için bkz. “WCF Bağlantısı seçenekleri” sayfa 1221.

SOAP/Non-JMS arabirimi

WCF için IBM MQ özel kanalı hem SOAP/JMS arabirimini destekler (IBM WebSphere MQ 7.0.1 adresinden edinilebilir) ve Non-SOAP/Non-JMS arabirimi.

WCF mimarisi aşağıdaki şemada gösterilmiştir:



Şekil 150. SOAP/Non-JMS arabirimi için WCF mimarisi

WCF için IBM MQ özel kanallarının kullanılması

Windows Communication Foundation (WCF) için IBM MQ özel kanallarını kullanan programcılar için sağlanan bilgilere genel bakış.

Microsoft Windows Communication Foundation, Microsoft.NET Framework 3 'teki web hizmetlerini ve ileti sistemi desteğini destekler. IBM MQ , Microsofttarafından sunulan yerleşik kanallarla aynı şekilde .NET Framework 3 'te WCF içinde özel bir kanal olarak kullanılabilir.

Özel kanal üzerinden iletilen iletiler, IBM MQ' un JMS üzerinden SOAP uygulamasına göre biçimlendirilir. Uygulamalar daha sonra WCF tarafından barındırılan hizmetlerle ya da WebSphere SOAP over JMS hizmet altyapısıyla iletişim kurabilir.

WCF Özel kanal özellikleri ve yetenekleri

WCF özel kanal özellikleri ve yetenekleriyle ilgili bilgi için aşağıdaki konuları kullanın.

WCF özel kanal şekilleri

IBM MQ ' in Microsoft Windows Communication Foundation (WCF) özel kanalları içinde kullanılabileceği özel kanal şekillerine genel bakış.

WCF için IBM MQ özel kanalı iki kanal şekillerini destekler:

- Tek Yönlü
- İstek-yanıt

WCF, barındırılmakta olan hizmet sözleşmesine göre kanal şeklini otomatik olarak seçer.

Yalnızca **IsOneWay** parametresini kullanan yöntemleri içeren sözleşmelere tek yönlü kanal şekli hizmet vermektedir; örneğin:

```
[OperationContract(IsOneWay = true)]
void printString(String text);
```

Tek yönlü ve istek-yanıt yöntemlerinin ya da tüm istek-yanıt yöntemlerinin karışımını içeren sözleşmeler, istek-yanıt kanalı şekliyle servis edilir. Örneğin:

```
[OperationContract]
int subtract(int a, int b);

[OperationContract(IsOneWay = true)]
void printString(string text);
```

Not: Tek yönlü ve istek-yanıt yöntemlerini aynı sözleşmede karıştırırken, tek yönlü yöntemler hizmetten boş bir yanıt alıncaya kadar beklediğinden, davranışın özellikle karışık bir ortamda çalışırken amaçlandığı gibi olduğundan emin olmanız gerekir.

Tek yönlü kanal

WCF için IBM MQ tek yönlü özel kanalı kullanılır; örneğin, tek yönlü kanal şeklini kullanarak bir WCF istemcisinden ileti göndermek için. Kanal, iletileri yalnızca bir yönde gönderebilir; örneğin, bir istemci kuyruk yöneticisinden WCF hizmetindeki bir kuyruğa.

İstek-yanıt kanalı

WCF için IBM MQ istek yanıtı özel kanalı, örneğin, iletileri zamanuyumsuz olarak iki yönde göndermek için kullanılır; Zamanuyumsuz ileti alışverişi için aynı istemci yönetim ortamı kullanılmalıdır. Kanal, iletileri bir yönde (örneğin, bir istemci kuyruk yöneticisinden WCF hizmetindeki bir kuyruğa) gönderebilir ve daha sonra, WCF ' den istemci kuyruk yöneticisindeki bir kuyruğa yanıt iletisi gönderebilir.

WCF URI değiştirge adları ve değerleri

SOAP/JMS arabirimi ve Non-SOAP/Non JMS arabirimi için URI parametre adları ve değerleri.

SOAP/JMS arabirimi

connectionFactory
connectionFactory değiştirgesi gereklidir.

initialContextFactory
initialContextFactory parametresi gereklidir ve WebSphere Application Server ve diğer ürünlerle uyumluluk için "com.ibm.mq.jms.Nojndi" olarak ayarlanmalıdır.

SOAP/Non JMS arabirimi

URI biçimi, MA93 belirtimlerine uygundur. IBM MQ IRI belirtimlerine ilişkin daha fazla ayrıntı için bkz. SupportPac - MA93 .

IBM MQ URI sözdizimi

```
wmq-iri = "wmq:" [ "//" connection-name ] "/" wmq-dest ["?" parm *("&" parm)]
connection-name = tcp-connection-name / other-connection-name
tcp-connection-name = ihost [ ":" port ]
other-connection-name = 1*(iunreserved / pct-encoded)
wmq-dest = queue-dest / topic-dest
queue-dest = "msg/queue/" wmq-queue ["@" wmq-qmgr]
wmq-queue = wmq-name
wmq-qmgr = wmq-name
wmq-name = 1*48( wmq-char )
topic-dest = "msg/topic/" wmq-topic
wmq-topic = segment *( "/" segment )
```

IBM MQ IRI örneği
Aşağıdaki örnek IRI, bir hizmet isteğine 1414 numaralı kapıda example.com adlı bir makineye IBM MQ TCP istemci bağlama bağlantısı kullanabileceğini ve QM1kuyruk yöneticisinde SampleQ adlı bir

kuyruğa kalıcı istek iletileri yerleştirebileceğini bildirir. IIRI, hizmet sağlayıcısının SampleReplyQ adlı bir kuyruğa yanıt koyacağını belirtir.

```
1)wmq://example.com:1414/msg/queue/SampleQ@QM1?
ReplyTo=SampleReplyQ&persistence=MQPER_NOT_PERSISTENT
2)wmq://localhost:1414/msg/queue/Q1?
connectQueueManager=QM1&replyTo=Q2&connectionmode=managed
```

TLS etkin bağlantılar için

WCF İstemcisi/Hizmeti 'ni kullanarak Güvenli (TLS) bağlantılar yapmak için URI 'de uygun değerlerle aşağıdaki özellikleri ayarlayın. Güvenli bir bağlantı kurmak için öneki "*" olan tüm özellikler zorunludur.

- **sslKeyRepository**: *SYSTEM ya da *USER
- * **sslCipherSpec**: geçerli bir CipherSpec; örneğin, TLS_RSA_WITH_AES_128_CBC_SHA256.
- **sslCertRevocationCheck**: true ya da false.
- **sslKeyResetCount**: 32kb' den büyük bir değer.
- **sslPeerName:** sunucu sertifikasının ayırt edici adı

Örneğin:

```
"wmq://localhost:1414/msg/queue/SampleQ?
connectQueueManager=QM1&sslkeyrepository=*SYSTEM&sslcipherspec=
TLS_RSA_WITH_AES_128_CBC_SHA&sslcertrevocationcheck=true&"sslpe
ername=" + "" + "CN=ibmwebspheremqqmm&sslkeyresetcount=45000"
```

WCF özel kanal güvenli teslim

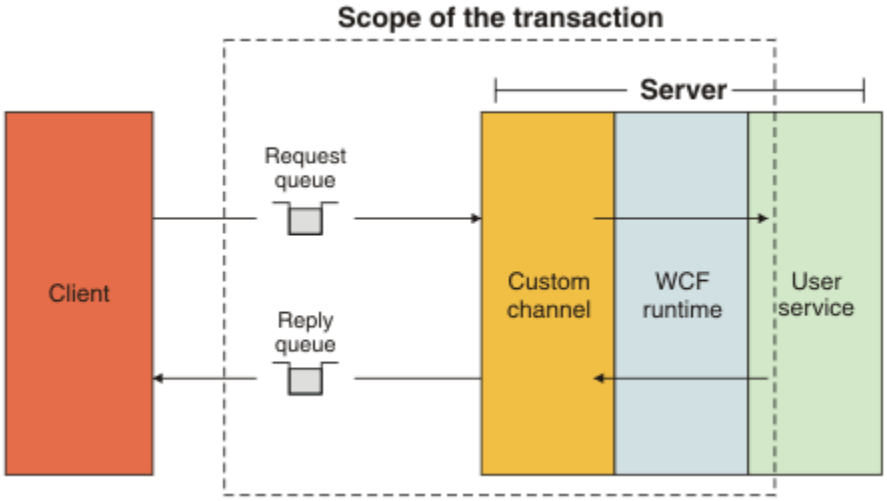
Garantili Teslim, bir hizmet isteğinin ya da yanıtın işlem görmekte olduğunu ve kaybolmadığını garanti eder.

Bir istek iletisi alınır ve herhangi bir yanıt iletisi, yürütme zamanı hatası durumunda geriye işlenebilecek yerel bir hareket eşitleme noktası altında gönderilir. Bu hatalara örnek olarak şunlar verilebilir: Hizmet tarafından verilen işlenmemiş bir kural dışı durum, iletiyi hizmete gönderememe ya da yanıt iletisi teslim edilemedi.

`AssuredDelivery` , bir hizmet sözleşmesinde belirtilen ve bir hizmet tarafından alınan herhangi bir istek iletisinin ve bir hizmetten gönderilen herhangi bir yanıt iletisinin, yürütme sırasında hata oluşması durumunda kaybolmamasını garanti eden güvenli teslim özniteliğidir.

Sistem arızası ya da güç kesintisi durumunda iletilerin de korunduğundan emin olmak için iletilerin kalıcı olarak gönderilmesi gerekir. Kalıcı iletileri kullanmak için istemci uygulamasının uç noktası URI 'sinde bu seçenek belirtilmiş olmalıdır.

Dağıtılmış hareketler desteklenmez ve işlemin kapsamı, IBM MQtarafından gerçekleştirilen istek ve yanıt iletilerinin işlenmesini aşmaz. Hizmet içinde gerçekleştirilen herhangi bir iş, iletinin yeniden alınmasına neden olan bir hatanın sonucu olarak yeniden çalıştırılabilir. Aşağıdaki çizge hareketin kapsamını gösterir:



Aşağıdaki örnekte gösterildiği gibi hizmet sınıfına `AssuredDelivery` özniteliği uygulanarak güvenli teslim etkinleştirilir:

```
[AssuredDelivery]
class TestCalculatorService : IWMQSampleCalculatorContract
{
    public int add(int a, int b)
    {
        int ans = a + b;
        return ans;
    }
}
```

`AssuredDelivery` özniteliğini kullanırken aşağıdaki noktaları bilmeniz gerekir:

- Bir kanal, bir ileti geriye işlenip yeniden alındığında bir hatanın yeniden oluşma olasılığını belirlediğinde, ileti zehirli bir ileti olarak işlenir ve yeniden işlenmek üzere istek kuyruğuna döndürülmez. Örneğin: Alınan ileti doğru biçimlendirilmediyse ya da bir hizmete dağıtılamıyorsa. Bir hizmet işleminden verilen işlenmemiş kural dışı durumlar, ileti istek kuyruğunun geriletme eşiği özelliği tarafından belirlenen sayı üst sınırı kadar yeniden teslim edilinceye kadar her zaman yeniden gönderilir. Daha fazla bilgi için bkz: “WCF özel kanal zehirleme iletileri” sayfa 1219
- Kanal, hareket bütünlüğünü zorlamak için tek bir yürütme iş parçacığını kullanarak atomik bir işlem olarak her bir istek iletisinin okunmasını, işlenmesini ve yanıtlanmasını gerçekleştirir. Hizmet işlemlerinin eşzamanlı olarak çalışmasını sağlamak için kanal, WCF ' nin kanalın birden çok yönetim ortamını yaratmasını sağlar. İstekleri işlemek için kullanılabilecek kanal eşgörünümlerinin sayısı, `MaxConcurrentCalls`bağ tanımlama özelliği tarafından denetlenir. Daha fazla bilgi için bkz: “WCF bağ tanımı yapılanış seçenekleri” sayfa 1227
- Güvenli teslim işlevi hem IOperationInvoker hem de IErrorHandler WCF genişletilebilirlik noktalarını kullanır. Bu genişletilebilirlik noktaları bir uygulama tarafından dışarıdan kullanılıyorsa, uygulamanın önceden kayıtlı genişletilebilirlik noktalarının çağrıldığından emin olması gerekir. IErrorHandler için böyle bir hata yapılmaması, raporlanmayan hatalarla sonuçlanabilir. IOperationInvoker için bunu yapmamanız, WCF ' nin yanıt vermeyi durdurmasına neden olabilir.

WCF özel kanal güvenliği

WCF için IBM MQ özel kanalı, TLS ' nin yalnızca kuyruk yöneticisine yönelik yönetilmeyen istemci bağlantıları için kullanılmasını destekler.

İstemci kanal tanımlama çizelgesinde (CCDT) bir giriş kullanarak TLS belirtin. CCDT ' ler hakkında daha fazla bilgi için bkz. İstemci kanal tanımlama çizelgesi.

WCF istemci kanal tanımlama çizelgeleri (CCDT)

WCF için IBM MQ özel kanalı, istemci bağlantıları için bağlantı bilgilerini yapılandırmak üzere istemci kanal tanımlama çizelgelerinin (CCDT) kullanılmasını destekler.

CCDT ' ler şu iki ortam değişkeniyle denetlenir:

- *MQCHLLIB* çizelgenin bulunduğu dizini belirtir.
- *MQCHLTAB* çizelgenin dosya adını belirtir.

Bu ortam değişkenleri tanımlandıysa, bunlar URI ' de belirtilen istemci bağlantısı ayrıntılarına göre önceliklidir.

İstemci kanal tanımlama çizelgeleriyle ilgili daha fazla bilgi için bakınız: Client channel definition table.

WCF özel kanal zehirleme iletileri

Bir hizmet bir istek iletisini işleyemezse ya da yanıt kuyruğuna yanıt iletisi teslim edilemezse, ileti zehirli bir ileti olarak işlenir.

Zehirli istek iletileri

Bir istek iletisi işlenemiyorsa, bu ileti zehirli bir ileti olarak işlenir. Bu işlem, hizmetin aynı işlenemez iletiyi yeniden almasını engeller. İşlenemeyen bir istek iletisinin zehirli bir ileti olarak ele alınabilmesi için aşağıdaki durumlardan birinin doğru olması gerekir:

- İleti geriletme sayısı, istek kuyruğunda belirtilen geriletme eşiğini aştı; bu değer, yalnızca hizmet için güvenli teslim belirtildiyse ortaya çıkar. Güvenli teslimat hakkında daha fazla bilgi için bkz: “WCF özel kanal güvenli teslim” sayfa 1217
- İleti doğru biçimlendirilmedi ve JMS üzerinden SOAP iletisi olarak yorumlanamadı.

Zehirli yanıt iletileri

Bir hizmet yanıt kuyruğuna yanıt iletisi gönderemezse, yanıt iletisi zehirli bir ileti olarak işlenir. Yanıt iletileri için bu işlem, sorunun saptanmasına yardımcı olmak üzere yanıt iletilerinin daha sonra alınmasını sağlar.

Zehirli ileti işleme

Bir zehirli ileti için yapılan işlem, kuyruk yöneticisi yapılandırmasına ve iletinin rapor seçeneklerinde ayarlanan değerlere bağlıdır. JMSüzerinden SOAP için, varsayılan olarak istek iletilerinde aşağıdaki rapor seçenekleri ayarlanır ve yapılandırılamaz:

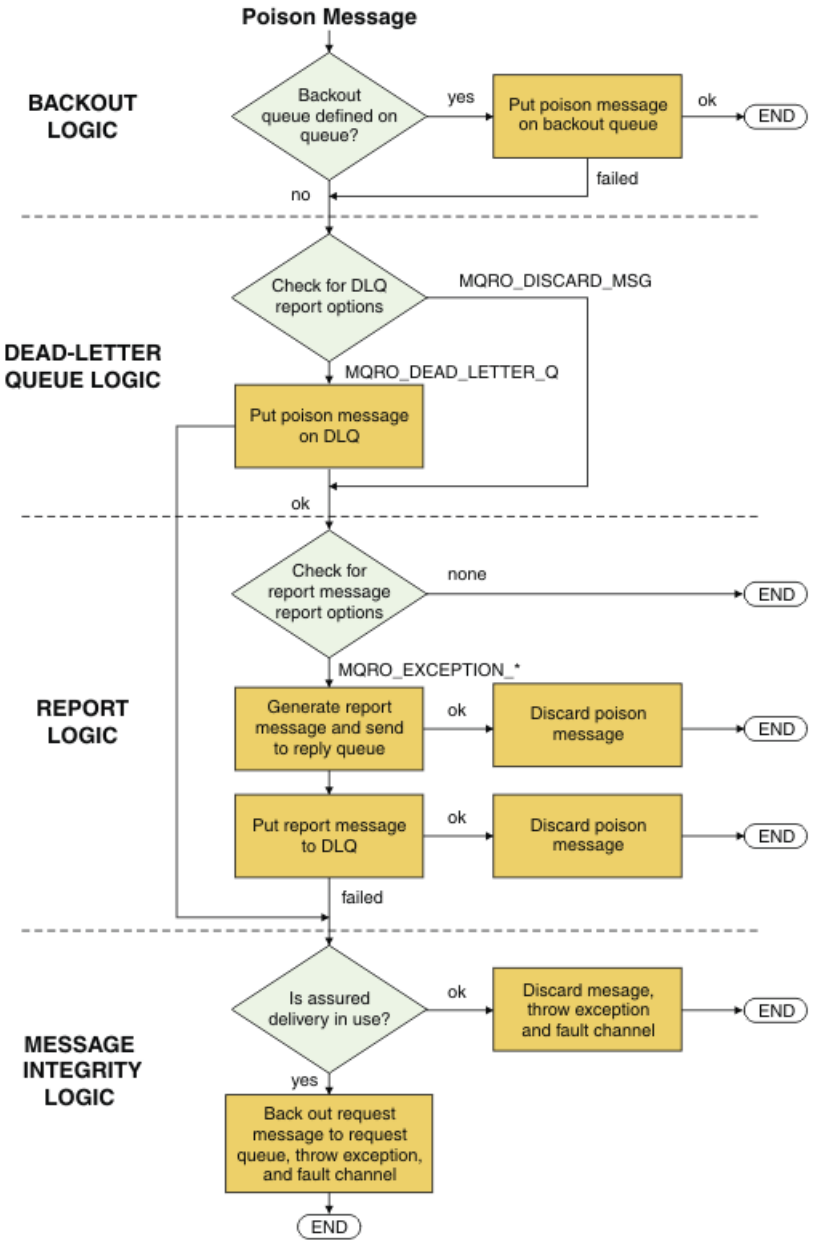
- MQRO_EXCEPTION_WITH_FULL_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_DISCARD_MSG

JMSüzerinden SOAP için aşağıdaki rapor seçeneği varsayılan olarak yanıt iletilerinde ayarlanır ve yapılandırılamaz:

- MQRO_DEAD_LETTER_Q

İletiler WCF olmayan bir kaynaktan geliyorsa, o kaynağa ilişkin belgelere bakın.

Aşağıdaki şekil, olası işlemleri ve zehirli ileti işleme başarısız olursa gerçekleştirilecek adımları göstermektedir:



WCF uygulamaları için IBM MQ ileti yetenekleri

SOAP/Non-JMS (yani, IBM MQ) WCF uygulamaları için ileti yetenekleri.

Non-SOAP/Non-JMS arabirimi için, WCF uygulamalarına ilişkin IBM MQ ileti yetenekleri aşağıdaki gibidir:

- WCF uygulamaları, herhangi bir IBM MQ uygulaması tarafından işlenebilecek temel IBM MQ iletilerini gönderebilir ve alabilir.
- WCF uygulamaları MQMD ' yi ve bilgi yükünü güncellemek için tam denetime sahip.
- WCF istemcisi, herhangi bir IBM MQ istemcisi tarafından kullanılabilecek IBM MQ iletilerini (örneğin, C, Java, JMSve .NET istemcileri) gönderebilir.

WCF for Non-SOAP/Non-JMS arabirimi, ileti bilgi yükünü ve MQMD ' yi ayarlamak için aşağıdaki sınıfları kullanmalıdır:

- WmqStringString tipindeki bir bilgi yükü için ileti
- WmqBytesBytes tipindeki bir bilgi yükü için ileti
- WmqXmlXML tipinde bir bilgi yükü için ileti

İletinin bilgi yükünü ayarlamak için, bilgi yükü tipine bağlı olarak WmqStringİletisi, WmqBytesİletisi ya da WmqXmlİleti sınıfı için **Data** özelliğini kullanın. Örneğin, Dizgi tipinde bir bilgi yükü ayarlamak için aşağıdaki kodu kullanın:

```
WmqStringMessage strMsg = new WmqStringMessage();
//Setting the Message PayLoad
strMsg.Data = "Hello World";
//MQMD property
strMsg.Format = WmqMessageFormat.MQFMT_STRING;
```

WCF Bağlantısı seçenekleri

WCF için IBM MQ özel kanalını bir kuyruk yöneticisine bağlamak için üç kip vardır. Gereksinimlerinize en uygun bağlantı tipini göz önünde bulundurun.

Bağlantı seçenekleri hakkında daha fazla bilgi için bkz: “Bağlantı farkları” sayfa 556

WCF mimarisi hakkında daha fazla bilgi için bkz: “WCF mimarisi” sayfa 1213

Yönetilmeyen istemci bağlantısı

Bu kipte yapılan bir bağlantı, IBM MQ istemcisi olarak yerel makinede ya da uzak makinede çalışan bir IBM MQ sunucusuna bağlanır.

WCF ' ye ilişkin IBM MQ özel kanalını IBM MQ istemcisi olarak kullanmak için, bu kanalı IBM MQ MQI clientile IBM MQ sunucusuna ya da ayrı bir makineye kurabilirsiniz.

Yönetilmeyen sunucu bağlantısı

Sunucu bağ tanımları kipinde kullanıldığında, WCF için IBM MQ özel kanalı, bir ağ üzerinden iletişim kurmak yerine kuyruk yöneticisi API 'sini kullanır. Bağ tanımı bağlantılarının kullanılması, IBM MQ uygulamaları için ağ bağlantılarını kullanmaktan daha iyi performans sağlar.

Bağ tanımları bağlantısını kullanmak için, IBM MQ sunucusuna WCF için IBM MQ özel kanalını kurmanız gerekir.

Yönetilen istemci bağlantısı

Bu kipte yapılan bir bağlantı, IBM MQ istemcisi olarak yerel makinede ya da uzak makinede çalışan bir IBM MQ sunucusuna bağlanır.

Bu kipte bağlanan .NET 3 için IBM MQ özel kanal sınıfları .NET yönetilen kodunda kalır ve yerel hizmetlere çağrı yapmaz. Yönetilen kodla ilgili daha fazla bilgi için Microsoft belgelerine bakın.

Yönetilen istemcinin kullanılmasına ilişkin bazı sınırlamalar vardır. Bu sınırlamalarla ilgili daha fazla bilgi için bkz. “Yönetilen istemci bağlantıları” sayfa 556.

WCF için IBM MQ özel kanalının oluşturulması ve yapılandırılması

WCF için IBM MQ özel kanalları, Microsofttarafından sunulan aktarım WCF kanallarıyla aynı şekilde çalışır. WCF için IBM MQ özel kanalı iki yoldan biriyle yaratılabilir.

Bu görev hakkında

IBM MQ özel kanalı, WCF aktarım kanalı olarak WCF ile bütünleşir ve bu nedenle, bir uygulama tarafından kullanılabilecek tam bir kanal yığını yaratabilmesi için bir ileti kodlayıcı ve isteğe bağlı protokol kanallarıyla eşleştirilmelidir. Tam bir kanal yığınının başarıyla yaratılması için iki öğe gereklidir:

1. Bağ tanımlama tanımlaması: İletim kanalı, ileti kodlayıcı ve tüm protokoller ve genel yapılandırma ayarları da içinde olmak üzere, uygulama kanalı yığınını oluşturmak için hangi öğelerin gerekli olduğunu belirtir. Özel kanal için, bağ tanımlama tanımlaması WCF özel bağ tanımı biçiminde yaratılmalıdır.

2. Uç nokta tanımlaması: Hizmet sözleşmesini bağ tanımlama tanımıyla bağlar ve uygulamanın nereye bağlanabileceğini açıklayan gerçek bağlantı URI 'sini de sağlar. Özel kanal için URI, JMS URI üzerinden SOAP biçimindedir.

Bu tanımlamalar iki farklı yoldan biriyle yaratılabilir:

- Yönetimsel olarak; Tanımlamalar, ayrıntılar bir uygulama yapılandırma dosyasında (örneğin: `app.config`) sağlanarak oluşturulur.
- Program açısından; Tanımlamalar doğrudan uygulama kodundan oluşturulur.

Tanımları yaratmak için hangi yöntemin kullanılacağı konusundaki karar, uygulamanın gereksinimlerine aşağıdaki şekilde dayalı olmalıdır:

- Yapılandırma için Yönetim yöntemi, uygulamayı yeniden oluşturmadan hizmet ve istemci devreye alma sonrası ayrıntılarını değiştirme esnekliği sağlar.
- Yapılandırma için Programmatik yöntem, yapılandırma hatalarına karşı daha fazla koruma ve çalıştırma zamanında dinamik olarak bir yapılandırma oluşturma yeteneği sağlar.

Bir uygulama yapılandırma dosyasında bağ tanımı ve uç noktası bilgileri sağlayarak yönetimsel olarak WCF özel kanalı oluşturma

WCF için IBM MQ özel kanalı, aktarım düzeyi bir WCF kanalıdır. Özel kanalı kullanmak için bir uç nokta ve bağ tanımı tanımlanmalıdır; bu tanımlar, bir uygulama yapılandırma dosyasında bağ tanımı ve uç nokta bilgileri sağlanarak gerçekleştirilebilir.

İletim düzeyi WCF kanalı olan WCF için IBM MQ özel kanalını yapılandırmak ve kullanmak için bir bağ tanımı ve bir uç nokta tanımlaması tanımlanmalıdır. Bağ tanımı, kanala ilişkin konfigürasyon bilgilerini içerir ve uç nokta tanımı bağlantı ayrıntılarını içerir. Bu tanımlamalar iki şekilde yaratılabilir:

- Burada açıklandığı gibi, programsal olarak doğrudan uygulama kodundan: “Bağ tanımlama ve uç nokta bilgilerini programlı olarak sağlayarak WCF özel kanalı yaratılması” sayfa 1224
- Denetimci olarak, aşağıdaki yordamda açıklandığı gibi, bir uygulama yapılandırma dosyasında ayrıntıları sağlayarak.

İstemci ya da hizmet uygulaması yapılanış kütüğü yaygın olarak *`yourappname`*`.exe.config` olarak adlandırılır; burada *uygulamanız* , uygulamanızın adıdır. Uygulama yapılandırma dosyası, `SvcConfigEditor.exe` adlı Microsoft hizmet yapılandırma düzenleyicisi aracı kullanılarak aşağıdaki şekilde kolayca değiştirilir:

- `SvcConfigEditor.exe` yapılandırma düzenleyicisi aracını başlatın. Aracın varsayılan kuruluş konumu şöyledir: *`Drive:`*`\Program Files\Microsoft SDKs\Windows\v6.0\Bin\SvcConfigEditor.exe` ; burada *Sürücü:* , kuruluş sürücüsünün adıdır.

Adım 1: WCF ' nin özel kanalı bulmasını sağlamak için bir bağ tanımlama öğesi uzantısı ekleyin

1. Menüyü açmak için **Gelişmiş** > **Uzantı** > **bağlama öğesi** seçeneğini sağ tıklatın ve **Yeni** seçeneğini belirleyin.
2. Alanları bu çizelgede gösterildiği gibi tamamlayın:

Çizelge 190. Yeni bağ tanımı öğesi alanları

Alan	**Değer**
NAME	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`
Tür	Global Assembly Cache (GAC) içindeki `IBM.XMS.WCF.dll` öğesine gidin ve `IBM.XMS.WCFSoapJmsIbmTransportBindingElementConfig` seçeneğini belirleyin.

Adım 2: Özel kanalı bir WCF ileti kodlayıcısıyla eşleyen özel bir bağ tanımı yaratılması

1. Menüyü açmak için **Bağ Tanımları** öğesini farenin sağ düğmesiyle tıklatın ve **Yeni Bağ Tanımı Yapılanışı** öğesini seçin.
2. Alanları bu çizelgede gösterildiği gibi tamamlayın:

Çizelge 191. Yeni bağ tanımlama yapılanışı alanları	
Alan	**Değer**
NAME	`CustomBinding_WMQ`
BindingElement 1	`textMessageEncoding (MessageVersion: Soap11)`
BindingElement 2	`IBM.XMS.WCF.SoapJmsIbmTransportChannel`

Adım 3: Bağ tanımlama özelliklerinin belirtilmesi

1. *IBM.XMS.WCF.SoapJmsIbmTransportChannel* aktarım bağ tanımı yarattığınız bağ tanımından: “Adım 2: Özel kanalı bir WCF ileti kodlayıcısıyla eşleyen özel bir bağ tanımı yaratılması” sayfa 1223
2. Özelliklerin varsayılan değerlerinde gerekli değişiklikleri yapın: “WCF bağ tanımı yapılanış seçenekleri” sayfa 1227

Adım 4: Uç nokta tanımlaması yaratılması

İçinde yarattığınız özel bağ tanımına gönderme yapan bir uç nokta tanımlaması yaratın: “Adım 2: Özel kanalı bir WCF ileti kodlayıcısıyla eşleyen özel bir bağ tanımı yaratılması” sayfa 1223 ve hizmetin bağlantı ayrıntılarını sağlar. Bu bilgilerin belirlenme şekli, tanımın bir istemci uygulaması için mi, yoksa bir hizmet uygulaması için mi olduğuna bağlıdır.

Bir istemci uygulaması için, istemci kısmına aşağıdaki gibi bir uç noktası tanımlaması ekleyin:

1. Menüyü açmak için **İstemci** > **Uç Noktaları** öğesini farenin sağ düğmesiyle tıklatın ve **Yeni İstemci Uç Noktası** öğesini seçin.
2. Alanları bu çizelgede gösterildiği gibi tamamlayın:

Çizelge 192. Yeni istemci uç noktası alanları	
Alan	**Değer**
NAME	`Endpoint_WMQ`
Adres	*Hizmete erişmek için gereken WMQ bağlantısı ayrıntılarını açıklayan SOAP/JMS URI.* Daha fazla ayrıntı için bkz: “WCF uç noktası URI adresi biçimi için IBM MQ özel kanalı” sayfa 1225
Bağ Tanımı	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Sözleşme	*Hizmet sözleşmesi arabiriminizin adı*

Bir hizmet uygulaması için, hizmetler bölümüne aşağıdaki gibi bir hizmet tanımı ekleyin:

1. Menüyü açmak için **Services** (Hizmetler) öğesini sağ tıklatın ve **New Service**(Yeni Hizmet) seçeneğini belirleyin ve barındırılacak hizmet sınıfını seçin.
2. Yeni hizmetinize ilişkin **Uç Noktalar** bölümüne bir uç nokta tanımlaması ekleyin ve alanları bu çizelgede gösterildiği gibi tamamlayın:

Çizelge 193. Yeni hizmet uç noktası alanları	
Alan	**Değer**
NAME	`Endpoint_WMQ`
Adres	*Hizmete erişmek için gereken WMQ bağlantısı ayrıntılarını açıklayan SOAP/JMS URI.* Daha fazla ayrıntı için bkz: "WCF uç noktası URI adresi biçimi için IBM MQ özel kanalı" sayfa 1225
Bağ Tanımı	`customBinding`
BindingConfiguration	`CustomBinding_WMQ`
Sözleşme	*Hizmet gerçekleştirme sınıfınızın adı*

Bağ tanımlama ve uç nokta bilgilerini programlı olarak sağlayarak WCF özel kanalı yaratılması

WCF için IBM MQ özel kanalı, aktarım düzeyi bir WCF kanalıdır. Özel kanalı kullanmak için bir uç nokta ve bağ tanımı tanımlanmalıdır ve bu tanımlar doğrudan uygulama kodundan programlı olarak gerçekleştirilebilir.

İletim düzeyi WCF kanalı olan WCF için IBM MQ özel kanalını yapılandırmak ve kullanmak için bir bağ tanımı ve bir uç nokta tanımlaması tanımlanmalıdır. Bağ tanımı, kanala ilişkin konfigürasyon bilgilerini içerir ve uç nokta tanımı bağlantı ayrıntılarını içerir. Daha fazla bilgi için bkz. "WCF örneklerinin kullanılması" sayfa 1234.

Bu tanımlamalar iki şekilde yaratılabilir:

- Yönetimsel olarak, "Bir uygulama yapılandırma dosyasında bağ tanımı ve uç noktası bilgileri sağlayarak yönetimsel olarak WCF özel kanalı oluşturma" sayfa 1222başlıklı konuda açıklandığı gibi bir uygulama yapılandırma dosyasında ayrıntıları sağlayarak.
- Aşağıdaki alt başlıklarda açıklandığı gibi, doğrudan uygulama kodundan programlı olarak.

Bağlama ve uç nokta bilgilerinin programlı olarak tanımlanması: SOAP/JMS arabirimi

SOAP/JMS arabirimi için, doğrudan uygulama kodundan programlı olarak bir uç nokta ve bağ tanımı tanımlayabilirsiniz.

Bu görev hakkında

Bağlama ve uç nokta bilgilerini programlı olarak sağlamak için, aşağıdaki adımları tamamlayarak uygulamanıza gerekli kodu ekleyin.

Yordam

1. Uygulamanıza aşağıdaki kodu ekleyerek, kanalın iletim bağ tanımı öğesinin bir eşgörünümünü yaratın:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
```

2. Örneğin, `ClientConnectionMode`ayarını tanımlamak için uygulamanıza aşağıdaki kodu ekleyerek gerekli bağlama özelliklerini ayarlayın:

```
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.AS_URI;
```

3. Uygulamanıza aşağıdaki kodu ekleyerek, iletim kanalını bir ileti kodlayıcısıyla eşleyen özel bir bağ tanımı yaratın:

```
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
```

4. SOAP/JMS URI 'sini oluşturun.

   Hizmete erişmek için gereken IBM MQ bağlantı ayrıntılarını açıklayan SOAP/JMS URI, uç noktası adresi olarak sağlanmalıdır. Belirlediğiniz adres, kanalın bir hizmet uygulaması için mi, yoksa bir istemci uygulaması için mi kullanıldığına bağlıdır.

   - İstemci uygulamaları için, SOAP/JMS URI aşağıdaki gibi bir EndpointAddress olarak yaratılmalıdır:

     ```
     EndpointAddress address = new EndpointAddress("jms:/queue?
     destination=SampleQ@QM1&connectionFactory
     =connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
     ```

   - Hizmet uygulamaları için, SOAP/JMS URI, aşağıdaki gibi bir URI olarak oluşturulmalıdır:

     ```
     Uri address = new Uri("jms:/queue?destination=SampleQ@QM1&connectionFactory=
     connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
     ```

     Uç nokta adresleri hakkında daha fazla bilgi için bkz. “WCF uç noktası URI adresi biçimi için IBM MQ özel kanalı” sayfa 1225.

Bağlama ve uç nokta bilgilerini programlı olarak tanımlama: SOAP/Non-JMS arabirimi
Non-SOAP/Non-JMS arabirimi için, doğrudan uygulama kodundan bir uç nokta ve bağ tanımlama işlemi tanımlayabilirsiniz.

Bu görev hakkında

Bağlama ve uç nokta bilgilerini programlı olarak sağlamak için, aşağıdaki adımları tamamlayarak uygulamanıza gerekli kodu ekleyin.

Yordam

1. Uygulamanıza aşağıdaki kodu ekleyerek bir WmqBinding yaratın:

   ```
   WmqBinding binding = new WmqBinding();
   ```

   Bu kod, SOAP/Non-JMS arabirimi için gereken WmqMsgEncodingElement ve WmqIbmTransportBindingöğesini eşleyen bir bağ tanımı yaratır.
2. Hizmete erişmek için gereken IBM MQ bağlantı ayrıntılarını tanımlayan wmq: // URI değerini belirtin.

   wmq: // URI değerini sağlama biçiminiz, kanalın bir hizmet uygulaması için mi, yoksa bir istemci uygulaması için mi kullanıldığına bağlıdır.

   - İstemci uygulamaları için wmq: // URI, aşağıdaki gibi bir EndpointAddress olarak yaratılmalıdır:

     ```
     EndpointAddress address = new EndpointAddress
     ("wmq://localhost:1414/msg/queue/Q1?connectQueueManager=QM1&replyTo=Q2");
     ```

   - Hizmet uygulamaları için, wmq: // URI aşağıdaki gibi bir URI olarak yaratılmalıdır:

     ```
     Uri sampleAddress = new Uri(
     "wmq://localhost:1414/msg/queue/Q1?connectQueueManager=QM1&replyTo=Q2");
     ```

WCF uç noktası URI adresi biçimi için IBM MQ özel kanalı

Bir web hizmeti, konum ve bağlantı ayrıntılarını sağlayan bir URI (Universal Resource Identifier; Evrensel Kaynak Tanıtıcısı) kullanılarak belirtilir. URI biçimi SOAP/JMS arabirimini mi, yoksa SOAP/Non-JMS arabirimini mi kullandığınıza bağlıdır.

SOAP/JMS arabirimi

IBM MQ iletiminde SOAP için desteklenen URI biçimi, hedef hizmetlere erişilirken SOAP/ IBM MQ ' e özgü parametreler ve seçenekler üzerinde kapsamlı bir denetim derecesi sağlar. Bu biçim, WebSphere Application Server ve CICSile uyumludur ve IBM MQ ' in her iki ürünle de bütünleştirilmesini kolaylaştırır.

URI sözdizimi şöyledir:

```
jms:/queue? name=value&name=value...
```

Burada ad bir parametre adıdır ve *değer* uygun bir değerdir ve ad = *değer* öğesi, ikinci ve sonraki geçişler ve ardından gelen geçişler bir ve işareti (&) ile herhangi bir sayıda yinelenebilir.

Parametre adları, IBM MQ nesnelerinin adları gibi büyük ve küçük harfe duyarlıdır. Herhangi bir parametre bir kereden fazla belirtilirse, parametrenin son geçişi, istemci uygulamalarının URI ' ye ekleyerek parametre değerlerini geçersiz kılabileceği anlamına gelir. Tanınmayan ek parametreler eklenirse, bunlar yoksayılır.

Bir URI ' yi bir XML dizgisinde saklarsanız, ve karakterini "&" olarak göstermeniz gerekir. Benzer şekilde, bir URI bir komut dosyasında kodlandıysa, **&** gibi kaçış karakterlerine dikkat edin; tersi durumda, kabuk tarafından yorumlanır.

Bu, bir Axis hizmeti için basit bir URI örneğidir:

```
jms:/queue?destination=myQ&connectionFactory=()
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Aşağıda, bir .NET hizmeti için basit bir URI örneği verilmiştir:

```
jms:/queue?destination=myQ&connectionFactory=()&targetService=MyService.asmx
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Yalnızca gerekli parametreler sağlanır (`targetService` yalnızca .NET hizmetleri için gereklidir) ve `connectionFactory` ' ye seçenek verilmez.

Bu Eksen örneğinde, `connectionFactory` birkaç seçenek içerir:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)
binding(client)clientChannel(myChannel)clientConnection(myConnection)
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Bu Eksen örneğinde, `connectionFactory` `sslPeerName` seçeneği de belirtilmiştir. sslPeerAdının değeri, ad değeri çiftlerini ve anlamlı gömülü boşlukları içerir:

```
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
```

NON-SOAP/Non-JMS arabirimi

NON-SOAP/Non-JMS arabirimine ilişkin URI biçimi, hedef hizmetlere erişirken IBM MQ ' e özgü parametreler ve seçenekler üzerinde kapsamlı bir denetim derecesi sağlar.

URI sözdizimi şöyledir:

```
wmq://example.com:1415/msg/queue/INS.QUOTE.REQUEST@MOTOR.INS ?ReplyTo=msg/queue/
INS.QUOTE.REPLY@BRANCH452&persistence=MQPER_NOT_PERSISTENT
```

Bu IRI, bir hizmet isteğine, 1415 numaralı kapıda example.com adlı bir makineye IBM MQ TCP istemci bağlama bağlantısı kullanabileceğini ve kalıcı istek iletilerini INS.QUOTE.REQUEST kuyruk yöneticisinde MOTOR.INSadı verilen bir kuyruğa koyabileceğini bildirir. IIRI, hizmet sağlayıcısının yanıtları

INS.QUOTE.REPLY on kuyruk yöneticisi BRANCH452. URI biçimi SupportPac MA93için belirtilmiştir. IBM MQ IRI belirtimlerine ilişkin daha fazla bilgi için bkz. SupportPac MA93: IBM MQ -Hizmet Tanımı .

WCF bağ tanımı yapılanış seçenekleri

Özel kanal bağlama bilgilerine yapılandırma seçeneklerini uygulamanın iki yolu vardır. Özellikleri denetimci olarak ayarlayabilir ya da programlı olarak ayarlayabilirsiniz.

Bağ tanımlama yapılandırma seçenekleri iki farklı yoldan biriyle ayarlanabilir:

1. Yönetimsel olarak: Bağlama özelliği ayarları, uygulama yapılandırma dosyasındaki özel bağ tanımı tanımlamasının iletim kısmında belirtilmelidir; örneğin: `app.config`.
2. Programlı olarak: Özel bağ tanımının kullanıma hazırlanması sırasında özelliği belirtmek için uygulama kodu değiştirilmelidir.

Bağ tanımlama özelliklerinin yönetimsel olarak ayarlanması

Bağ tanımlama özelliği ayarları uygulama yapılandırma dosyasında belirtilebilir; örneğin: `app.config`. Yapılandırma dosyası, aşağıdaki örneklerde gösterildiği gibi **`svcutil`**tarafından oluşturulur.

SOAP/JMS arabirimi

```
<customBinding>
...
      <IBM.XMS.WCF.SoapJmsIbmTransportChannel maxBufferPoolSize="524288"
      maxMessageSize="4000000" clientConnectionMode="0" maxConcurrentCalls="16"/>
...
</customBinding>
```

SOAP/Non-JMS arabirimi

```
<customBinding>
      <IBM.WMQ.WCF.WmqMsgEncodingElement/>
      <IBM.WMQ.WCF.WmqIbmTransportChannel maxBufferPoolSize="524288"
      maxMessageSize="65536" clientConnectionMode="managedclient"/>
</customBinding>
```

Bağ tanımlama özelliklerinin programlı olarak ayarlanması

İstemci bağlantı kipini belirtmek üzere bir WCF bağ tanımı özelliği eklemek için, hizmet kodunu özel bağ tanımının kullanıma hazırlanması sırasında özelliği belirtecek şekilde değiştirmeniz gerekir.

Yönetilmeyen istemci bağlantı kipini belirtmek için aşağıdaki örneği kullanın:

```
SoapJmsIbmTransportBindingElement
transportBindingElement = new SoapJmsIbmTransportBindingElement();
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.CLIENT_UNMANAGED;

Binding sampleBinding = new CustomBinding(new TextMessageEncodingBindingElement(),
                                                    transportBindingElement);
```

WCF bağ tanımı özellikleri

Çizelge 194. Yönetimsel ya da programsal olarak ayarlanırken bağ tanımlama özelliklerinin değerleri

Özellik adı	İstemci ya da Hizmet uygulaması	Yönetim değeri	Programlı değer	Açıklama
maxBufferPoolSize	Her ikisi	0-64 bit işaretli tamsayı	0-64 bit işaretli tamsayı	Kanalın bir yönetim ortamına ilişkin WCF ileti arabelleklerini saklamak için kullanılabilecek bellek büyüklüğü üst sınırını belirtir.
maxMessageBoyutu	Her ikisi	1-32 bit arasında işaretli tamsayı	1-32 bit arasında işaretli tamsayı	Tek bir WCF iletisi için kullanılabilecek bellek üst sınırını belirtir.
clientConnectionKipi	Her ikisi	0 (Varsayılan değer) 1	AS_URI (Varsayılan değer) ISTEMCI YöNETILMEYEN	Aktarım kanalının istemci bağlantı kipini belirtir. 0 , istemci bağlantı kipinin URI ' de belirtildiği gibi olduğu anlamına gelir. Yalnızca istemci bağlantısı kullanıldığında kullanılır. İstemci bağlantı kipinin URI ' de belirtildiği gibi olduğunu belirtir. İstemci bağlantı kipi ayarlanmazsa, varsayılan değer 0 'dır. 1 , istemci bağlantı kipinin yönetilmeyen bir istemci olduğu anlamına gelir. Yalnızca istemci bağlantısı kullanıldığında kullanılır.
MaxConcurrentÇağrıları	Müşteri	Aralık 0-2 147 483 647 'dir. 16 varsayılan değerdir	Aralık 0-2 147 483 647 'dir. 16 varsayılan değerdir	Bu özellik, bir kerede tek bir istemci yetkili sunucusunda gerçekleştirilebilecek koşutzamanlı işlem sayısı üst sınırını tanımlar. Daha fazla işlem başlatılırsa, devam eden bir işlem tamamlanıncaya ya da zamanaşımına uğrayıncaya kadar kuyruğa alınır. Bu ayar, tek bir yetkili sunucu tarafından kullanılabilecek iş parçacığı ve kaynak sayısı üst sınırını denetlemek için kullanılabilir. 0 , tüm işlemlerin eşzamanlı olarak denenmesini sağlayarak bu sınırı kaldırır.

Çizelge 194. Yönetimsel ya da programsal olarak ayarlanırken bağ tanımlama özelliklerinin değerleri (devamı var)

Özellik adı	İstemci ya da Hizmet uygulaması	Yönetim değeri	Programlı değer	Açıklama
MaxConcurrentÇağrıları	Hizmet	Aralık 1-2 147 483 647 16 varsayılan değerdir	Aralık 1-2 147 483 647 16 varsayılan değerdir	Bu özellik yalnızca güvenli teslim özelliği etkinleştirildiyse kullanılır (Emin teslim hakkında daha fazla bilgi için bkz. “WCF özel kanal güvenli teslim” sayfa 1217). Belirtilen uç nokta için aynı anda devam edebilen koşutzamanlı işlem sayısı üst sınırını belirler. Bu ayar değiştirilirken dikkatli olun. Her eşzamanlı işlem, istekleri işlemek için ek kaynaklar, özellikle de özel kanalın yeni bir eşgörünümü ve iş parçacığı havuzundaki ilişkili iş parçacıkları gerektirir. Aşırı ayırma, karşı üretken olabilir ve performansı önemli ölçüde etkileyebilir. Bu özelliği desteklemek için uygun iş parçacığı havuzu yapılandırması yapılmalıdır.

WCF için oluşturma ve barındırma hizmetleri

WCF hizmetlerinin nasıl yaratılacağını ve yapılandırılacağını açıklayan Microsoft Windows Communication Foundation (WCF) hizmetlerine genel bakış.

WCF ve WCF hizmetlerini kullanan IBM MQ özel kanalı aşağıdaki yöntemlerle barındırılabilir:

- Kendi kendini barındırma
- Windows Hizmet

WCF için IBM MQ özel kanalı Windows Süreç Etkinleştirme Hizmeti 'nde bulunamaz.

Aşağıdaki konularda, ilgili adımları göstermek için basit kendi kendini barındırma örnekleri bulunmaktadır. Daha fazla bilgi ve en son ayrıntıları içeren Microsoft WCF çevrimiçi belgeleri, https://msdn.microsoft.comadresindeki Microsoft MSDN web sitesinde bulunabilir.

1. yöntemi kullanarak WCF hizmet uygulamaları oluşturulması: Bir uygulama yapılandırma dosyası kullanılarak yönetimsel olarak kendi kendini barındırma

Bir uygulama yapılandırma dosyası yaratıldıktan sonra, hizmetin bir örneğini açın ve belirtilen kodu uygulamanıza ekleyin.

Başlamadan önce

Hizmet için aşağıdaki konuda açıklandığı gibi bir uygulama yapılandırma dosyası oluşturun ya da düzenleyin: “Bir uygulama yapılandırma dosyasında bağ tanımı ve uç noktası bilgileri sağlayarak yönetimsel olarak WCF özel kanalı oluşturma” sayfa 1222

Bu görev hakkında

1. Hizmet anasisteminde hizmetin bir eşgörünümünü somutlaştırın ve açın. Hizmet tipi, hizmet yapılandırma dosyasında belirtilen hizmet tipiyle aynı olmalıdır.
2. Uygulamanıza aşağıdaki kodu ekleyin:

```
ServiceHost service = new ServiceHost(typeof(MyService));
service.Open();
...
service.Close();
```

2. yöntemi kullanarak WCF hizmet uygulamalarını oluşturma: Uygulamadan programlı olarak doğrudan kendi kendini barındırma

Bağ tanımlama özelliklerini ekleyin, gerekli hizmet sınıfının somut örneğiyle hizmet anasistemini yaratın ve hizmeti açın.

Başlamadan önce

1. Projeye özel kanal `IBM.XMS.WCF.dll` dosyasına bir başvuru ekleyin. `IBM.XMS.WCF.dll` , *`WMQInstallDir`*`\bin` içinde bulunur; burada *WMQInstallDir* , IBM MQ ' in kurulu olduğu dizindir.
2. `IBM.XMS.WCF` ad alanına bir *kullanma* deyimi ekleyin, örneğin: `using IBM.XMS.WCF`
3. “Bağ tanımlama ve uç nokta bilgilerini programlı olarak sağlayarak WCF özel kanalı yaratılması” sayfa 1224 konusunda açıklandığı gibi, kanal bağlama öğesinin ve uç noktasının bir eşgörünümünü yaratın:

Bu görev hakkında

Kanalın bağlama özelliklerinde değişiklik yapılması gerekiyorsa, aşağıdaki adımları tamamlayın:

1. Aşağıdaki örnekte gösterildiği gibi `transportBindingElement` ' e bağ tanımlama özelliklerini ekleyin:

```
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. Gerekli hizmet sınıfının somut örneğiyle hizmet anasistemini yaratın:

```
ServiceHost service = new ServiceHost(typeof(MyService));
```

3. Hizmeti aç:

```
service.AddServiceEndpoint(typeof(IMyServiceContract), binding, address);
service.Open();
...
service.Close();
```

HTTP uç noktası kullanarak meta verileri gösterme

WCF için IBM MQ özel kanalını kullanacak şekilde yapılandırılmış bir hizmetin meta verilerini gösterme yönergeleri.

Bu görev hakkında

Hizmet meta verilerinin gösterilmesi gerekiyorsa (örneğin, `svcutil` gibi araçların çevrimdışı bir WSDL dosyasından değil, doğrudan çalışan hizmetten erişebilmesi için), HTTP uç noktasıyla hizmet meta verilerinin gösterilmesi gerekir. Bu ek uç noktayı eklemek için aşağıdaki adımlar kullanılabilir.

1. Meta verilerin ServiceHost' a gösterilebilmesi için temel adresi ekleyin; örneğin:

```
ServiceHost service = new ServiceHost(typeof(TestService),
                      new Uri("http://localhost:8000/MyService"));
```

2. Hizmet açılmadan önce aşağıdaki kodu ServiceHost hizmetine ekleyin:

```
ServiceMetadataBehavior metadataBehavior = new ServiceMetadataBehavior();
  metadataBehavior.HttpGetEnabled = true;
  service.Description.Behaviors.Add(metadataBehavior);
  service.AddServiceEndpoint(typeof(IMetadataExchange),
     MetadataExchangeBindings.CreateMexHttpBinding(), "mex");
```

Sonuçlar

Meta veriler şimdi şu adreste kullanılabilir: `http://localhost:8000/MyService`

WCF için istemci uygulamaları oluşturuluyor

Microsoft Windows Communication Foundation (WCF) istemci uygulamalarının oluşturulmasına ve oluşturulmasına genel bakış.

Bir WCF hizmeti için istemci uygulaması yaratılabilir; istemci uygulamaları genellikle, uygulama tarafından doğrudan kullanılabilecek gerekli yapılandırma ve yetkili sunucu dosyalarını yaratmak için Microsoft ServiceModel Metadata Utility Tool (Svcutil.exe) kullanılarak oluşturulur.

Çalışmakta olan bir hizmetten meta verilerle svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılanış dosyaları oluşturulması

WCF için IBM MQ özel kanalını kullanacak şekilde yapılandırılmış bir hizmet için istemci oluşturmak üzere Microsoft svcutil.exe aracının kullanılmasına ilişkin yönergeler.

Başlamadan önce

Uygulama tarafından doğrudan kullanılabilecek gerekli yapılandırma ve yetkili sunucu dosyalarını yaratmak için svcutil aracını kullanmaya ilişkin üç önkoşul vardır:

- svcutil aracı başlatılmadan önce WCF hizmetinin çalışıyor olması gerekir.
- WCF hizmeti, çalışan bir hizmetten doğrudan istemci oluşturmak için IBM MQ özel kanal uç noktası başvurularına ek olarak HTTP kapısını kullanarak meta verilerini göstermelidir.
- Özel kanal, svcutil için yapılandırma verilerine kaydedilmelidir.

Bu görev hakkında

Aşağıdaki adımlarda, IBM MQ özel kanalını kullanacak şekilde yapılandırılmış bir hizmet için nasıl istemci oluşturulacağı açıklanmıştır, ancak çalıştırma zamanında meta verilerini ayrı bir HTTP kapısı aracılığıyla da gösterir:

1. WCF hizmetini başlatın (svcutil aracı başlatılmadan önce hizmet çalışır durumda olmalıdır).
2. Kuruluş kökünden `svcutil.exe` yapılandırma dosyasındaki ayrıntıları etkin svcutil yapılandırma dosyasına (genellikle `C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe.config` svcutil 'in IBM MQ özel kanalını tanıması için) ekleyin.
3. svcutil komutunu bir komut isteminden çalıştırın; örneğin:

```
svcutil /language:C# /r: installlocation\bin\IBM.XMS.WCF.dll
/config:app.config http://localhost:8000/IBM.XMS.WCF/samples
```

4. Oluşturulan `app.config` ve `YourService.cs` dosyalarını Microsoft Visual Studio istemci projesine kopyalayın.

Sonraki adım

Hizmetler meta verileri doğrudan alınamazsa, bunun yerine wsdl 'den istemci dosyalarını oluşturmak için svcutil kullanılabilir. Daha fazla bilgi için bkz: “WSDL ile svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturma” sayfa 1232

WSDL ile svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturma

Hizmetin meta verileri kullanılamıyorsa, WSDL ' den WCF istemcileri oluşturulmasına ilişkin yönergeler.

Çalışmakta olan bir hizmetten istemci oluşturmak için hizmetin meta verileri doğrudan alınamazsa, bunun yerine WSDL ' den istemci dosyaları oluşturmak için svcutil kullanılabilir. IBM MQ özel kanalının kullanılacağını belirtmek için WSDL ' de aşağıdaki değişiklikler yapılmalıdır:

1. Aşağıdaki ad alanı tanımlamalarını ve ilke bilgilerini ekleyin:

```
<wsdl:definitions
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">

        <wsp:Policy wsu:Id="CustomBinding_IWMQSampleContract_policy">
            <wsp:ExtactlyOne>
                <wsp:All>
                    <xms:xms xmlns:xms="http://sample.schemas.ibm.com/policy/xms" />
                </wsp:All>
            </wsp:ExactlyOne>
        </wsp:Policy>

...

</wsdl:definitions>
```

2. Bağ tanımları kısmını yeni ilke kısmına başvuruda bulunacak şekilde değiştirin ve `transport` tanımlamasını temeldeki bağ tanımlama öğesinden kaldırın:

```
<wsdl:definitions ...>

        <wsdl:binding ...>
                <wsp:PolicyReference URI="#CustomerBinding_IWMQSampleContract_policy" />
                <[soap]:binding ... transport="" />
                ...
        </wsdl:binding>
</wsdl:definitions>
```

3. svcutil komutunu bir komut isteminden çalıştırın; örneğin:

```
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
```

Uygulama yapılanış dosyasıyla istemci yetkili sunucusu kullanarak WCF istemci uygulamaları oluşturulması

Başlamadan önce

İstemci için aşağıda açıklandığı gibi bir uygulama yapılandırma dosyası oluşturun ya da düzenleyin: “Bir uygulama yapılandırma dosyasında bağ tanımı ve uç noktası bilgileri sağlayarak yönetimsel olarak WCF özel kanalı oluşturma” sayfa 1222

Bu görev hakkında

İstemci yetkili sunucusunun bir eşgörünümünü somutlaştırın ve açın. Oluşturulan yetkili sunucuya geçirilen parametre, istemci yapılandırma dosyasında belirtilen uç nokta adıyla aynı olmalıdır; örneğin, `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();
            }
```

Programlı yapılandırmayla istemci yetkili sunucusu kullanarak WCF istemci uygulamaları oluşturma

Başlamadan önce

1. Projeye özel kanal `IBM.XMS.WCF.dll` dosyasına bir başvuru ekleyin. `IBM.XMS.WCF.dll` , *`WMQInstallDir`*`\bin` dizininde bulunur; burada *WMQInstallDir* , IBM MQ ' in kurulu olduğu dizindir.
2. `IBM.XMS.WCF` ad alanına bir *kullanma* deyimi ekleyin, örneğin: `using IBM.XMS.WCF`
3. Şu konuda açıklandığı gibi, kanalın th ' bağ tanımı öğesinin ve uç noktasının bir eşgörünümünü yaratın: “Bağ tanımlama ve uç nokta bilgilerini programlı olarak sağlayarak WCF özel kanalı yaratılması” sayfa 1224

Bu görev hakkında

Kanalın bağlama özelliklerinde değişiklik yapılması gerekiyorsa, aşağıdaki adımları tamamlayın.

1. Aşağıdaki şekilde gösterildiği gibi bağlama özelliklerini `transportBindingElement` ' e ekleyin:

```
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. İstemci yetkili sunucusunu aşağıdaki şekilde gösterildiği gibi yaratın; burada *bağ tanımı* ve *uç nokta adresi* , adım 1 'de yapılandırılan ve geçirilen bağlama ve uç nokta adresidir:

```
 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();
            }
```

WCF örneklerinin kullanılması

Windows Communication Foundation (WCF) örnekleri, IBM MQ özel kanalının nasıl kullanılabileceğine ilişkin bazı basit örnekler sağlar.

Örnek projeleri oluşturmak için Microsoft.NET 3.5 SDK ya da Microsoft Visual Studio 2008 gereklidir.

Basit tek yönlü istemci ve sunucu WCF örneği

Bu örnek, tek yönlü kanal şeklini kullanarak bir WCF istemcisinden Windows Communication foundation (WCF) hizmetini başlatmak için kullanılan IBM MQ özel kanalını gösterir.

Bu görev hakkında

Hizmet, konsola dizgi çıkışı yapan tek bir yöntem uygular. İstemci, hizmet meta verilerini ayrı olarak gösterilen bir HTTP uç noktasından almak için `svcutil` aracı kullanılarak oluşturuldu; açıklamalar için bkz. "Çalışmakta olan bir hizmetten meta verilerle svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılanış dosyaları oluşturulması" sayfa 1231

Örnek, aşağıdaki yordamda açıklandığı gibi belirli kaynak adlarıyla yapılandırıldı. Kaynak adlarını değiştirmeniz gerekiyorsa, *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\app.config` dosyasında istemci uygulamasında ve *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\TestServices.cs` dosyasında hizmet uygulamasında ilgili değeri değiştirmeniz gerekir; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir. JMS uç nokta URI 'sinin biçimlendirilmesiyle ilgili daha fazla bilgi için IBM MQ ürün belgelerinde *IBM MQ Transport for SOAP* başlıklı konuya bakın. Örnek çözümü ve kaynağı değiştirmeniz gerekiyorsa, bir IDE gerekir; örneğin, Microsoft Visual Studio 8 ya da üstü.

Yordam

1. *QM1* adlı bir kuyruk yöneticisi oluşturun
2. *SampleQ* adlı bir kuyruk hedefi oluşturun
3. Dinleyicinin ileti beklemesi için hizmeti başlatın: *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\bin\Release\TestService.exe` dosyasını çalıştırın; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir.
4. İstemciyi bir kez çalıştırın: *MQ_INSTALLATION_PATH* `\tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\bin\Release\TestClient.exe` dosyasını çalıştırın; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir.

   İstemci uygulaması, *SampleQ* ' ya beş kez ileti göndererek beş kez döngü oluşturur.

Sonuçlar

Hizmet uygulaması, iletileri *SampleQ* ' dan alır ve ekranda beş kez `Hello World` görüntülenir.

Sonraki adım

Basit istek-yanıt istemcisi ve sunucu WCF örneği

Bu örnek, bir WCF istemcisinden Windows Communication foundation (WCF) hizmetini bir istek yanıt kanalı şeklini kullanarak başlatmak için kullanılan IBM MQ özel kanalını gösterir.

Bu görev hakkında

Bu hizmet, iki sayı ekleyip çıkarmak ve daha sonra sonucu döndürmek için bazı basit hesap makinesi yöntemleri sağlar. İstemci, hizmet meta verilerini ayrı olarak gösterilen bir HTTP uç noktasından almak için `svcutil` aracı kullanılarak oluşturuldu; açıklamalar için bkz. "Çalışmakta olan bir hizmetten meta

verilerle svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılanış dosyaları oluşturulması” sayfa 1231

Örnek, aşağıdaki yordamda açıklandığı gibi belirli kaynak adlarıyla yapılandırıldı. Kaynak adlarını değiştirmeniz gerekiyorsa, *MQ_INSTALLATION_PATH* `\Tools\wcf\samples\WCF\requestreply\client\app.config` dosyasındaki istemci uygulamasında ve *MQ_INSTALLATION_PATH* `\Tools\wcf\samples\WCF\requestreply\service\RequestReplyService.cs` dosyasındaki hizmet uygulamasında ilgili değeri değiştirmeniz gerekir; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir. JMS uç nokta URI 'sinin biçimlendirilmesiyle ilgili daha fazla bilgi için IBM MQ ürün belgelerinde *IBM MQ Transport for SOAP* başlıklı konuya bakın. Örnek çözümü ve kaynağı değiştirmeniz gerekiyorsa, bir IDE gerekir; örneğin, Microsoft Visual Studio 8 ya da üstü.

Yordam

1. *QM1* adlı bir kuyruk yöneticisi oluşturun
2. *SampleQ* adlı bir kuyruk hedefi oluşturun
3. *SampleReplyQ* adlı bir kuyruk hedefi oluşturun
4. Dinleyicinin ileti beklemesi için hizmeti başlatın: *MQ_INSTALLATION_PATH* `\Tools\wcf\samples\WCF\requestreply\service\bin\Release\SimpleRequestReply_Service.exe` dosyasını çalıştırın; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir.
5. İstemciyi bir kez çalıştırın: *MQ_INSTALLATION_PATH* `\Tools\wcf\samples\WCF\requestreply\client\bin\Release\SimpleRequestReply_Client.exe` dosyasını çalıştırın; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir.

Sonuçlar

İstemci çalıştırıldığında, aşağıdaki işlem dört kez başlatılır ve yinelenir; bu nedenle toplam beş ileti her bir yöne gönderilir:

1. İstemci, *SampleQ* üzerine bir istek iletisi koyar ve yanıt bekler.
2. Hizmet, istek iletisini *SampleQ*' dan alır.
3. Hizmet, iletinin içeriğini kullanarak bazı değerler ekler ve çıkarır.
4. Daha sonra hizmet, sonuçları *SampleReplyQ*' da bir iletiye koyar ve istemcinin yeni bir ileti koymasını bekler.
5. İstemci iletiyi *SampleReplyQ* ' dan alır ve sonuçları ekranda görüntüler.

Sonraki adım

IBM MQ örneği tarafından barındırılan bir .NET hizmeti için WCF istemcisi

.NET ve Javaiçin örnek istemci uygulamaları ve örnek hizmet yetkili sunucu uygulamaları sağlanır. Numuneler, hisse senedi teklifi talebini alan ve hisse senedi teklifini sağlayan bir Hisse Senedi Fiyat Teklifi hizmetini temel alır.

Başlamadan önce

Örnek, .NET SOAP over JMS hizmet barındırma ortamının IBM MQ içinde doğru bir şekilde kurulmasını ve yapılandırılmasını gerektirir ve bu ortama yerel bir kuyruk yöneticisinden erişilebilir.

.NET SOAP over JMS hizmet barındırma ortamı IBM MQ içinde doğru şekilde kurulup yapılandırıldığında ve yerel bir kuyruk yöneticisinden erişilebilir olduğunda, ek yapılandırma adımlarının tamamlanması gerekir.

1. **`WMQSOAP_HOME`** ortam değişkenini IBM MQ kuruluş dizinine ayarlayın, örneğin: `C:\Program Files\IBM\MQ`
2. Java derleyicinin `javac` kullanılabilir olduğunu ve PATH değişkeninde bulunduğunu doğrulayın.

3. `axis.jar` dosyasını, kuruluş görüntüsünün `prereqs/axis` dizininden IBM MQ üretim dizinine kopyalayın; örneğin: `C:\Program Files\IBM\MQ\java\lib\soap`
4. PATH değişkenine ekleyin: *MQ_INSTALLATION_PATH*`\Java\lib` ; burada *MQ_INSTALLATION_PATH* , IBM MQ ' un kurulu olduğu dizini temsil eder; örneğin: `C:\Program Files\IBM\MQ`
5. .NET yerinin *MQ_INSTALLATION_PATH*`\bin\amqwcallWSDL.cmd` içinde doğru şekilde belirtildiğinden emin olun; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizini temsil eder; örneğin: `C:\Program Files\IBM\MQ`. .NET yeri belirtilebilir; örneğin: `set msfwdir=%ProgramFiles% \Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Önceki adımlar tamamlandığında, hizmeti test edin ve çalıştırın:

1. JMS üzerinde SOAP çalışma dizininize gidin.
2. Doğrulama sınamasını çalıştırmak ve hizmet dinleyicisini çalışır durumda bırakmak için aşağıdaki komutlardan birini girin:
   - .NET: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt dotnet hold` için; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizini gösterir.
   - AXIS için: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt Dotnet2AxisClient hold` ; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizini gösterir.

`hold` bağımsız değişkeni, sınama tamamlandıktan sonra dinleyicilerin çalışmasını sağlar.

Bu yapılandırma sırasında hatalar bildirilirse, yordamın aşağıdaki şekilde yeniden başlatılabilmesi için tüm değişiklikleri kaldırabilirsiniz:

1. JMS üzerinden oluşturulan SOAP dizinini silin.
2. Kuyruk yöneticisini silin.

Bu görev hakkında

Bu örnek, bir WCF istemcisinden IBM MQ içinde sağlanan .NET SOAP over JMS örnek hizmetine tek yönlü kanal şekli kullanılarak yapılan bir bağlantıyı gösterir. Hizmet, konsola bir metin dizgisi çıkaran basit bir StockQuote örneği uygular.

İstemci, "WSDL ile svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturma" sayfa 1232 içinde açıklandığı gibi istemci dosyaları oluşturmak için WSDL kullanılarak oluşturulmuştur.

Örnek, aşağıdaki yordamda açıklandığı gibi belirli kaynak adlarıyla yapılandırıldı. Kaynak adlarını değiştirmeniz gerekirse, *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQNET\default\client\app.config` dosyasındaki istemci uygulamasında ve *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQNET\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl` dosyasındaki hizmet uygulamasında da karşılık gelen değeri değiştirmeniz gerekir; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinini gösterir. JMS uç nokta URI 'sinin biçimlendirilmesiyle ilgili daha fazla bilgi için IBM MQ ürün belgelerinde *IBM MQ Transport for SOAP* başlıklı konuya bakın.

Yordam

İstemciyi bir kez çalıştırın: *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQNET\default\client\bin\Release\TestClient.exe` dosyasını çalıştırın; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinini gösterir.

İstemci uygulaması, örnek kuyruğa beş ileti göndererek beş kez döngü oluşturur.

Sonuçlar

Hizmet uygulaması iletileri örnek kuyruktan alır ve ekranda beş kez `Hello World` görüntülenir.

IBM MQ örneği tarafından barındırılan bir Axis Java hizmeti için WCF istemcisi

Java ve .NETiçin örnek istemci uygulamaları ve örnek hizmet yetkili sunucu uygulamaları sağlanır. Numuneler, hisse senedi teklifi talebini alan ve hisse senedi teklifini sağlayan bir Hisse Senedi Fiyat Teklifi hizmetini temel alır.

Başlamadan önce

Bu örnek, .NET SOAP over JMS hizmet barındırma ortamının IBM MQ içinde doğru şekilde kurulmasını ve yapılandırılmasını gerektirir ve bu ortama yerel bir kuyruk yöneticisinden erişilebilir.

.NET SOAP over JMS hizmet barındırma ortamı IBM MQ içinde doğru şekilde kurulup yapılandırıldığında ve yerel bir kuyruk yöneticisinden erişilebilir olduğunda, ek yapılandırma adımlarının tamamlanması gerekir.

1. **WMQSOAP_HOME** ortam değişkenini IBM MQ kuruluş dizinine ayarlayın, örneğin: `C:\Program Files\IBM\MQ`
2. Java derleyicinin `javac` kullanılabilir olduğunu ve PATH değişkeninde bulunduğunu doğrulayın.
3. `axis.jar` dosyasını, kuruluş görüntüsünün `prereqs/axis` dizininden IBM MQ kuruluş dizinine kopyalayın.
4. PATH değişkenine ekleyin: *MQ_INSTALLATION_PATH*`\Java\lib` ; burada *MQ_INSTALLATION_PATH* , IBM MQ ' un kurulu olduğu dizini temsil eder; örneğin: `C:\Program Files\IBM\MQ`
5. .NET yerinin *MQ_INSTALLATION_PATH*`\bin\amqwcallWSDL.cmd` içinde doğru şekilde belirtildiğinden emin olun; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizini temsil eder; örneğin: `C:\Program Files\IBM\MQ`. .NET yeri belirtilebilir; örneğin: `set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin`

Önceki adımlar tamamlandığında, hizmeti test edin ve çalıştırın:

1. JMS üzerinde SOAP çalışma dizininize gidin.
2. Doğrulama sınamasını çalıştırmak ve hizmet dinleyicisini çalışır durumda bırakmak için aşağıdaki komutlardan birini girin:
   - .NET: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt dotnet hold` için; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizini gösterir.
   - AXIS için: *MQ_INSTALLATION_PATH*`\Tools\soap\samples\runivt Dotnet2AxisClient hold` ; burada *MQ_INSTALLATION_PATH* , IBM MQ ' in kurulu olduğu dizini gösterir.

`hold` bağımsız değişkeni, sınama tamamlandıktan sonra dinleyicilerin çalışmasını sağlar.

Bu yapılandırma sırasında hatalar bildirilirse, yordamın aşağıdaki şekilde yeniden başlatılması için tüm değişiklikleri kaldırabilirsiniz:

1. JMS üzerinden oluşturulan SOAP dizinini silin.
2. Kuyruk yöneticisini silin.

Bu görev hakkında

Örnek, tek yönlü kanal şekli kullanılarak IBM MQ içinde sağlanan bir WCF istemcisinden Axis Java SOAP over JMS örnek hizmetine yapılan bir bağlantıyı gösterir. Hizmet, geçerli dizinde saklanan bir dosyaya bir metin dizgisi çıkaran basit bir StockQuote örneği uygular.

İstemci, “WSDL ile svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturma” sayfa 1232 içinde açıklandığı gibi istemci dosyaları oluşturmak için WSDL kullanılarak oluşturulmuştur.

Örnek, bu paragrafta açıklandığı gibi belirli kaynak adlarıyla yapılandırılmıştır. Kaynak adlarını değiştirmeniz gerekiyorsa, *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WMQAxis\default\client\app.config` dosyasındaki istemci uygulamasında ve *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WMQAxis\default\service\WmqDefaultSample_StockQuoteDotNet.ws`

dl dosyasındaki hizmet uygulamasında da karşılık gelen değeri değiştirmeniz gerekir; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinini gösterir.

Yordam

İstemciyi bir kez çalıştırın: *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WMQAxis\default\client\bin\Release\TestClient.exe` dosyasını çalıştırın; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinini gösterir.

İstemci uygulaması, örnek kuyruğa beş ileti göndererek beş kez döngü oluşturur.

Sonuçlar

Hizmet uygulaması iletileri örnek kuyruktan alır ve yürürlükteki dizindeki bir dosyaya beş kez `Hello World` ekler.

WCF istemcisinden Java hizmetine WebSphere Application Server örneği

WebSphere Application Server 6için örnek istemci uygulamaları ve örnek hizmet yetkili sunucu uygulamaları sağlanır. Bir istek-yanıt hizmeti de sağlanır.

Başlamadan önce

Bu örnek, aşağıdaki IBM MQ yapılandırmasının kullanılmasını gerektirir:

Çizelge 195. IBM MQ gerekli yapılandırma

Nesne	**Gerekli ad**
Kuyruk yöneticisi	QM1
Yerel kuyruk	HelloWorld
Yerel kuyruk	HelloWorldYanıtı

Bu örnek, WebSphere Application Server 6 barındırma ortamının doğru bir şekilde kurulmasını ve yapılandırılmasını da gerektirir. WebSphere Application Server 6 , varsayılan olarak IBM MQ ' a bağlanmak için bir bağ tanımlama kipi bağlantısı kullanır. Bu nedenle WebSphere Application Server 6 , kuyruk yöneticisiyle aynı makineye kurulmalıdır.

WAS ortamı yapılandırıldıktan sonra aşağıdaki ek yapılandırma adımları tamamlanmalıdır:

1. WebSphere Application Server JNDI havuzunda aşağıdaki JNDI nesnelerini yaratın:
   a. `HelloWorld` adlı bir JMS kuyruk hedefi
      - JNDI adını `jms/HelloWorld` olarak ayarlayın
      - Kuyruk adını `HelloWorld` olarak ayarlayın
   b. `HelloWorldQCF` adlı bir JMS kuyruk bağlantısı üreticisi
      - JNDI adını `jms/HelloWorldQCF` olarak ayarlayın
      - Kuyruk yöneticisi adını `QM1` olarak ayarlayın
   c. `WebServicesReplyQCF` adlı bir JMS kuyruk bağlantısı üreticisi
      - JNDI adını `jms/WebServicesReplyQCF` olarak ayarlayın
      - Kuyruk yöneticisi adını `QM1` olarak ayarlayın
2. Aşağıdaki yapılandırmayla WebSphere Application Server içinde `HelloWorldPort` adlı bir İleti Dinleyici Kapısı oluşturun:
   - Bağlantı üreticisi JNDI adını `jms/HelloWorldQCF` olarak ayarlayın
   - Hedef JNDI adını `jms/HelloWorld` olarak ayarlayın

3. Web hizmeti `HelloWorldEJBEAR.ear` uygulamasını WebSphere Application Server sunucunuza aşağıdaki şekilde kurun:
   a. **Uygulamalar** > **Yeni Uygulama** > **Yeni Kurumsal Uygulama**seçeneğini tıklatın.
   b. *MQ_INSTALLATION_PATH*`\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` adresine gidin; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir.
   c. Sihirbazdaki varsayılan seçeneklerin hiçbirini değiştirmeyin ve uygulama kurulduktan sonra uygulama sunucusunu yeniden başlatın.

WAS yapılandırması tamamlandığında, hizmeti bir kez çalıştırarak test edin:

1. JMS üzerindeki çalışma dizininize gidin.
2. Örneği çalıştırmak için bu komutu girin: *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\TestClient.exe` ; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir.

Bu görev hakkında

Örnek, bir WCF istemcisinden IBM MQiçinde bulunan WCF örneklerinde verilen WebSphere Application Server SOAP over JMS örnek hizmetine bir istek yanıt kanalı şekli kullanılarak yapılan bağlantıyı gösterir. WCF ile WebSphere Application Server kullanan IBM MQ kuyrukları arasındaki ileti akışı. Hizmet, bir dizgi alan ve istemciye bir karşılama döndüren `HelloWorld(...)` yöntemini uygular.

İstemci, hizmet meta verilerini “Çalışmakta olan bir hizmetten meta verilerle svcutil aracını kullanarak WCF istemcisi yetkili sunucusu ve uygulama yapılanış dosyaları oluşturulması” sayfa 1231 içinde açıklandığı gibi ayrı olarak gösterilen bir HTTP uç noktasından almak için svcutil aracı kullanılarak oluşturulmuştur.

Örnek, aşağıdaki yordamda açıklandığı gibi belirli kaynak adlarıyla yapılandırıldı. Kaynak adlarını değiştirmeniz gerekiyorsa, *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\default\client\app.config` dosyasındaki istemci uygulamasında ve *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\HelloWorldsEJBEAR.ear` içindeki hizmet uygulamasında ilgili değeri değiştirmeniz gerekir; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir.

Hizmet ve müşteri, IBM Developer makalede *JMS ve WebSphere Studio üzerinden SOAP kullanarak JMS web hizmeti oluşturma*özetlenen hizmet ve istemciyi temel alır. IBM MQ WCF özel kanalı ile uyumlu JMS üzerinden SOAP web hizmetleri geliştirme hakkında daha fazla bilgi için bkz. https://www.ibm.com/developerworks/websphere/library/techarticles/0402_du/0402_du.html.

Yordam

İstemciyi bir kez çalıştırın: *MQ_INSTALLATION_PATH* `\tools\wcf\samples\WAS\default\client\bin\Release\TestClient.exe` dosyasını çalıştırın; burada *MQ_INSTALLATION_PATH* , IBM MQkuruluş dizinidir.

İstemci uygulaması, örnek kuyruğa iki ileti göndererek aynı anda her iki hizmet yöntemini de başlatır.

Sonuçlar

Hizmet uygulaması iletileri örnek kuyruktan alır ve istemci uygulamasının konsola çıkardığı `HelloWorld(...)` yöntemi çağrısına yanıt verir.

Özel notlar

Bu belge, ABD'de kullanıma sunulan ürünler ve hizmetler için hazırlanmıştır.

IBM, bu belgede sözü edilen ürün, hizmet ya da özellikleri diğer ülkelerde kullanıma sunmayabilir. Bulunduğunuz yerde kullanıma sunulan ürün ve hizmetleri yerel IBM müşteri temsilcisinden ya da çözüm ortağından öğrenebilirsiniz. Bir IBM ürün, program ya da hizmetine gönderme yapılması, açık ya da örtük olarak, yalnızca o IBM ürünü, programı ya da hizmetinin kullanılabileceğini göstermez. Aynı işlevi gören ve IBM'in fikri mülkiyet haklarına zarar vermeyen herhangi bir ürün, program ya da hizmet de kullanılabilir. Ancak, IBM dışı ürün, program ya da hizmetlerle gerçekleştirilen işlemlerin değerlendirilmesi ve doğrulanması kullanıcının sorumluluğundadır.

IBM'in, bu belgedeki konularla ilgili patentleri ya da patent başvuruları olabilir. Bu belgenin size verilmiş olması, patentlerin izinsiz kullanım hakkının da verildiği anlamına gelmez. Lisansla ilgili sorularınızı aşağıdaki adrese yazabilirsiniz:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Çift byte (DBCS) bilgilerle ilgili lisans soruları için, ülkenizdeki IBM'in Fikri Haklar (Intellectual Property) bölümüyle bağlantı kurun ya da sorularınızı aşağıda adrese yazın:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japonya

İzleyen paragraf, bu tür kayıt ve koşulların, yasalarıyla bağdaşmadığı ülkeler ya da bölgeler için geçerli değildir: IBM BU YAYINI, "OLDUĞU GİBİ" , HİÇBİR KONUDA AÇIK YA DA ÖRTÜK GARANTİ VERMEKSİZİN SAĞLAMAKTADIR; TİCARİ KULLANIMA UYGUNLUK AÇISINDAN HER TÜRLÜ GARANTİ VE BELİRLİ BİR AMACA UYGUNLUK İDDİASI AÇIKÇA REDDEDİLİR. Bazı ülkeler bazı işlemlerde garantinin açık ya da örtük olarak reddedilmesine izin vermez; dolayısıyla, bu bildirim sizin için geçerli olmayabilir.

Bu yayın teknik yanlışlar ya da yazım hataları içerebilir. Buradaki bilgiler üzerinde düzenli olarak değişiklik yapılmaktadır; söz konusu değişiklikler sonraki basımlara yansıtılacaktır. IBM, önceden bildirimde bulunmaksızın, bu yayında açıklanan ürünler ve/ya da programlar üzerinde iyileştirmeler ve/ya da değişiklikler yapabilir.

Bu belgede IBM dışı Web sitelerine yapılan göndermeler kullanıcıya kolaylık sağlamak içindir ve bu Web sitelerinin onaylanması anlamına gelmez. Bu Web sitelerinin içerdiği malzeme, bu IBM ürününe ilişkin malzemenin bir parçası değildir ve bu tür Web sitelerinin kullanılmasının sorumluluğu size aittir.

IBM'e bilgi ilettiğinizde, IBM bu bilgileri size karşı hiçbir yükümlülük almaksızın uygun gördüğü yöntemlerle kullanabilir ya da dağıtabilir.

(i) Bağımsız olarak yaratılan programlarla, bu program da içinde olmak üzere diğer programlar arasında bilgi değiş tokuşuna ve (ii) değiş tokuş edilen bilginin karşılıklı kullanımına olanak sağlamak amacıyla bu program hakkında bilgi sahibi olmak isteyen lisans sahipleri şu adrese yazabilirler:

IBM Corporation
Yazılım Birlikte Çalışabilirlik Koordinatörü, Bölüm 49XA
3605 Karayolu 52 N
Rochester, MN 55901
U.S.A.

Bu tür bilgiler, ilgili kayıt ve koşullar altında ve bazı durumlarda bedelli olarak edinilebilir.

Bu belgede açıklanan lisanslı program ve bu programla birlikte kullanılabilecek tüm lisanslı malzeme, IBM tarafından IBM Müşteri Sözleşmesi, IBM Uluslararası Program Lisans Sözleşmesi ya da taraflar arasında yapılan herhangi bir eşdeğer sözleşmenin koşulları kapsamında sağlanır.

Burada belirtilen performans verileri denetimli bir ortamda elde edilmiştir. Bu nedenle, başka işletim ortamlarında çok farklı sonuçlar alınabilir. Bazı ölçümler geliştirilme düzeyindeki sistemlerde yapılmıştır ve bu ölçümlerin genel kullanıma sunulan sistemlerde de aynı olacağı garanti edilemez. Ayrıca, bazı sonuçlar öngörü yöntemiyle elde edilmiş olabilir. Dolayısıyla, gerçek sonuçlar farklı olabilir. Bu belgenin kullanıcıları, kendi ortamları için geçerli verileri kendileri doğrulamalıdırlar.

IBM dışı ürünlerle ilgili bilgiler, bu ürünleri sağlayan firmalardan, bu firmaların yayın ve belgelerinden ve genel kullanıma açık diğer kaynaklardan alınmıştır. IBM bu ürünleri sınamamıştır ve IBM dışı ürünlerle ilgili performans doğruluğu, uyumluluk gibi iddiaları doğrulayamaz. IBM dışı ürünlerin yeteneklerine ilişkin sorular, bu ürünleri sağlayan firmalara yöneltilmelidir.

IBM'in gelecekteki yönelim ve kararlarına ilişkin tüm bildirimler değişebilir ve herhangi bir duyuruda bulunulmadan bunlardan vazgeçilebilir; bu yönelim ve kararlar yalnızca amaç ve hedefleri gösterir.

Bu belge, günlük iş ortamında kullanılan veri ve raporlara ilişkin örnekler içerir. Örneklerin olabildiğince açıklayıcı olması amacıyla kişi, şirket, marka ve ürün adları belirtilmiş olabilir. Bu adların tümü gerçek dışıdır ve gerçek iş ortamında kullanılan ad ve adreslerle olabilecek herhangi bir benzerlik tümüyle rastlantıdır.

YAYIN HAKKI LİSANSI:

Bu belge, çeşitli işletim platformlarında programlama tekniklerini gösteren, kaynak dilde yazılmış örnek uygulama programları içerir. Bu örnek programları, IBM'e herhangi bir ödemede bulunmadan, örnek programların yazıldığı işletim altyapısına ilişkin uygulama programlama arabirimiyle uyumlu uygulama programlarının geliştirilmesi, kullanılması, pazarlanması ya da dağıtılması amacıyla herhangi bir biçimde kopyalayabilir, değiştirebilir ve dağıtabilirsiniz. Bu örnekler her koşul altında tüm ayrıntılarıyla sınanmamıştır. Dolayısıyla, IBM bu programların güvenilirliği, bakım yapılabilirliği ya da işlevleri konusunda açık ya da örtük güvence veremez.

Bu bilgileri elektronik kopya olarak görüntülediyseniz, fotoğraflar ve renkli resimler görünmeyebilir.

Programlama arabirimi bilgileri

Sağlandıysa, programlama arabirimi bilgileri, bu programla birlikte kullanılmak üzere uygulama yazılımı oluşturmanıza yardımcı olmak amacıyla hazırlanmıştır.

Bu kitapta, müşterinin WebSphere MQhizmetlerini elde etmek üzere program yazmasına olanak sağlayan amaçlanan programlama arabirimlerine ilişkin bilgiler yer alır.

Ancak, bu bilgiler tanılama, değiştirme ve ayarlama bilgilerini de içerebilir. Tanılama, değiştirme ve ayarlama bilgileri, uygulama yazılımlarınızda hata ayıklamanıza yardımcı olur.

Önemli: Bu tanılama, değiştirme ve ayarlama bilgilerini bir programlama arabirimi olarak kullanmayın; bu bilgiler değişebilir.

Ticari Markalar

IBM, IBM logosu, ibm.com, IBM Corporation 'ın dünya çapında birçok farklı hukuk düzeninde kayıtlı bulunan ticari markalarıdır. IBM ticari markalarının güncel bir listesine Web üzerinde "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml(Telif hakkı ve ticari marka bilgileri) başlıklı konudan ulaşılabilir. Diğer ürün ve hizmet adları IBM'in veya diğer şirketlerin ticari markaları olabilir.

Microsoft ve Windows, Microsoft Corporation firmasının ABD'de ve/ya da diğer ülkelerdeki markalarıdır.

UNIX, The Open Group şirketinin ABD ve diğer ülkelerdeki tescilli ticari markasıdır.

Linux, Linus Torvalds'ın ABD ve/ya da diğer ülkelerdeki tescilli ticari markasıdır.

Bu ürün, Eclipse Project (https://www.eclipse.org/) tarafından geliştirilen yazılımları içerir.

Java ve Java tabanlı tüm markalar ve logolar, Oracle firmasının ve/ya da iştiraklerinin markaları ya da tescilli markalarıdır.

Parça numarası:

(1P) P/N: