

9.1

*IBM MQ İin Uygulamalar Geliştirilmesi*

**IBM**

**Not**

Bu bilgileri ve desteklediđi ürünü kullanmadan önce, "[Özel notlar](#)" sayfa 1323 bölümündeki bilgileri okuyun.

Bu basım, yeni basımlarında tersi belirtilmediđi sürece, IBM® MQ sürüm 9 yayın düzeyi 1 'i ve sonraki tüm yayın düzeyleri ve deđişiklikler için geçerlidir.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 2007, 2024.**

# İçindekiler

<b>Uygulamaların geliştirilmesi.....</b>	<b>5</b>
Uygulama geliştirme kavramları.....	7
Uygulamalarının gerçekleştirebileceği işlemler.....	7
Uygulamalar, uygulama adları ve uygulama örnekleri.....	9
MQI kullanan uygulama programları.....	10
Birden çok IBM MQ kuyruk yöneticisine bağlanmak için istemci bağlantılarının kullanılması.....	11
Nesne yönelimli uygulamalar.....	14
IBM MQ ileti.....	16
Microsoft Transaction Server uygulamalarının hazırlanması ve çalıştırılması.....	46
IBM MQ uygulamaları için dikkat edilmesi gereken noktalar.....	46
Desteklenen programlama dillerinde uygulama adının belirtilmesi.....	49
İletilere ilişkin tasarım teknikleri.....	54
Uygulama tasarımı ve performansı ile ilgili önemli noktalar.....	56
Gelişmiş uygulamalar için tasarım teknikleri.....	58
IBM i uygulamaları için tasarım ve performans konuları.....	59
Linux on POWER Systems - Little Endian uygulamaları için dikkat edilmesi gereken noktalar.....	61
z/OS uygulamaları için tasarım ve performans konuları.....	61
IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları.....	65
JMS ve Java uygulamalarının geliştirilmesi.....	76
kullanmaIBM MQ classes for JMS.....	77
kullanmaIBM MQ classes for Java.....	321
IBM MQ kaynak bağdaştırıcısının kullanılması.....	415
IBM MQ ve WebSphere Application Server ' in birlikte kullanılması.....	472
IBM MQ Üstbilgileri paketini kullanma.....	490
Setting up IBM MQ on IBM i with Java and JMS.....	492
Bir Maven havuzu kullanarakJava uygulama geliştirilmesi.....	499
C++ uygulamaları geliştirilmesi.....	500
C++ örnek programları.....	503
C++ dilindeki önemli noktalar.....	507
C++ dilinde ileti alışverişi.....	511
Building IBM MQ C++ programs.....	517
.NET uygulamalarının geliştirilmesi.....	528
IBM MQ classes for .NET olanağını kullanmaya başlama.....	529
IBM MQ .NET programlarının yazılması ve konuşlandırılması.....	547
XMS .NET uygulamalarının geliştirilmesi.....	581
XMStarafından desteklenen ileti alışverişi stilleri.....	582
XMS nesne modeli.....	582
XMS ileti modeli.....	585
İleti alışverişi sunucusu ortamının ayarlanması.....	585
XMS örnek uygulamalarının kullanılması.....	594
XMS uygulamaları yazılıyor.....	597
XMS .NET uygulamaları yazılıyor.....	615
Working with XMS .NET administered objects.....	620
Uygulamaların daha yeni bir XMS sürümünü kullanmalarını önleme.....	627
XMS uygulamaları için iletişimi güvenlik altına alma.....	628
XMS ileti.....	631
Component Object Model Interface olanağının kullanılması ( ActiveXiçinIBM MQ Otomasyon Sınıfları).....	640
ActiveXiçin IBM MQ Otomasyon Sınıflarını kullanarak tasarlama ve programlama.....	641
ActiveX içinIBM MQ Otomasyon Sınıfları başvurusu.....	646
ActiveXiçin IBM MQ Otomasyon Sınıflarını İzleme.....	713
MQAI ' yeActiveX arabirimi.....	718

ActiveX Starter örnekleri için IBM MQ Otomasyon Sınıfları Hakkında.....	727
AMQP istemci uygulamalarının geliştirilmesi.....	731
MQ Light ve AMQP (Gelişmiş İleti Kuyruğa Yollama İletişim Kuralı).....	732
AMQP 1.0 desteği.....	733
AMQP ve IBM MQ ileti alanlarının eşlenmesi.....	734
AMQP ile ileti teslim güvenilirliği.....	741
IBM MQ ile AMQP istemcileri için topolojiler.....	743
IBM MQ ile REST uygulamaları geliştirilmesi.....	749
REST API komutunu kullanarak ileti alışverişi.....	750
Developing MQI applications with IBM MQ.....	755
IBM MQ veri tanımlama dosyaları.....	756
Kuyruğa alma için bir yordamsal uygulama yazma.....	759
İstemci yordamsal uygulamaları yazılıyor.....	945
Kullanıcı çıkışları, API çıkışları ve IBM MQ kurulabilir hizmetleri.....	968
Yordamsal uygulama oluşturulması.....	1030
Yordamsal program hatalarının işlenmesi.....	1073
Çoklu yayın programlama.....	1078
C içinde kodlama.....	1084
Visual Basic içinde kodlama.....	1087
COBOL içinde kodlama.....	1088
System/390 çevirici dilinde kodlama (İleti kuyruğu arabirimi).....	1088
Coding IBM MQ programs in RPG (IBM i only).....	1091
PL/I içinde kodlama (yalnızca z/OS).....	1091
IBM MQ örnek yordamsal programlarının kullanılması.....	1092
Managed File Transfer için uygulama geliştirilmesi.....	1249
MFT ile çalışmak üzere programların belirtilmesi.....	1249
Apache Ant ile MFT komutunu kullanma.....	1252
MFT ürününü kullanıcı çıkışlarıyla özelleştirme.....	1256
Aracı komut kuyruğuna ileti yerleştirerek MFT 'i denetleme.....	1269
MQ Telemetry için uygulama geliştirilmesi.....	1269
IBM MQ Telemetry Transport Örnek programlar.....	1270
MQTT istemci programlama kavramları.....	1271
IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi.....	1292
.NET ile WCF için IBM MQ özel kanalına giriş.....	1292
WCF için IBM MQ özel kanallarını kullanma.....	1297
WCF örneklerinin kullanılması.....	1316
<b>Özel notlar.....</b>	<b>1323</b>
Programlama arabirimi bilgileri.....	1324
Ticari Markalar.....	1324

# IBM MQ için uygulama geliştirilmesi

İletileri göndermek ve almak, kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için uygulamalar geliştirebilirsiniz. IBM MQ , birçok farklı dil ve çerçeve içinde yazılmış uygulamaları destekler.

## IBM MQ için uygulamalar geliştirmekte yeni misiniz?

IBM MQ için uygulama geliştirme hakkında bilgi edinmek için IBM Developer 'ı ziyaret edin:

- [LearnMQ \(temel bilgileri öğrenin, gösterimi çalıştırın, bir uygulamayı kodlayın, daha gelişmiş öğretici programlar edinin\)](#)
- [MQ geliştirici yüklemeleri \(ücretsiz geliştirici sürümleri ve deneme sürümleri dahil\)](#)

Aşağıdaki bölümlerde açıklanan kavramlara alışkınsanız, uygulamalarınızı geliştirmenin daha kolay olduğunu da bulabilirsiniz:

- [“Uygulama geliştirme kavramları” sayfa 7](#)
- [“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#)

## Nesne yönelimli dil ve çerçeveler için destek

IBM MQ , aşağıdaki dillerde ve çerçevelerde geliştirilen uygulamalar için çekirdek desteği sağlar:

- [JMS](#)
- [Java](#)
- [C++](#)
- [.NET](#)
- [ActiveX \(deprecated; use .NET\)](#)

Ayrıca bkz. [“Nesne yönelimli uygulamalar” sayfa 14.](#)

.NET , birçok dilde geliştirilen uygulamaları destekler. To illustrate using the IBM MQ classes for .NET to access IBM MQ queues, the MQ product documentation contains information for the following languages:

- [C# örnek kodu ve örnek uygulamalar](#)
- [C++ örnek uygulamaları](#)
- [Visual Basic örnek uygulamaları](#)

Bkz. [“IBM MQ .NET programlarının yazılması ve konuşlandırılması” sayfa 547.](#)

[V 9.1.2](#) [V 9.1.1](#) IBM MQ supports .NET Core for applications in Windows environments from IBM MQ 9.1.1 and for applications in Linux® environments from IBM MQ 9.1.2. Daha fazla bilgi için bkz [“kurmaIBM MQ classes for .NET Standard” sayfa 530.](#)

[ULW](#) IBM MQ , OASIS AMQP 1.0 iletişim kuralını uygulayan MQ Light API ' yı da destekler. Aşağıdaki diller için ileti alışverişi API ' ları vardır:

- [Node.js](#)
- [Ruby](#)
- [Java](#)
- [Python](#)
- [Maven \(çatı projesi; Java api kullanır\)](#)
- [Gradle \(çatı projesi; Java api kullanır\)](#)



Ayrıca bkz. [“AMQP istemci uygulamalarının geliştirilmesi” sayfa 731.](#)

Aşağıdaki dil bağlamaları şu şekilde sağlanır:

- [Git bağlama](#)
- [Node.js uygulamalarıyla çalışan bir JavaScript API 'si 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](#)
-  [IBM z/OS Connect EE](#)
- [IBM Integration Bus](#)
- [IBM DataPower Ağ Geçidi](#)

Bkz. “IBM MQ ile REST uygulamaları geliştirilmesi” sayfa 749ve aynı zamanda IBM Developer 'ın IBM MQ alanındaki eğitimini [IBM MQ ileti sistemi REST API ' yı kullanmaya başlama](#) . Bu eğitimde, IBM MQ messaging REST API ile birlikte kullanılmak üzere sağlanan şu dillerde örnekler yer almaktadır:

- MQ ileti sistemi REST API ' yı kullanan bir örnek
- HTTPS modülünü kullanarakNode.js örneği
- Promise modüleNode.js örneği

## Yordamsal programlama dilleri için destek

IBM MQ , aşağıdaki yordamsal programlama dillerinde geliştirilmiş uygulamalar için destek sağlar:

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

Bu diller, ileti kuyruğa alma hizmetlerine erişmek için ileti kuyruğu arabirimini (MQI) kullanır. Bkz. “Developing MQI applications with IBM MQ” sayfa 755. Nesne yönelimli diller ve çerçeveler tarafından kullanılan IBM MQ Nesne Modeli 'nin, MQI kullanılarak yordamsal dillerin kullanımına sunulmayan ek işlevler de sağladığını unutmayın.

## Uygulama adının belirtilmesi



IBM MQ 9.1.2öncesinde, Java ya da JMS istemci uygulamalarında bir uygulama adı belirtebilirsiniz. IBM MQ 9.1.2' tan, ek programlama dillerinde uygulama adını da belirtebilirsiniz. Daha fazla bilgi için bkz “Desteklenen programlama dillerinde uygulama adının belirtilmesi” sayfa 49.

### İlgili görevler

[“MQ Telemetryiçin uygulama geliştirilmesi” sayfa 1269](#)

[“IBM MQile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi” sayfa 1292](#)

IBM MQ için Microsoft Windows Communication Foundation (WCF) özel kanalı, WCF istemcileri ve hizmetleri arasında ileti gönderir ve iletileri alır.

### İlgili başvurular

[“Managed File Transferiçin uygulama geliştirilmesi” sayfa 1249](#)

Managed File Transfer ile çalıştırılacak programları belirtin, Apache Ant komutunu kullanarak Managed File Transfer komutunu kullanın, kullanıcı çıkışlarıyla Managed File Transfer uyarlayın ve aracı komut kuyruğuna ileti koyarak Managed File Transfer denetimini kullanın.

## Uygulama geliştirme kavramları

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.

IBM MQ için yazabileceğiniz uygulama tipleriyle ilgili bilgi için bkz. [“IBM MQ için uygulama geliştirilmesi” sayfa 5](#) ve [“Uygulamalarınızın gerçekleştirebileceği işlemler” sayfa 7](#).

### İlgili kavramlar

[“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#)

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğinize karar verdiğinizde, IBM MQ tarafından sunulan özelliklerin nasıl kullanılacağına karar vermeniz gerekir.

## Uygulamalarınızın gerçekleştirebileceği işlemler

İş süreçlerinizi desteklemek için gereksinim duyduğunuz iletileri göndermek ve almak için uygulamalar geliştirebilirsiniz. Kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için de uygulamalar geliştirebilirsiniz.

## Uygulamalarınızın IBM MQ for Multiplatforms üzerinde gerçekleştirebileceği eylemler

### Multi

Multiplatforms' ta, aşağıdaki eylemleri gerçekleştiren uygulamalar yazabilirsiniz:

- Aynı işletim sistemleri altında çalışan diğer uygulamalara ileti gönderme. Uygulamalar aynı ya da başka bir sistemde olabilir.
- Diğer IBM MQ platformlarında çalışan uygulamalara ileti gönderin.
- Aşağıdaki sistemler için CICS içinden ileti kuyruğa alma olanağını kullanın:

–  TXSeries - AIX

–  IBM i

–  Solaris

–  Windows

- Aşağıdaki sistemler için Encina içinden ileti kuyruklama olanağını kullanın:

–  AIX

–  Solaris

–  Windows

- Aşağıdaki sistemler için Tuxedo içinden ileti kuyruklama olanağını kullanın:

–  AIX

– AT & T

–  Solaris

–  Windows

- Use IBM MQ as a transaction manager, coordinating updates made by external resource managers within IBM MQ units of work. Aşağıdaki dış kaynak yöneticileri desteklenir ve X/XX\_ENCODE\_CASE\_ONE open XA arabirimiyle uyumludur.
  - Db2
  - Informix
  - Oracle
  - Sybase
- Birden çok iletiyi, kesinleştirilebilecek ya da yedeklenebilecek tek bir iş birimi olarak işler.
- Tam IBM MQ ortamından çalıştırın ya da bir IBM MQ istemci ortamından çalıştırın.

## Uygulamalarınızın IBM MQ for z/OS üzerinde gerçekleştirebileceği eylemler

### z/OS

z/OS' ta, aşağıdaki eylemleri gerçekleştiren uygulamalar yazabilirsiniz:

- Use message queuing within CICS or IMS.
- Toplu iş, CICS ve IMS uygulamaları arasında, her bir işlev için en uygun ortamı seçerek ileti gönderin.
- Diğer IBM MQ platformlarında çalışan uygulamalara ileti gönderin.
- Birden çok iletiyi, kesinleştirilebilecek ya da yedeklenebilecek tek bir iş birimi olarak işler.
- Send messages to, and interact with, IMS applications by means of the IMS bridge.
- 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 olmamalarıdır, ancak her ortamın avantajlarından yararlanmak için dağıtılabilirler. Örneğin, son kullanıcı arabirimlerini TSO ya da CICS kullanarak geliştirebilir, işlem yoğun modülleri z/OS toplu kipte çalıştırabilirsiniz ve veritabanı uygulamalarını IMS ya da CICS olanağında çalıştırabilirsiniz. Her durumda, uygulamanın çeşitli kısımları iletiler ve kuyruklar kullanarak iletişim kurabilir.

IBM MQ uygulamalarının tasarımcıları, bu ortamların uyguladığı farklardan ve sınırlamalardan haberdar olmalıdır. Örneğin:

- IBM MQ , kuyruk yöneticileri arasında iletişim sağlayan olanaklar sağlar (bu, *dağıtım kuyruğu alma* olarak bilinir).
- Değişiklikleri kesinleştirme ve geri yükleme yöntemleri, toplu iş ile CICS ortamları arasında farklılık gösterir.
- IBM MQ for z/OS , çevrimiçi ileti işleme programları (MPP), etkileşimli hızlı yol programları (IFP ' ler) 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 1060](#) and [“z/OS toplu iş konuları” sayfa 769](#) for z/OS batch programs.
- Tek bir z/OS sisteminde birden çok IBM MQ for z/OS örneği bulunsa da, 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 bağlanabilirler.
- IBM MQ for z/OS , yerel kuyrukların bir grup kuyruk yöneticisi tarafından paylaşılmasını sağlayarak, üretilen iş hacminden ve kullanılabilirliğin artırılmasına olanak tanır. Bu tür kuyruklar *paylaşılan kuyruklar* olarak adlandırılır ve kuyruk yöneticileri, aynı paylaşılan kuyruklardaki iletileri işleyebilen bir *kuyruk paylaşım grubu* oluşturur. Toplu iş uygulamaları, bir kuyruk paylaşım grubu içindeki birkaç kuyruk yöneticisinden birine, belirli bir kuyruk yöneticisi adı yerine kuyruk paylaşım grubu adı belirtilerek bağlanabilir. Bu, *grup toplu eklemeye* da daha basit *grup eklemeye* olarak bilinir. Bkz. [Paylaşılan kuyruklar ve kuyruk paylaşım grupları](#).

### z/OS

Desteklenen ortamlar ve sınırlamaları arasındaki farklar, [“Uygulamaların IBM MQ for z/OS üzerinde kullanılması ve yazılması” sayfa 922](#) ' te daha ayrıntılı olarak açıklanmıştır.



## İlgili kavramlar

[“Uygulama geliştirme kavramları” sayfa 7](#)

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.

[“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#)

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğinize karar verdiğinizde, IBM MQ tarafından sunulan özelliklerin nasıl kullanılacağına karar vermeniz gerekir.

[“Kuyruğa alma için bir yordamsal uygulama yazma” sayfa 759](#)

Kuyruk yöneticisi, yayınlama/abone olma, nesnelere açma ve kapama nesnelere bağlanma ve bağlantıyı kesme, kuyruğa alma ve bağlantı kesme hakkında bilgi edinmek için bu bilgileri kullanın.

[“İstemci yordamsal uygulamaları yazılıyor” sayfa 945](#)

Bir yordamsal dil kullanarak IBM MQ üzerinde istemci uygulamaları yazmak için bilmeniz gerekenler.

[“kullanmaIBM MQ classes for JMS” sayfa 77](#)

IBM MQ classes for Java Message Service (IBM MQ classes for JMS), IBM MQ ile birlikte verilen JMS sağlayıcısıdır. javax.jms paketinde tanımlanan arabirimlerin yanı sıra, IBM MQ classes for JMS , JMS API ' ye iki uzantı kümesi sağlar.

[“Component Object Model Interface olanağının kullanılması \( ActiveX için IBM MQ Otomasyon Sınıfları\)” sayfa 640](#)

The IBM MQ Automation Classes for ActiveX (MQAX) are ActiveX components that provide classes that you can use in your application to access IBM MQ.

[“kullanmaIBM MQ classes for Java” sayfa 321](#)

Java ortamında IBM MQ kullanın. IBM MQ classes for Java allow a Java application to connect to IBM MQ as an IBM MQ client, or connect directly to an IBM MQ queue manager.

[“.NET uygulamalarının geliştirilmesi” sayfa 528](#)

IBM MQ classes for .NET , .NET programlama çerçevesinde yazılmış bir programın IBM MQ ile IBM MQ MQI client arasında bağlantı kurmasını ya da bir IBM MQ sunucusuna doğrudan bağlanmasını sağlar.

[“C++ uygulamaları geliştirilmesi” sayfa 500](#)

IBM MQ provides C++ classes equivalent to IBM MQ objects and some additional classes equivalent to the array data types. Bu, MQI aracılığıyla olmayan bazı özellikleri sağlar.

[“Yordamsal uygulama oluşturulması” sayfa 1030](#)

Bir IBM MQ uygulamasını birden çok yordamsal dilden birine yazabilir ve uygulamayı farklı platformlarda çalıştırabilirsiniz.

## İlgili görevler

[“IBM MQ örnek yordamsal programlarının kullanılması” sayfa 1092](#)

Bu örnek programlar yordamsal dillere yazılır ve İletim Kuyruğu Arabirimi 'nin (MQI) tipik kullanımları gösterir. Farklı platformlardaki IBM MQ programları.

[“IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi” sayfa 1292](#)

IBM MQ için Microsoft Windows Communication Foundation (WCF) özel kanalı, WCF istemcileri ve hizmetleri arasında ileti gönderir ve iletileri 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 özneliği olarak görüntülenir.

#### Notlar:

1. IBM MQ client Daha önceki IBM MQ 9.1.2 sürümünü kullanan uygulamalar için uygulama adı, IBM MQ client tarafından otomatik olarak ayarlanır. Değeri, uygulama programlama diline ve uygulamanın çalıştığı platforma bağlıdır. Daha fazla bilgi için bkz. PutApplAdı .
2. **V 9.1.2** 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. Ek bilgi için Desteklenen programlama dillerinde uygulama adının kullanılması başlıklı konuya bakın.

## Uygulama örnekleri

### Multi

Bağlantılar *uygulama yönetim ortamlarına* alt bölümlere ayrılır. Bir uygulamanın eşgörünümü, o uygulama için bir 'yürütme birimi' sağlayan, yakından ilişkili bağlantılar kümesidir. 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 Etiketili 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 dengelemesi kullanılırken 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 gerçekleştirilemiyor; özellikle:

- Aynı işlemde paylaşılan bir etkileşimle farklı uygulama adları kullanılarak birden çok bağlantı kurulursa.
- Daha eski düzeydeki istemci kitaplıkları kullanılıyorsa. 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 çağırısı MQCNO\_RECONNECT\_INCOMPATIBLE ile reddedilir.

#### İlgili kavramlar

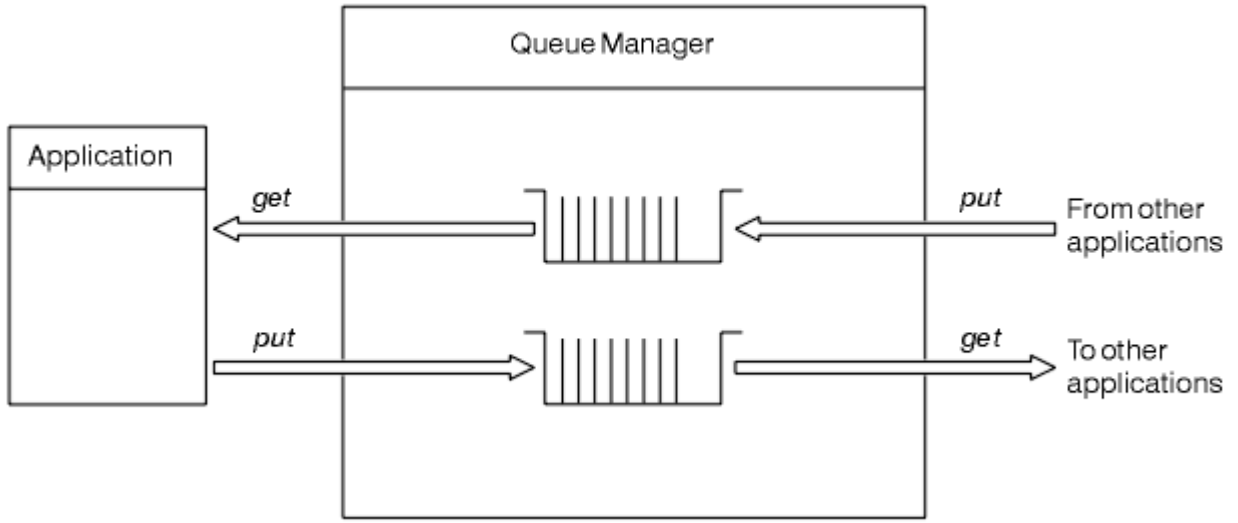
“Desteklenen programlama dillerinde uygulama adının belirtilmesi” sayfa 49

IBM MQ 9.1.2 öncesinde, Java ya da JMS istemci uygulamalarında bir uygulama adı zaten belirtebilirsiniz. IBM MQ 9.1.2 ' dan bu özellik, IBM MQ for Multiplatforms üzerindeki diğer programlama dillerine uzatılır.

## MQI kullanan uygulama programları

IBM MQ uygulama programlarının başarıyla çalıştırılabilmesi için bazı nesnelere gereksinim vardır.

Şekil 1 sayfa 11 , bir kuyruktan iletileri kaldıran, bunları işleyen ve daha sonra, aynı kuyruk yöneticisinden başka bir kuyruğa sonuç gönderen bir uygulamayı gösterir.



Şekil 1. Kuyruklar, iletiler ve uygulamalar

Uygulamalar, iletileri yerel ya da uzak kuyruklara (MQPUT kullanarak) koyabilse de, yalnızca yerel kuyruklardan doğrudan ileti 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 yerleştirdiği ikinci kuyruğun da tanımlanması gerekir.
- Uygulamanın kuyruk yöneticisine bağlanabilmesi gerekir. Bunu yapmak için IBM MQ ile bağlantı oluşturulmalıdır. Bkz. [“Yordamsal uygulama oluşturulması” sayfa 1030.](#)
- İletileri ilk sıraya koyan uygulamalar aynı zamanda bir kuyruk yöneticisine bağlanmalıdır. Bunlar uzaksa, iletim kuyrukları ve kanallarıyla da ayarlanmalıdır. Sistemin bu bölümü [Şekil 1 sayfa 11](#) içinde gösterilmez.

## Birden çok IBM MQ kuyruk yöneticisine bağlanmak için istemci bağlantılarının kullanılması

İstemci kanal tanımlama çizelgelerini (CCDTs), bağlantı adı listelerini (CONNNAME listesi), yük dengelemeyi ve kod sınırlı kod öbeklerini, belirli bir gereksinime ilişkin her seçeneğin avantajlarıyla ve dezavantajlarıyla birbirine bağlamak için kullanabileceğiniz kod öbeklerini nasıl kullanabilirsiniz.

Aşağıdaki metinde:

### CCDT-çok QMGR

Aynı grubu içeren birden çok istemci bağlantısı (CLNTCONN) içeren bir CCDT dosyası, farklı CLNTCONN girişlerinin farklı kuyruk yöneticilerine çözümlediği kuyruk yöneticisi adı istemci bağlantısı (QMNAME CLNTCONN) özniteliği anlamına gelir.

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

CCDT çoklu kuyruk yöneticisi yaklaşımını seçerseniz, girişlerin önceliklerini mi, yoksa rasgele iş yükü yönetimi mi (WLM) mi seçmeniz gerektiğini seçmeniz gerekir:

### Önceliklendirilmiş

Son iyi bağlantıyı anımsamak için CLNTWGHT (1) ve AFFINITY (TERCIH EDİLEN) öznitelikleriyle alfabetik olarak sıralanmış birden çok giriş kullanın.

### **Rasgele hale getirilmiş**

CLNTWGHT (1) ve AFFINITY (NONE) özniteliklerini kullanın. CLNTWGHT ' yi ayarlayarak, farklı ölçeklenmiş IBM MQ sunucularında WLM ağırlıklandırmasını ayarlayabilirsiniz.

**Not:** Kanallar arasında CLNTWGHT ' de büyük farklar oluşmasını önlemeniz gerekir.

### **Yük dengeleyici**

Birden çok IBM MQ kuyruk yöneticisinin TCP/IP dinleyicilerinin kapı izlemesi ile yapılandırılmış bir Sanal IP adresi (VIP) olan bir ağ aygıtı anlamına gelir. VIP ' nin ağ aygıtında nasıl yapılandırıldığı, kullanmakta olduğunuz ağ aracına bağlıdır.

Aşağıdaki seçenekler, yalnızca iletileri gönderen uygulamalar ya da zamanuyumlu istek ve yanıt ileti alışverişi başlatma ilişkisi içerir. Bu iletiler ve isteklere hizmet veren uygulamalar için dikkat edilmesi gereken noktalar (örneğin, dinleyiciler tamamen ayrı) ve "İleti dinleyicisini bir kuyruğa bağlama" (TBD) içinde ayrıntılı olarak ele alınmıştır.

## **Tek bir kuyruk yöneticisine bağlanan var olan uygulamalar için gereken kod değişikliği ölçekleri**

### **CONNNAME listesi, CCDT çoklu QMGR ve Yük dengeleyici**

MQCONN ("QMNAME") MQCONN 'a ("\*QMNAME")

Kuyruk yöneticisi adı, Java Enterprise Edition (EE) uygulamaları için Java Naming and Directory Interface (JNDI) yapılandırılabilir. Ters durumda, bu bir karakter kodu değişikliği gerektirir.

Bunun, kod üzerinde olumlu bir etkisi olabilir.

### **Kod kalıntısı**

Var olan JMS ya da MQI bağlantı mantığını kod sınırlı kod öbeğiyle değiştirin.

Bu, büyük olasılıkla kod üzerinde olumsuz bir etki olabilir.

## **Farklı WLM stratejilerine ilişkin destek**

### **CONNNAME listesi**

Yalnızca öncelik sırasına göre belirlenir.

Bu, büyük olasılıkla kod üzerinde olumsuz bir etki olabilir.

### **CCDT çoklu QMGR**

Öncelik sırasına göre ya da rasgele.

Bu, kod üzerinde herhangi bir etki gösteremez.

### **Yük dengeleyici**

Tüm iletiler için her bağlantı da dahil olmak üzere, herhangi biri.

Bunun, kod üzerinde olumlu bir etkisi olabilir.

### **Kod kalıntısı**

Tüm iletiler için her ileti dahil herhangi biri.

Bunun, kod üzerinde olumlu bir etkisi olabilir.

## **Birincil kuyruk yöneticisi kullanılmazken başarımların üst yükü**

### **CONNNAME listesi**

Her zaman ilk olarak listede dener.

Bu, büyük olasılıkla kod üzerinde olumsuz bir etki olabilir.

### **CCDT çoklu QMGR**

Son iyi bağlantıyı hatırlıyor.

Bunun, kod üzerinde olumlu bir etkisi olabilir.

### **Yük dengeleyici**

Kapı izleme, hatalı kuyruk yöneticilerini önler.

Bunun, kod üzerinde olumlu bir etkisi olabilir.

### **Kod kalıntısı**

Son iyi bağlantıyı anımsayabilir ve akıllı bir şekilde yeniden deneyin.  
Bunun, kod üzerinde olumlu bir etkisi olabilir.

## **XA hareketi desteği**

### **CONNNAME listesi, CCDT çoklu QMGR ve Yük dengeleyici**

İşlem yöneticisinin, aynı kuyruk yöneticisi kaynağına yeniden bağlanan kurtarma bilgilerini saklamaya gerek vardır.

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 çözümlenmelidir.

Bu, büyük olasılıkla kod üzerinde olumsuz bir etki olabilir.

### **Kod kalıntısı**

Kod sınırlı kod öbeği, bir hareket yöneticisine ilişkin XA gereksinmelerini (örneğin, birden çok bağlantı fabrikası) karşılayabilir.

Bunun, kod üzerinde olumlu bir etkisi olabilir.

## **Farklı WLM stratejilerine ilişkin destek**

### **CONNNAME listesi**

Yalnızca DNS.

Bu, büyük olasılıkla kod üzerinde olumsuz bir etki olabilir.

### **CCDT çoklu QMGR**

DNS ve paylaşılan kütük sistemi ya da paylaşılan kütük sistemi ya da CCDT dosyası gönderme.

Bu, kod üzerinde herhangi bir etki gösteremez.

### **Yük dengeleyici**

Dinamik sanal IP adresi (VIP).

Bunun, kod üzerinde olumlu bir etkisi olabilir.

### **Kod kalıntısı**

DNS ya da tek kuyruk yöneticisi CCDT girişleri.

Bu, kod üzerinde herhangi bir etki gösteremez.

## **Planlı bakım sırasında kesintiye uğrama**

Bir kuyruk yöneticisinin planlı bakımı sırasında, son kullanıcılar tarafından görülebilen hatalar ve zamanaşımaları gibi uygulamalarda kesintiye neden olmamanız için göz önünde bulundurmanız ve planlamak istediğiniz başka bir durum vardır. Kesintilikten kaçınmak için en iyi yaklaşım, durdurulmadan önce bir kuyruk yöneticisinden tüm işleri kaldırmasıdır.

Bir istek ve yanıt senaryosu göz önünde bulundurun. Tüm uçuş isteklerinin tamamlanmasını ve yanıtların uygulama tarafından işlenmesini istiyorsanız, ancak ek bir çalışmanın sisteme gönderilmesini istemediğinizi belirleyin. Simply quiescing the queue manager does not fulfill this need, as well-coded applications receive a return code RC2161 MQRC\_Q\_MGR\_QUIESCING exception, before they receive their reply messages for in-flight requests.

Yanıt kuyruklarını hem PUT (ETKIN) hem de GET (ENABETLE) bırakırken, iş göndermek için kullanılan istek kuyruklarına PUT (DISABLE) ayarlayabilirsiniz. Bu şekilde, istek, iletim ve yanıt kuyruklarının derinliğini izleyebilirsiniz. Tümü stabilize olduğunda, bu, uçuş içi istekleri tamamlanır ya da zamanaşımına uğratır, kuyruk yöneticisini durdurabilirsiniz.

Ancak, istekte bulunan uygulamalarda iyi kodlamanın, bir iletiyi gönderme girişimi sırasında RC2051 MQRC\_PUT\_INHIBITED dönüş kodunu döndüren bir PUT (DISABLE) istek kuyruğunu işlemek için kullanılması gerekir.

IBM MQ ile bağlantı oluşturulurken ya da istek kuyruğunun açılması sırasında kural dışı durumun oluşmadığını unutmayın. Kural dışı durum yalnızca, MQPUT çağrısını kullanarak bir ileti gönderme girişiminde bulunulduğunda oluşur.

İstek ve yanıt senaryoları için bu hata işleme mantığını içeren ve uygulama ekiplerinizin gelecekte bu tür bir kod öbeğini kullanmasını isteyen bir kod parçası oluşturmak, tutarlı bir davranışa sahip uygulamalar geliştirmenize yardımcı olabilir.

## Nesne yönelimli uygulamalar

IBM MQ , JMS, Java, C + + , .NET ve ActiveX için destek sağlar. Bu diller ve çerçeveler, IBM MQ çağrıları ve yapılarıyla aynı işlevselliği sağlayan sınıfları sağlayan IBM MQ Object Model 'i kullanır.

IBM MQ Nesne Modeli 'ni kullanan bazı diller ve çerçeveler, ileti kuyruğu arabirimi (MQI) ile yordamsal dilleri kullandığınızda kullanılabilir olmayan ek işlevler sağlar.

Bu model tarafından sağlanan sınıfların, yöntemlerin ve özelliklerin ayrıntıları için bkz. [“IBM MQ Nesne Modeli” sayfa 15.](#)

### JMS

IBM MQ , Java Message Service (JMS) belirtimini gerçekleştiren sınıfları sağlar. IBM MQ classes for JMS ile ilgili ayrıntılar için bkz. [Using IBM MQ classes for JMS.](#) Hangisini kullanacağınıza karar vermenize yardımcı olması için IBM MQ classes for Java ile IBM MQ classes for JMS arasındaki farklar hakkında bilgi için bkz. [“JMS ve Java uygulamalarının geliştirilmesi” sayfa 76.](#)

IBM Message Service Client for C/C++ and IBM Message Service Client for .NET provide an application programming interface (API) called XMS that has the same set of interfaces as the Java Message Service (JMS) API. Daha fazla bilgi için bkz [“XMS .NET uygulamalarının geliştirilmesi” sayfa 581.](#)

### Java

See [IBM MQ classes for Java](#) for information about coding programs using the IBM MQ Object Model in Java. IBM , IBM MQ classes for Java 'de başka hiçbir geliştirme yapmayacaktır ve bunlar, IBM MQ 8.0' ta teslim edilen düzeyde işlevsel olarak sabitlenirler. Hangi kullanıma karar vereceğine karar vermenize yardımcı olması için IBM MQ classes for Java ile IBM MQ classes for JMS arasındaki farklar hakkında bilgi için bkz. [“JMS ve Java uygulamalarının geliştirilmesi” sayfa 76.](#)

### C++

IBM MQ provides C++ classes equivalent to IBM MQ objects and some additional classes equivalent to the array data types. Bu, MQI aracılığıyla olmayan bazı özellikleri sağlar. C + + içinde IBM MQ Nesne Modeli 'ni kullanan kodlama programlarına ilişkin bilgi için bkz. [Using C++ C/C++ için Message Service Clients ve .NET , XMS olarak adlandırılan ve Java Message Service \(JMS\) ile aynı arabirim kümesine sahip bir uygulama programlama arabirimi \(API\) sağlar.](#) API.

### .NET

IBM MQ .NET sınıflarını kullanarak .NET programlarının kodlanması hakkında bilgi için bkz. [Developing .NET applications . C/C++ için İleti Hizmeti İstemcileri ve .NET , Java Message Service \(JMS\) ile aynı arabirim kümesine sahip XMS olarak adlandırılan bir uygulama programlama arabirimi \(API\) sağlar.](#) API.

### ActiveX

IBM MQ ActiveX yaygın olarak MQAX olarak bilinir. MQAX, IBM MQ for Windows' in bir parçası olarak içerilir. ActiveX desteği, IBM WebSphere MQ 6.0 düzeyinde sabitlendi. [ActiveX Component Object Model Interface \(WebSphere MQ Automation Classes for ActiveX\)](#) içinde IBM MQ Nesne Modeli 'ni kullanarak programları kodlamaya ilişkin bilgi için.

IBM MQ 9.0' tan Microsoft Active X için IBM MQ desteği kullanımdan kaldırılmıştır. IBM MQ classes for .NET are the recommended replacement technology. Daha fazla bilgi için bkz [Developing .NET applications.](#)

### İlgili kavramlar

[“Developing MQI applications with IBM MQ” sayfa 755](#)

IBM MQ , C, Visual Basic, COBOL, Assembler, RPG, pTAL ve PL/I için destek sağlar. Bu yordamsal diller, ileti kuyruğa alma hizmetlerine erişmek için ileti kuyruğu arabirimini (MQI) kullanır.

[Teknik genel bakış](#)

[“Uygulama geliştirme kavramları” sayfa 7](#)

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz.

IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.

## **İlgili başvurular**

[Uygulama geliştirme başvurusu](#)

## **IBM MQ Nesne Modeli**

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

IBM MQ Nesne Modeli şunlardan oluşur:

- *Sınıflar* , kuyruk yöneticileri, kuyruklar ve iletiler gibi bilinen IBM MQ kavramlarını temsil eder.
- MQI çağrılarına karşılık gelen her sınıftaki *yöntemler* .
- IBM MQ nesnelерinin özniteliklerine karşılık gelen her bir sınıftaki *Özellikler* .

IBM MQ Nesne Modeli 'ni kullanarak bir IBM MQ uygulaması yaratırken, uygulamada bu sınıfların somut örnekleri yaratıyorsunuz demektir. Nesne yönelimli programlamada bir sınıfın somut örneğinin adı *nesne* olarak adlandırılır. Bir nesne yaratıldığında, nesnenin özelliklerinin değerlerini inceleyerek ya da ayarlarken (MQINQ ya da MQSET çağrısının verilmesinin eşdeğeri) ve nesne için yöntem çağrıları yaparak (diğer MQI çağrılarının verilmesiyle 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, kuyruk yöneticisine yönelik bir bağlantıyı temsil eder. Connect (), Disconnect (), Commit () ve Backout () için yöntemleri vardır (MQCONN ya da MQCONNX, MQDISC, MQCMIT ve MQBACK ' in eşdeğeri). Bir kuyruk yöneticisinin özniteliklerine karşılık gelen özellikleri vardır. Bir kuyruk yöneticisi özniteliği özelliğine erişilmesi, önceden bağlantı kurulmamışsa kuyruk yöneticisine örtük olarak bağlanır. Bir MQQueueManager nesnesinin yok olması kuyruk yöneticisinden örtük olarak bağlantıyı keser.

### **MQQueue**

MQQueue sınıfından bir nesne bir kuyruğu gösterir. Kuyruktan ve kuyruktan (MQPUT ve MQGET ile eşdeğer) iletiler () ve Get () için yöntemler içerir. Bir kuyruğun özniteliklerine karşılık gelen özellikleri vardır. Bir kuyruk özniteliği özelliğine erişilmesi ya da bir put () ya da get () yöntemi çağrısının verilmesi, kuyruğu örtük olarak açar (bu MQPEL ' in eşdeğeri). Bir MQQueue nesnesini yok etmek, kuyruğu örtük olarak kapar (MQCLOSE ' nin eşdeğeri).

### **MQTopic**

MQTopic sınıfından bir nesne bir konuyu gösterir. Bu konuya (MQPUT ve MQGET eşdeğeri) ilişkin iletiler () (yayınlama) ve Get () (alma ya da abone olma) iletileri yerleştirme (alma ya da abone olma) yöntemlerine sahiptir. Bir konunun özniteliklerine karşılık gelen özellikleri vardır. Bir MQTopic nesnesine yalnızca yayın ya da abonelik için erişilebilir, aynı anda hem eşzamanlı olarak değil hem de abonelik için erişilebilir. İleti almak için kullanıldığında, MQTopic nesnesi yönetilmeyen ya da yönetilen bir abonelik ve dayanıklı ya da dayanıklı olmayan bir abone olarak yaratılabilir. Bu farklı senaryolar için çok sayıda aşırı yüklenmiş oluşturucular sağlanır.

### **MQMessage**

MQMessage sınıfının bir nesnesi, kuyruğa konmak üzere bir iletiyi gösterir ya da kuyruktan alındı. Bir arabellek içerir ve hem uygulama verilerini, hem de MQMD ' yi sarmalayın. Bu, MQMD alanlarına karşılık gelen özellikleri ve (örneğin, dizgiler, uzun tamsayılar, kısa tamsayılar, tek baytlar) arabelleğinden ve arabellekten kullanıcı verilerini yazmanızı ve okumanızı sağlayan yöntemlere sahiptir.

### **MQPutMessageSeçenekleri**

MQPutMessageSeçenekleri sınıfından bir nesne, MQPMO yapısını gösterir. Bu, MQPMO alanlarına karşılık gelen özelliklere sahiptir.

### **MQGetMessageSeçenekleri**

MQGetMessageSeçenekleri sınıfından bir nesne MQGMO yapısını gösterir. Bu, MQGMO alanlarına karşılık gelen özelliklere sahiptir.

### **MQProcess**

MQProcess sınıfının bir nesnesi bir süreç tanımlamasını gösterir (tetikleme ile kullanılır). Bir süreç tanımlamasının özniteliklerini temsil eden özelliklere sahiptir.

**Multi**

### **MQDistributionList**

MQDistributionList sınıfının bir nesnesi bir dağıtım listesini gösterir (tek bir MQPUT ile birden çok ileti göndermek için kullanılır). MQDistributionListÖğe nesnelere bir listesini içerir.

**Multi**

### **MQDistributionListÖgesi**

MQDistributionListÖğe sınıfından bir nesne tek bir dağıtım listesi hedefini gösterir. MQOR, MQRR ve MQPMR yapılarını sarsalır ve bu yapıların alanlarına karşılık gelen özellikleri içerir.

## **nesne başvuruları**

MQI kullanan bir IBM MQ programında, IBM MQ program için bağlantı noktalarını ve nesne tanıtıcılarını döndürür.

Bu tutamaçların, sonraki IBM MQ çağrılarında parametre olarak geçirilmeleri gerekir. IBM MQ Nesne Modeli ile bu tutamaçlar uygulama programından gizlenir. Bunun yerine, bir nesne başvurusundan uygulama programına döndürülemekte olan bir sınıf sonuçlarından bir nesne yaratılması. Nesne için yöntem çağrılar ve özellik erişimleri yapılırken kullanılan bu nesne başvuru.

## **Dönüş kodları**

Bir yöntem çağrısının verilmesi ya da bir özellik değerinin ayarlanması, dönüş kodlarının belirlenmesine neden olur.

Bu dönüş kodları bir tamamlama kodudur ve bir neden kodudur ve nesnenin kendi özellikleridir. Tamamlanma kodu ve neden kodunun değerleri, nesne yönelimli ortama özgü bazı ek değerler ile, MQI için tanımlananlarla aynıdır.

## **IBM MQ ileti**

IBM MQ iletisi, ileti özellikleri ve uygulama verilerinden oluşur. İleti tanımlayıcısı (MQMD), ileti gönderme ve alma işlemi arasında bir ileti dolaşırken, uygulama verileriyle birlikte gönderilen denetim bilgilerini içerir.

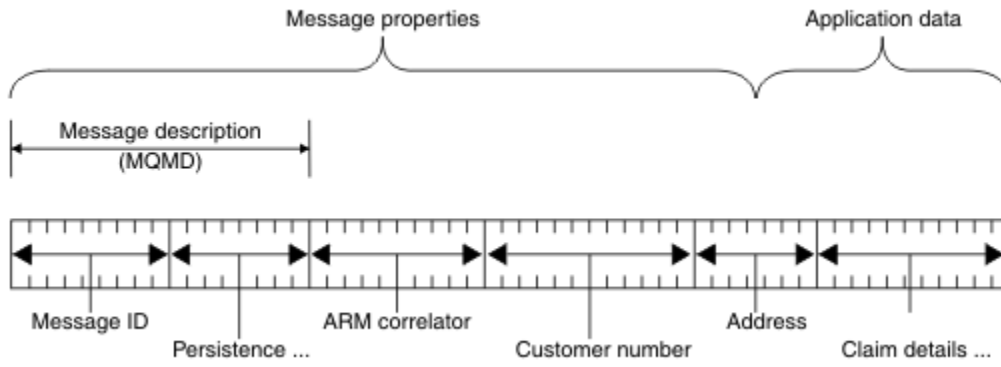
### **İletinin bölümleri**

IBM MQ iletileri iki bölümden oluşur:

- İleti Özellikleri
- Uygulama Verileri

Şekil 2 sayfa 17 , bir iletiyi gösterir ve ileti özellikleri ve uygulama verilerine mantıksal olarak nasıl bölüneceğini gösterir.





Şekil 2. İletinin gösterimi

Bir IBM MQ iletilisinde taşınan uygulama verileri, üzerinde veri dönüştürme işlemi gerçekleştirilmediği sürece bir kuyruk yöneticisi tarafından değiştirilmez. Ayrıca, IBM MQ bu verilerin içeriğine herhangi bir kısıtlama koymaz. Her iletteki verilerin uzunluğu hem kuyruğun, hem de kuyruk yöneticisinin **MaxMsgLength** özneliğinin değerini aşamaz.

**ULW** UNIX, Linux, and Windows üzerinde, kuyruk yöneticisinin **MaxMsgLength** özneliği ve kuyruk varsayılan olarak 4 MB (4 194 304 bayt) değerine ayarlanır; bu değer gerekirse en fazla 100 MB (104 857 600 bayt) değerini değiştirebilirsiniz.

**IBM i** IBM üzerinde, kuyruk yöneticisinin **MaxMsgLength** özneliği ve kuyruk varsayılan olarak 4 MB (4 194 304 bayt) değerine ayarlanır; bu değer gerekirse en fazla 100 MB (104 857 600 bayt) değerini değiştirebilirsiniz. If you are intending to use IBM MQ messages greater than 15 MB on IBM i, see “Building your procedural application on IBM i” sayfa 1036.

**z/OS** z/OS üzerinde, kuyruk yöneticisinin **MaxMsgLength** özneliği 100 MB 'de sabittir ve kuyruğun **MaxMsgLength** özneliği 4 MB' ye (4 194 304 bayt) ayarlanır; bu değer, gerekirse en fazla 100 MB ' ye kadar değişebilir.

Bazı durumlarda, iletilerinizi **MaxMsgLength** özneliğinin değerinden biraz daha kısa hale getiriniz. Daha fazla bilgi için bkz “İletinizdeki veriler” sayfa 796.

MQPOT ya da MQPUT1 MQI çağrılarını kullandığınızda bir ileti yaratırsınız. Bu çağrılara giriş olarak, denetim bilgilerini (iletinin önceliği ve yanıt kuyruğu adı gibi) ve verilerinizi ve aramayı, iletiyi bir kuyruğa yerleştirdiğinizde sağladınız. Bu çağrılar hakkında daha fazla bilgi için bkz. [MQPUT](#) ve [MQPUT1](#) .

## İleti tanımlayıcısı

İleti denetimi bilgilerine, *ileti tanımlayıcısı* nı tanımlayan MQMD yapısını kullanarak erişebilirsiniz.

MQMD yapısının tam açıklaması için bakınız: [MQMD-Message descriptor](#).

İletinin kökenine ilişkin bilgiler içeren MQMD içindeki alanların nasıl kullanılacağı hakkında bilgi için bkz. “İleti bağlamı” sayfa 44.

İleti açıklayıcısının farklı sürümleri var. İletilerin gruplanmasına ve bölümlerine ilişkin ek bilgiler (bkz. “İleti grupları” sayfa 41), ileti tanımlayıcısının (ya da MQMDE) Sürüm 2 'de sağlanır. Bu, Sürüm 1 ileti tanımlayıcısı ile aynıdır, ancak ek alanlarla aynıdır. Bu alanlar [MQMDE-Message descriptor extensioni](#) içinde açıklanmıştır.

## İleti tipleri

IBM MQ tarafından tanımlanan dört tip ileti vardır.

Bu dört ileti şunlardır:

- [Veri Paketi](#)

- [İstek iletileri](#)
- [Yanıt iletileri](#)
- [İletileri raporla](#)
  - [Rapor iletisi tipleri](#)
  - [Rapor iletisi seçenekleri](#)

Uygulamalar, kendi aralarında bilgi aktarmak için ilk üç tip ileti tipini kullanabilir. Dördüncü tip, rapor, uygulamalar ve kuyruk yöneticilerinin, bir hatanın ortaya çıkma gibi olaylarla ilgili bilgileri raporlamak için kullanabilmeleri içindir.

Her ileti tipi bir MQMT\_\* değeriyle tanıtlır. Kendi ileti tiplerinizi de tanımlayabilirsiniz. Kullanabileceğiniz değerler aralığı için bkz. [MsgType](#).

## Veri Paketleri

İletiyi alan uygulamadan (yani, iletiyi kuyruktan alır) yanıtlamak zorunda kalmadığınızda bir *veri paketi* kullanın.

Veri paketlerini kullanabilecek bir uygulama örneği, bir havaalanı salonunda uçuş bilgilerini görüntüleyen bir uygulamadır. Bir ileti, tüm uçuş bilgileri ekranının verilerini içerebilir. Bir iletinin teslim edilmemesi büyük olasılıkla önemli olmadığı için, böyle bir uygulamanın ileti için bir onay isteme olasılığı düşük. Uygulama kısa bir süre sonra bir güncelleme iletisi gönderir.

## İstek iletileri

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

İstek iletilerini kullanabilecek bir uygulama örneği, bir denetleme hesabının bakiyesini gösteren bir uygulamadır. İstek iletisi hesap numarasını içerebilir ve yanıt iletisi hesap bakiyesini içerir.

Yanıt iletinizi istek iletinizle bağlantılamak istiyorsanız, iki seçenek vardır:

- İstek iletisine, istek iletisine ilişkin bilgi koymasını sağlamaktan sorumlu istek iletisini işleyen uygulamayı yapın.
- 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 kopyalanabileceğini (varsayılan işlemin *MsgId* kopyasıdır) isteyebilirsiniz.
  - Yanıt iletisi için yeni bir *MsgId* oluşturulduğunu ya da özgün iletinin *MsgId* ' in yanıt iletisinin *MsgId* alanına kopyalanabileceğini (varsayılan işlem yeni bir ileti tanıtıcısı oluşturmasıdır) isteyebilirsiniz.

## Yanıt iletileri

Başka bir iletiyi yanıtladığınızda bir *yanıt iletisi* kullanın.

Bir yanıt iletisi oluşturduğunuzda, yanıtlamakta olduğunuz iletinin ileti tanımlayıcısında ayarlanan tüm seçeneklere saygı göstermeniz gerekir. Rapor seçenekleri, ileti tanıtıcısı (*MsgId*) ve ilinti tanıtıcısı (*CorrelId*) alanlarının içeriğini belirtir. Bu alanlar, yanıtı alan uygulamanın, yanıtı özgün isteğiyle ilintilendirmesine olanak tanır.

## Rapor iletileri

*Rapor iletileri* , bir iletiyi işlerken oluşan bir hatanın ortaya çıkma gibi olaylar hakkında uygulamaları bilgilendirir.

Bu bilgiler aşağıdaki tarafından oluşturulabilir:

- Kuyruk yöneticisi,
- Bir ileti kanalı aracısı (örneğin, iletiyi teslim edemezse) ya da

- Bir uygulama (örneğin, iletide verileri kullanamazsa).

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

### Rapor iletisi türleri

Bir kuyruğa ileti yerleştirdiğinizde, aşağıdaki bilgileri almayı seçebilirsiniz:

- *Kural dışı durum raporu iletisi*. Bu, kural dışı durum işareti ayarına sahip bir iletiye yanıt olarak gönderilir. İleti kanalı aracısı (MCA) ya da uygulama tarafından üretilir.
- *Bir süre bitimi raporu iletisi*. Bu, bir uygulamanın süre sonu eşliğine ulaşmış bir iletiyi almayı denediğini; ileti atılır olarak işaretlendi. Bu rapor tipi, kuyruk yöneticisi tarafından oluşturulur.
- *Bir geliş onayı (COA) raporu iletisi*. Bu, iletinin hedef kuyruğuna ulaştığını gösterir. Kuyruk yöneticisi tarafından üretilir.
- *Teslim edilme (COD) raporu iletisi*. Bu, iletinin alan bir uygulama tarafından alındığını gösterir. Kuyruk yöneticisi tarafından üretilir.
- *Pozitif işlem bildirim (PAN) raporu iletisi*. Bu, bir isteğin başarıyla hizmet verdiğine (yani, iletide istenen işlemin başarıyla gerçekleştirildiğini) gösterir. Bu rapor tipi uygulama tarafından oluşturulur.
- *Negatif eylem bildirim (NAN) raporu iletisi*. Bu, bir isteğin başarıyla gerçekleştirilmediğini gösterir (yani, iletide istenen işlem başarıyla gerçekleştirilmedi). Bu rapor tipi uygulama tarafından oluşturulur.

**Not:** Her rapor iletisi tipi aşağıdakilerden birini içerir:

- Özgün iletinin tamamı
- Özgün iletteki ilk 100 byte veri
- Özgün iletiden veri yok

Bir kuyruğa ileti yerleştirdiğinizde birden çok rapor iletisi tipi isteyebilirsiniz. Teslim onayı rapor iletisini ve kural dışı durum raporu ileti seçeneklerini seçerseniz, ileti teslim edilmezse, bir kural dışı durum raporu alırsınız. Ancak, yalnızca teslim onayı rapor iletisi seçeneğini belirlerseniz ve ileti teslim edilmezse, bir kural dışı durum raporu iletisi almayınız.

Belirli bir iletiyi oluşturmaya ilişkin ölçütler karşılandığında, sizin istediğiniz rapor iletileri, yalnızca sizin aldığınız iletilerdir.

### Rapor iletisi seçenekleri

Bir kural dışı durum arıktan sonra bir iletiyi *atabilirsiniz*. Atma 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, ölü mektup kuyruğuna giden ileti sayısını azaltabilirsiniz. Ancak, yalnızca veri paketi iletileri göndermediği sürece başvurunuzun döndürülen iletile uğraşmak zorunda olduğunuz anlamına gelir. Bir kural dışı durum raporu iletisi oluşturulduğunda, özgün iletinin kalıcı olarak kalıcılığını devralır.

Bir rapor iletisi teslim edilemezse (kuyruk doluysa, örneğin), rapor iletisi ölü harf kuyruğunda yerleştirilir.

Bir rapor iletisi almak istiyorsanız, *ReplyToQ* alanında yanıtınızın 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 tanımlayıcısında (*MQMD*) diğer rapor seçeneklerini de kullanabilirsiniz:

- Özgün iletinin *MsgId* ya da *CorrelId* 'inin rapor iletisinin *CorrelId* alanına kopyalanması için istekte bulunabilirsiniz. Varsayılan işlem, ileti tanıtıcısını kopyalamandır. *MQRO\_COPY\_MSG\_ID\_TO\_CORRELID* iletisini kullanın; bu ileti, yanıtı ya da rapor iletisini özgün iletiyle ilintilendirmek için ileti göndericisini etkinleştirir. Yanıtın ya da rapor iletisinin ilinti tanıtıcısı, özgün iletinin ileti tanıtıcısıyla aynı.

- Rapor iletisi için yeni bir *MsgId* oluşturulduğunu ya da özgün iletinin *MsgId* 'inin rapor iletisinin *MsgId* alanına kopyalanabileceğini isteyebilirsiniz. Varsayılan işlem, yeni bir ileti tanıtıcısı oluşturmandır. MQRO\_NEW\_MSG\_ID 'yi kullanın; sistemdeki her iletinin farklı bir ileti tanıtıcısı olmasını güvenceye alır ve sistemdeki diğer tüm iletilerden ayırt edilemez bir şekilde ayırt edilebilir.
- Özel uygulamaların MQRO\_PASS\_MSG\_ID ya da MQRO\_PASS\_COREL\_ID kullanması gerekebilir. Ancak, kuyruktan iletileri okuyan uygulamayı, örneğin, kuyruğun aynı ileti tanıtıcısına sahip birden çok ileti içerdiğini doğrulamak için kuyruktan okuyan bir uygulamayı tasarlamamız gerekir.

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

İstekte bulunanın uygulaması ile sunucu uygulaması arasında aracı olarak işlev görmekte olan uygulamalar, bu işaretlerin ayarlarını denetlemesine gerek yoktur. This is because these applications typically need to forward the message to the server application with the *MsgId*, *CorrelId*, and *Report* fields unchanged. Bu, sunucu uygulamasının, yanıt iletisinin *CorrelId* alanındaki özgün iletiden *MsgId* dosyasını kopyalamasına olanak sağlar.

Bir iletiyle ilgili rapor oluştururken, sunucu uygulamalarının bu seçeneklerden herhangi birinin ayarlanmış olup olmadığını test etmesi gerekir.

Rapor iletilerinin nasıl kullanılacağı hakkında daha fazla bilgi için [Rapor](#)'a bakın.

Raporun niteliyi belirtmek için, kuyruk yöneticileri bir dizi geribildirim kodu kullanır. Bu kodları, bir rapor iletisinin ileti tanımlayıcısının *Feedback* alanına koydular. Queue managers can also return MQI reason codes in the *Feedback* field. IBM MQ , uygulamaların kullanması için bir geribildirim kodu aralığı tanımlar.

Geribildirim ve neden kodlarıyla ilgili daha fazla bilgi için bkz. [Feedback](#).

Geribildirim kodu kullanabilecek bir programa örnek olarak, kuyruğa hizmet eden diğer programların iş yüklerini izleyen bir program örneği. Kuyruğa gönderme yapan bir programın birden çok eşgörünümü varsa ve kuyruğa gelen iletilerin sayısı artık bu durumu haklı çıkarmadıysa, bu tür bir program, programın etkinliğini sonlandırması gerektiğini belirtmek üzere hizmet veren programlardan birine (geri bildirim kodu MQFB\_QUIT ile) bir rapor iletisi gönderebilirler. (Bir izleme programı, bir kuyruğa kaç program hizmet verdiğini öğrenmek için MQINQ çağrısını kullanabilir.)

## **Multi** Raporlar ve kesimlere ayrılmış iletiler

IBM MQ for z/OS üzerinde desteklenmez.

Bir ileti bölümlendiye ve rapor oluşturulmasını istesenez, iletinin bölümlenmemesinden daha fazla rapor alabilirsiniz.

Bölümlenmiş iletilere ilişkin açıklamalar için bkz. [“İleti bölümlenmesi” sayfa 829](#).

### **IBM MQ tarafından oluşturulan raporlar için**

İletilerinizi bölümlediyseniz ya da kuyruk yöneticisinin bunu yapmasını izin verdiyseniz, iletinin tamamı için tek bir rapor almayı bekleyebileceğiniz tek bir vaka vardır. Bu, yalnızca COD raporlarını istediğinizde ve uygulama alma işlemi sırasında MQGMO\_COMPLETE\_MSG belirtmeniz yer alıyor.

Diğer durumlarda, uygulamanızın çeşitli raporlarla başa çıkabilmek için hazırlanması gerekir; genellikle her bir kesim için bir tane olmalıdır.

**Not:** İletilerinizi bölümlediyseniz ve döndürülebilme için özgün ileti verilerinin yalnızca ilk 100 baytına gereksinim duyarsanız, 100 ya da daha fazla görel konumu olan kesimler için veri olmadan rapor isteyecek rapor seçeneklerinin ayarını değiştirin. Bunu yapmazsanız ve her bir kesim 100 baytlık veri istemesi için ayarı bırakırsanız ve rapor iletilerini tek bir MQGET ile, MQGMO\_COMPLETE\_MSG belirtilerek alırsınız. Raporlar, her uygun görel konumda 100 baytlık okuma verisi içeren büyük bir iletiye toplanma yapar. Bu durumda, büyük bir arabelleğe gereksinim duyarsanız ya da MQGMO\_ACCEPT\_TRUNCATED\_MSG belirtmeniz gerekir.

## Uygulamalar tarafından oluşturulan raporlar için

Uygulamanız raporlar oluşturduysa, her zaman özgün ileti verilerinin başlangıcındaki mevcut olan IBM MQ üstbilgilerini rapor iletisi verisine kopyalayın.

Daha sonra, rapor iletisi verilerine özgün ileti verilerinin (ya da genellikle içereceği diğer miktarı) 100 byte 'ı ya da tümünü ekleyin.

MQMD ' den başlayarak ve var olan üstbilgiler arasında devam eden ardışık biçim adlarına bakarak kopyalanması gereken IBM MQ üstbilgilerini tanıyabilirsiniz. Aşağıdaki Format adları şu IBM MQ üstbilgilerini gösterir:

- MQMDE
- MQDLH
- MQXQH
- MQIH
- MQH\*

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

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

Bir Sürüm 1 MQMD kullanıyorsanız ve bir kesime ya da bir gruptaki bir iletiye ya da bölümlenmeye izin verilmeye izin verilen bir iletiye bildiriyorsanız, rapor verileri bir MQMDE ile başlamalıdır. Set the *OriginalLength* field to the length of the original message data excluding the lengths of any IBM MQ headers that you find.

## Raporlar alınıyor

COA ya da COD raporları için, MQGMO\_COMPLETE\_MSG ile sizin için yeniden derlemelerini isteyebilirsiniz.

Bir MQGMO\_COMPLETE\_MSG içeren bir MQGET, yeterli sayıda rapor iletisi (tek tip, örneğin COA ve aynı *GroupId*le birlikte), tek bir tam özgün iletiyi göstermek için kuyruğunda hazır olduğunda memnun olur. Bu, rapor iletileri tamamen özgün verileri içermese de geçerlidir; her bir rapor iletisinde *OriginalLength* alanı, verinin kendisi mevcut olmasa bile, o rapor iletiyle temsil edilen özgün verilerin uzunluğunu verir.

Bu tekniği, kuyruğunda (örneğin, hem COA hem de COD gibi) birkaç farklı rapor tipi olsa da, MQGMO\_COMPLETE\_MSG ile bir MQGET işlemi yalnızca aynı *Feedback* kodlarına sahip olduğunda iletileri yeniden birleştirdiği için kullanabilirsiniz. Ancak, genellikle bu tekniği kural dışı durum raporları için kullanamazsınız; çünkü, genel olarak, bunlar farklı *Feedback* kodlarına sahiptir.

Bu tekniği, iletinin tamamının geldiğine ilişkin olumlu bir gösterge elde etmek için kullanabilirsiniz. Ancak, bazı kesimlerin başkaları tarafından bir özel durum (ya da süre bitimi) oluşturabilirken bazı kesimlerin gelmesi olasılığına karşı en çok ihtiyaç duyarsanız, bu durumda bir süre daha gerekir. Bu durumda MQGMO\_COMPLE\_MSG kullanamazsınız; genel olarak, farklı bölümler için farklı *Feedback* kodları alabilirsiniz ve bir bölüm için birden çok rapor alabilirsiniz. Ancak, MQGMO\_ALL\_SEGMENTS\_AVALABILIR seçeneğini kullanabilirsiniz.

Bunun için izin vermek için raporları geldikleri gibi almanız ve özgün iletiye ne olduğu ile ilgili uygulamanıza bir resim oluşturmanız gerekebilir. Rapor iletisinde *GroupId* alanını, özgün iletinin *GroupId* ile ilintilendirmek için rapor iletisinde ve her bir rapor iletisinin tipini tanımlamak için *Feedback* alanını kullanabilirsiniz. Bunu yapma şekliniz, uygulama gereksinimlerinize bağlıdır.

Bir yaklaşım aşağıdaki gibidir:

- COD raporları ve kural dışı durum raporları isteyin.
- Belirli bir süreden sonra, MQGMO\_COMPLETE\_MSG kullanılarak bir COD raporlarının eksiksiz bir kümesinin alınıp alınmadığını denetleyin. Böyle bir durumda, uygulamanız iletinin tamamının işlendiğini bilir.

- Bu ileti yoksa ve bu iletiyle ilgili kural dışı durum raporları varsa, sorunu bölümlere ayrılmamış iletiler için kullanın, ancak bir noktada artık artık bölümleri temizlediğinizden emin olun.
- Herhangi bir türde rapor olmayan bölümler varsa, özgün kesimler (ya da raporlar), bir kanalın yeniden bağlanmasını bekliyor olabilir ya da ağ bir noktada aşırı yüklenmiş olabilir. Herhangi bir kural dışı durum raporu alınmadıysa (ya da yalnızca geçici olduğunu düşünüyorsanız), uygulamanızın biraz daha beklemesine izin verebileceğinizi düşünebilirsiniz.

Daha önce olduğu gibi, bu durum, bölümlenmemiş iletiler ile çalışırken sahip olduğunuz dikkate alınacak hususlara benzer; ancak, artık yetim kesimlerinin temizlenme olasılığını da göz önünde bulundurmanız gerekir.

Özgün ileti kritik değilse (örneğin, bir sorguysa ya da daha sonra yinelenebilecek bir iletiyse), artık kesimlerin kaldırılmasını sağlamak için bir süre bitimi ayarlayın.

## Arka düzey kuyruk yöneticileri

Bir rapor, bölümlenmeyi destekleyen bir kuyruk yöneticisi tarafından oluşturulduğunda, ancak segmentasyonu desteklemeyen bir kuyruk yöneticisinde alındığında, MQMDE yapısı (reportile temsil edilen *Offset* ve *OriginalLength* 'yi tanımlar) rapor verilerine her zaman sıfır, 100 bayt ya da iletteki özgün verilerin yanı sıra dahil edilir.

Ancak, bir iletinin bir bölümü, segmentasyonu desteklemeyen bir kuyruk yöneticisinde geçerse, orada bir rapor oluşturulduysa, özgün iletteki MQMDE yapısı tamamen veri olarak işlenir. Bu nedenle, özgün verilerin sıfır byte 'ı istendiye, rapor verisine dahil edilmez. MQMDE olmadan, rapor iletisi yararlı olmayabilir.

Bir iletinin arka düzey kuyruk yöneticisi aracılığıyla seyahat edebileceğinden emin olmak için, raporlarda en az 100 bayt veri isteyin.

## İleti denetim bilgileri ve ileti verileri 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 dizilimi alanlarındaki denetim bilgilerinin, kuyruk yöneticisi tarafından kullanılan karakter kümesinde olması gerekir.

Kuyruk yöneticisi nesnesinin **CodedCharSetId** özniteliği bu karakter kümesini tanımlar. Denetim bilgileri bu karakter kümesinde olmalıdır; çünkü, uygulamalar bir kuyruk yöneticisinden diğerine iletileri aktarırken, iletileri ileten ileti kanalı araçları, hangi veri dönüştürmenin gerçekleştirileceğini saptamak için bu özniteliğin değerini kullanır.

### İleti verilerinin biçimi

Aşağıdakilerden herhangi birini belirleyebilirsiniz:

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

Bunu yapmak için şu alanları kullanın:

#### **Format**

Bir iletinin alıcısına, iletindeki uygulama verilerinin biçiminin bir ileti olduğunu gösterir.

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

Kuyruk yöneticisinin adları MQ(örneğin, MQFMT\_STRING) ile başlayan *yerleşik biçimler* sayısı. Bunlar gereksinimleriniz karşılamazsa, kendi biçimlerinizi ( *kullanıcı tanımlı biçimler* ) tanımlayabilir, ancak bunlar için MQ ile başlayan adları kullanmamalısınız.

Kendi biçimlerinizi yarattığınızda ve kullandığınızda, iletiyi MQGMO\_CONVERT kullanarak almak için bir veri dönüştürme çıkışı yazmanız gerekir.

### **CodedCharSetId**

Bu, iletteki karakter verilerinin karakter kümesini tanımlar. Bu karakter kümesini kuyruk yöneticisinde ayarlamak istiyorsanız, bu alanı MQCCSI\_Q\_MGR ya da MQCCSI\_INHERIT değişmezine ayarlayabilirsiniz.

Bir kuyruktan ileti aldığınızda, *CodedCharSetId* alanının değerini, uygulamanızın beklediği değerle karşılaştırın. İki değer farklı olursa, iletteki herhangi bir karakter verilerini dönüştürmeniz ya da bir veri dönüştürme iletisi çıkışı kullanmanız gerekebilir.

### **Encoding**

Bu, ikili tamsayıları, paketlenmiş ondalık tamsayıları ve kayan nokta numaralarını içeren sayısal ileti verilerinin biçimini açıklar. Genellikle, kuyruk yöneticisinin üzerinde çalıştığı belirli bir makineye göre kodlanır.

Bir kuyruğa ileti koyduğunuzda, genellikle *Encoding* alanında MQKEN\_NATIVE değişmezini belirtiyorsunuz. Bu, ileti verilerinizin kodlamasının, uygulamanızın çalıştığı makineyle aynı olduğu anlamına gelir.

Bir kuyruktan ileti aldığınızda, ileti tanımlayıcısındaki *Encoding* alanının değerini, makinenizdeki sabit MQENC\_NATIVE değişmezinin değeriyle karşılaştırın. İki değer farklı olursa, iletteki sayısal verileri dönüştürmeniz ya da varsa, veri dönüştürme iletisi çıkışı kullanmanız gerekebilir.

### **Uygulama verileri dönüştürme**

Uygulama verilerinin karakter kümesine dönüştürülmesine ve farklı platformların ilgilendiği başka bir uygulama için gereken kodlamaya dönüştürülmesi gerekebilir.

Bu değer, gönderme kuyruğu yöneticisinde ya da alıcı kuyruk yöneticisinde dönüştürülebilmektedir. Yerleşik biçimlerin kitaplığı gereksinimleriniz karşılamazsa, kendi biçiminizi tanımlayabilirsiniz. Dönüştürme tipi, ileti tanımlayıcısının biçim alanında, MQMD ' de belirtilen ileti biçimine bağlıdır.

**Not:** Belirtilen MQFMT\_NONE ile iletiler dönüştürülmedi.

### **Gönderme kuyruğu yöneticisinde dönüştürme**

Uygulama verilerini dönüştürmek için gönderilen ileti kanalı aracısına (MCA) gerek duyarsanız, CONVERT kanalı özniteliğini YES olarak ayarlayın.

Dönüştürme, belirli yerleşik biçimler için gönderme kuyruğu yöneticisinde ve uygun bir kullanıcı çıkışı sağlandıysa, kullanıcı tanımlı biçimler için gerçekleştirilir.

### **Yerleşik biçimler**

Bu üyeler şunlardır:

- Tüm karakterler (MQFMT\_STRING biçim adını kullanarak) olan iletiler
- IBM MQ tanımlı iletiler (Programlanır Komut Biçimleri gibi)

IBM MQ , denetim iletileri ve olayları için Programlanır 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 dönüştürmeden yararlanabilirsiniz.

Kuyruk yöneticisi yerleşik biçimlerinin tümünün MQFMT ile başlayan adları var. Bunlar listelenir ve [Biçim](#) alanında açıklanmaktadır.

## Uygulama tanımlı biçimler

Kullanıcı tanımlı biçimler için, uygulama verileri dönüştürmesi bir veri dönüştürme çıkış programı tarafından gerçekleştirilmelidir (daha fazla bilgi için bkz. "[Veri dönüştürme çıkışları yazılıyor](#)" sayfa 1014). İstemci-sunucu ortamında, çıkış sunucuya yüklenir ve dönüştürme işlemi burada gerçekleşir.

## Alıcı kuyruk yöneticisinde dönüştürme

Uygulama iletisi verileri, hem yerleşik hem de kullanıcı tanımlı biçimler için alıcı kuyruk yöneticisi tarafından dönüştürülebilmektedir.

MQMO\_CONVERT seçeneğini belirtirseniz, dönüştürme işlemi bir MQGET çağrısının işlenmesi sırasında gerçekleştirilir. Ayrıntılar için [Seçenekler](#)' e bakın.

## Kodlanmış karakter takımları

IBM MQ ürünleri, temeldeki işletim sistemi tarafından sağlanan kodlanmış karakter takımlarını destekler.

Bir kuyruk yöneticisi yarattığınızda, kullanılan kuyruk yöneticisi kodlanmış karakter takımı tanıtıcısı (CCSID), temeldeki ortamın temelini temel alır. Karma kod sayfasıysa, IBM MQ , karma kod sayfasının SBCS kısmını kuyruk yöneticisi CCSID 'si olarak kullanır.

Genel veri dönüştürmesi için, temeldeki 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 verileri dönüştürme, biçim adları ve kullanıcı çıkışlarını dikkate almanız gerekir. Veri dönüştürme çıkışlarının çağrılmasına ve yazılmasına ilişkin bilgi için bkz. "[Veri dönüştürme çıkışları yazılıyor](#)" sayfa 1014 .

## İleti öncelikleri

İletin önceliğini sayısal bir değere ayarlayabilir ya da iletinin, kuyruğun varsayılan önceliğini almasını da belirleyebilirsiniz.

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

Kuyruğun **MsgDeliverySequence** özneliği, kuyrukta bulunan iletilerin FIFO (ilk giren, ilk çıkış) sırasıyla ya da öncelik sırası içinde FIFO ' da saklanıp saklanmayacağını belirler. Bu öznelik MQMDS\_PRIORITY değerine ayarlıysa, iletiler ileti tanımlayıcılarının *Priority* alanında belirtilen önceliğe göre kuyruğa alınır; ancak, bu, MQMDS\_FIFO olarak ayarlandıysa, iletiler kuyruğun varsayılan önceliğiyle kuyruğa alınır. Eşit önceliğe sahip iletiler geliş sırasına göre kuyrukta saklanır.

Bir kuyruğun **DefPriority** özneliği, kuyruğa konmakta olan iletiler için varsayılan öncelik değerini ayarlar. Bu değer, kuyruk yaratıldığında ayarlanır, ancak daha sonra değiştirilebilir. Diğer ad kuyrukları ve uzak kuyruklara ilişkin yerel tanımlamalar, çözümledikleri temel kuyruklardan farklı varsayılan önceliklere sahip olabilir. Çözüm yolunda birden çok kuyruk tanımlaması varsa (bkz. "[Ad çözünürlüğü](#)" sayfa 783 ), varsayılan öncelik, açık komutta belirtilen kuyruğun **DefPriority** özneliğinin değerinden (put işleminin sırasında) alınır.

Kuyruk yöneticisinin **MaxPriority** özneliğinin değeri, o kuyruk yöneticisi tarafından işlenen bir iletiye atayabileceğiniz en yüksek önceliğidir. Bu özneliğin değerini değiştiremezsiniz. IBM MQ içinde, öznelik 9 değerine sahiptir; 0 (en düşük) ile 9 (en yüksek) arasında öncelikler içeren iletiler oluşturabilirsiniz.

## İleti Özellikleri

Bir uygulamanın işlenecek iletileri seçmesine izin vermek ya da MQMD ya da MQRFH2 üstbilgilerine erişmeden bir iletiyle ilgili bilgileri almak için ileti özelliklerini kullanın. Ayrıca, IBM MQ ve JMS uygulamaları arasında iletişimi de kolaylaştırır.



İleti özelliği, metinli bir ad ve belirli bir tipteki bir değerden oluşan bir iletiyle ilişkilendirilmiş verilerdir. İleti özellikleri, yayınların konulara süzmesi ya da kuyruktan seçmeli olarak ileti almak için ileti seçicileri tarafından kullanılır. İleti özellikleri, iş verilerini ya da durum bilgilerini uygulama verilerinde saklamaya gerek kalmadan içermek için kullanılabilir. Bu veri yapılarındaki alanlar, Message Queue Interface (MQI) işlev çağrıları kullanılarak ileti özellikleri olarak erişilebildiğinden, uygulamaların MQ Message Descriptor (MQMD) ya da MQRFH2 üstbilgilerindeki verilere erişmek zorunda kalmaması gerekir.

The use of message properties in IBM MQ mimics the use of properties in JMS. Bu, özellikleri bir JMS uygulamasında ayarlayabileceğiniz ve bunları bir yordamsal IBM MQ uygulamasında ya da başka bir şekilde alabileceğiniz anlamına gelir. Bir özelliği bir JMS uygulaması için kullanılabilir yapmak için, "usr" önekini atayın; daha sonra, JMS ileti kullanıcı özelliği olarak kullanılabilir (önek olmadan). Örneğin, IBM MQ özelliği *usr.myproperty* (bir karakter dizisi) JMS uygulaması `message.getStringProperty('myproperty')` çağrısı kullanılarak JMS uygulaması tarafından erişilebilir. JMS uygulamalarının, iki ya da daha fazla U+002E (".") içerirse "usr" öneğine sahip özelliklere erişemediğine dikkat edin. karakterdir. Öneki olmayan ve U+002E olmayan bir özellik (".") karakter, "usr" öneğine sahip olduğu gibi işlem görür. Bunun tersine, bir JMS uygulamasına ayarlanan bir kullanıcı özelliği kümesine "usr" eklenerek bir IBM MQ uygulaması içinde erişilebilir. Bir MQINQMP çağrısında sorgulanacak özellik adına önek ekleyin.

### **İleti özellikleri ve ileti uzunluğu**

Bir IBM MQ kuyruk yöneticisinde herhangi bir iletiyle akabilecek özelliklerin büyüklüğünü denetlemek için kuyruk yöneticisi özneteliğini *MaxPropertiesLength* kullanın.

Genel olarak, özellikleri ayarlamak için MQSETMP kullandığınızda, özelliğin büyüklüğü, byte cinsinden özellik adının uzunluğunun yanı sıra, MQSETMP çağrısına aktarılan byte cinsinden özellik değerinin uzunluğunu da içerir. Bu, özelliğin karakter kümesi ve özelliğin Unicode 'a dönüştürülebilmesi nedeniyle hedefe iletilmesi sırasında değişebilir; bu durumda, özelliğin büyüklüğü değişebilir.

Bir MQPUT ya da MQPUT1 çağrısında, iletinin özellikleri, kuyruk ve kuyruk yöneticisi için iletinin uzunluğuna doğru sayılmaz, ancak kuyruk yöneticisi tarafından algılanan özelliklerin uzunluğuna kadar (bu iletiler, MQI çağrıları kullanılarak ayarlanmış ya da değil), özelliklerin uzunluğuna kadar sayı sayılır.

Özelliklerin büyüklüğü özellik uzunluğu üst sınırını aşarsa, bu ileti MQRC\_PROTETIES\_TOO\_BüFK ile reddedilir. Özelliklerin boyutu, gösterimine bağlı olduğundan, özellik uzunluğu üst sınırını bir brüt düzeyde ayarlamalısınız.

Arabellek özellikleri içeriyorsa, bir uygulamanın, *MaxMsgUzunluğu* değerinden daha büyük bir arabelleğe sahip bir iletiyi başarıyla yerleştirmesi mümkündür. Bunun nedeni, MQRFH2 öğeleri olarak gösterilse bile, ileti özellikleri iletinin uzunluğuna doğru sayılmaz. MQRFH2 üstbilgi alanları, özellikler uzunluğuna yalnızca bir ya da daha çok klasör varsa ve üstbilgideki her klasörde özellikler içeriyorsa, bu alanlara veri eklenir. Bir ya da daha çok klasör MQRFH2 üstbilgisinde bulunuyorsa ve herhangi bir klasör özellik içermiyorsa, MQRFH2 üstbilgi alanları ileti uzunluğuna doğru sayılır.

Bir MQGET çağrısında, iletinin özellikleri, ileti uzunluğuna kadar, kuyruk ve kuyruk yöneticisi tarafından dikkate almaz. Ancak, özellikler ayrı olarak sayıldığından, bir MQGET çağrısının döndürdüğü arabelleğin *MaxMsgUzunluğu* özneteliğinin değerinden büyük olması mümkündür.

Uygulamalarınız *MaxMsgUzunluğu* değerini sorgulamaz ve MQGET 'ı çağırılmadan önce bu büyüklükte bir arabellek ayıramaz; bunun yerine, yeterince büyük değerlendirdiğiniz bir arabellek ayırın. If the MQGET fails, allocate a buffer guided by the size of the *DataLength* parameter.

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

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

IBM WebSphere MQ 7.0' dan önceki bir kuyruk yöneticisine veri akışı sırasında, ileti tanımlayıcısında bulunan iletinin özellikleri, iletinin uzunluğuna doğru sayılır. Therefore, you should either raise the value of the *MaxMsgUzunluğu* attribute of channels going to a system earlier than IBM WebSphere MQ 7.0 as necessary, to compensate for the fact that more data might be sent for each message. Diğer bir seçenek

olarak, kuyruk ya da kuyruk yöneticisi *MaxMsgLength* değerini azaltabilir; böylece, sistemdeki genel veri düzeyi verileri aynı kalır.

İleti tanımlayıcısı ya da her ileti için uzantı dışında, ileti özellikleri için 100 MB ' lik bir uzunluk sınırı vardır.

İç gösteriminde bir özelliğin büyüklüğü, adın uzunluğunun yanı sıra, değerinin büyüklüğü artı özellik için bazı denetim verilerini içerir. İletiyeye bir özellik eklendikten sonra özellikler kümesi için bazı denetim verileri de vardır.

### **Özellik adları**

Özellik adı bir karakter dizisidir. Belirli kısıtlamalar, uzunluğuna ve kullanılacak karakter kümesine uygulanır.

Özellik adı, bağlamla sınırlı olmadığı sürece, +4095 karakterle sınırlı olmak üzere, büyük/küçük harfe duyarlı bir karakter dizisidir. Bu sınır, MQ\_MAX\_PROPERTY\_NAME\_LENGTH sabiti içinde yer alır.

Bir ileti özelliği MQI çağrısı kullanırken bu uzunluk üst sınırını aştığınızda, çağrı neden kodu MQRC\_PROPERTY\_NAME\_LENGTH\_ERR ile başarısız olur.

JMS içinde maksimum özellik adı uzunluğu olmadığından, bir JMS uygulamasının, bir MQRFH2 yapısında saklandığında geçerli bir IBM MQ özellik adı olmayan geçerli bir JMS özellik adı ayarlamasıdır.

Bu durumda, ayrıştırıldığında, özellik adının yalnızca ilk 4095 karakteri kullanılır; aşağıdaki karakterler kesilir. Bu, seçicileri kullanan bir uygulamanın bir seçim dizisiyle eşleşmemesi ya da birden çok özelliğin aynı adla kesilebileceğinden, beklemediği bir dizgiyle eşleşmesine neden olabilir. Bir özellik adı kısaltıldığında, WebSphereMQ bir hata günlüğü iletisi yayınlar.

All property names must follow the rules defined by the Java Language Specification for Java Identifiers, with the exception that Unicode character U+002E (.) is permitted as part of the name - but not the start. Java Tanıtıcılarına ilişkin kurallar, özellik adlarına ilişkin JMS belirtiminde yer alan kurallara eşittir.

Beyaz alan karakterleri ve karşılaştırma işlemleri yasaklanmıştır. Bir özellik adında gömülü boş değere izin verilir, ancak önerilmez. Gömülü boş değerler kullanırsanız, bu durum, değişken uzunluklu dizgileri belirtmek için MQCHARV yapısıyla birlikte kullanıldığında MQVS\_NULL\_TERMINATED değişiminin kullanılmasını önler.

Uygulamalar özellik adlarına dayalı iletileri seçebildiğinden ve seçiciye ilişkin ad ile seçicinin karakter kümesi arasındaki dönüştürmenin beklenmeyen bir şekilde başarısız olmasına neden olabileceğinden, özellik adlarını basit tutun.

IBM MQ özellik adları, mantıksal özellikler gruplaması için U+002E (.) karakterini kullanır. Bu işlem, ad alanını özellikler için böler. Küçük ya da büyük harflerin herhangi bir karışımında aşağıdaki örnekleri içeren özellikler, ürün tarafından kullanılmak üzere ayrılır:

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

Ad çakışmalarını önlemek için iyi bir yol, tüm uygulamaların ileti özelliklerinin İnternet etki alanı adlarıyla önlerini önlediğinden emin olun. For example, if you are developing an application using domain name ourcompany.com you could name all properties with the prefix com.ourcompany. Bu adlandırma kuralı, özelliklerin kolay seçilmesine de olanak sağlar; örneğin, bir uygulama com.ourcompany.%' un başlatılmasına başlayarak tüm ileti özelliklerini sorgulayabilir.

Özellik adlarının kullanımına ilişkin ek bilgi için [Özellik adı kısıtlamaları](#) başlıklı konuya bakın.

### Özellik adı kısıtlamaları

Bir özelliği adladığınızda, bazı kuralları izlemeniz gerekir.

Özellik adları için aşağıdaki kısıtlamalar geçerlidir:

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

JMS özellikleri için eşanlamlıları sağlayan aşağıdaki özelliklerden başka istisnalar vardır:

Özellik	Eşanlamlı
JMSCorrelationID	Kök.MQMD.CorrelId ya da jms.Cid
JMSDeliveryMode	Kök.MQMD.Persistence ya da jms.Dlv
JMSHedef	jms.Dst
JMSSüresi	Kök.MQMD.Expiry ya da jms.Exp
JMSMessageID	Kök.MQMD.MsgId
JMSönceliği	Kök.MQMD.Priority ya da jms.Pri
JMSRedird	Kök.MQMD.BackoutCount
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
JMSXAppID	Kök.MQMD.PutApplName
JMSXDeliveryCount	Kök.MQMD.BackoutCount
JMSXGroupID	Kök.MQMD.GroupId ya da jms.Gid
JMSXGroupSeq	Kök.MQMD.MsgSeqNumber ya da jms.Seq
JMSXUserID	Kök.MQMD.UserIdentifier

These synonyms allow an MQI application to access JMS properties in a similar fashion to IBM MQ classes for JMS client application. Bu özelliklerin yalnızca JMSCorrelationID, JMSReplyTo, JMSType, JMSXGroupIDve JMSXGroupSeq MQI kullanılarak ayarlanabileceği.

IBM MQ classes for JMS içinden kullanılabilir olan JMS\_IBM\_\* özelliklerinin MQI kullanılarak kullanılmadığını unutmayın. JMS\_IBM\_\* properties başvurusuna, MQI uygulamaları tarafından başka yollarla erişilebilir.

2. Alt ya da büyük harflerin herhangi bir karışımında "NULL", "TRUE", "FALSE", "NOT", "AND", "OR", "BETWELE", "LIKE", "IN", "IS" ve "ESCAPE" gibi bir özellik çağrılmamalı. Bunlar, seçim dizgilerinde kullanılan SQL anahtar sözcüklerinin adlarıdır.
3. Bir özellik adı başlangıcı "mq" "mq\_usr" küçük harf ya da büyük harf karışımında yalnızca tek bir "." karakterini içerebilir. karakter (U+002E). Çoklu "." bu önekleri içeren özelliklerde karakterlerin kullanılmasına izin verilmez.
4. İki "." Karakterler arasında başka karakterler de içermeli; sıradüzeninde boş bir noktana sahip olamazsınız. Benzer şekilde bir özellik adı "." ile bitemez. karakterini kullanın.
5. Bir uygulama "a.b" özelliğini ve sonra "a.b.c" özelliğini ayarlarsa, "b" sıradüzeninde bir değer mi, yoksa başka bir mantıksal grupta içerip içermediği belirsiz. Böyle bir hiyerarşi "karma içerik" dir ve bu desteklenmez. Karma içeriğe neden olan bir özelliği ayarlamasına izin verilmez.

Bu kısıtlamalar, geçerlilik denetimi düzeneği tarafından aşağıdaki gibi uygulanır:

- Property names are validated when setting a property using the [MQSETMP- Set message property](#) call, if validation was requested when the message handle was created . Bir özelliği doğrulama girişimi üstlenilirse ve özellik adı belirtiminde bir hata nedeniyle başarısız olursa, tamamlanma kodu MQCC\_FAILED ile başarısız olur:
  - 1-4 nedenleri için MQRC\_XX\_ENCODE\_CASE\_ONE Property\_name\_error.
  - MQRC\_MIXED\_CONTENT\_NOT\_ALLOWED, neden 5.
- Doğrudan MQRFH2 öğeleri olarak belirtilen özelliklerin adları, MQPUT çağrısıyla doğrulanmasını garantilmiyor.

#### Özellik olarak ileti tanımlayıcı alanları

Çoğu ileti tanımlayıcı alanı özellik olarak değerlendirilebilir. Özellik adı, ileti tanımlayıcısı alanının adına bir örnek eklenerek oluşturulur.

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

Özellik adı	İleti tanımlayıcı alanı
Root.MQMD.Alan	Alan

C dili bildirimindeki MQMD yapı alanları için aynı büyük-küçük harf durumu ile *Field* değerini belirtin. Örneğin, Root .MQMD .AccountingToken özellik adı, ileti tanımlayıcısının AccountingToken alanına erişir.

İleti tanımlayıcısının StructId ve Version alanlarında gösterilen sözdizimi kullanılarak erişilebilir değil.

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

İleti verileri kuyruk yöneticisi tarafından onurlandırılan bir MQMDE ile başlıyorsa, açıklanan Root .MQMD .Field gösterimi kullanılarak MQMDE alanlarına erişilebilir. Bu durumda, MQMDE alanları, bir özellikler perspektifinden MQMD ' nin mantıksal olarak kabul edilir. Bkz. [MQMDE ' ye Genel Bakış](#).

### Özellik veri tipleri ve değerleri

Bir özellik, boole, bayt dizilimi, karakter dizilimi ya da kayan noktalı ya da tamsayı olabilir. Bağlam, bağlam tarafından aksi belirtilmediği sürece, 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, ancak tanımlı bir değeri yok; boş değerli bir özeldir. Boş değerli bir özellik, bir byte özelliğinden (MQBYTE []) ya da karakter dizgisi özelliğinden (MQCHAR []) farklı, ancak değeri sıfır olan bir değeri olan boş bir değer içeriyor.

Byte dizgisi, JMS ya da XMSiçinde geçerli bir özellik veri tipi değil. You are advised not to use byte string properties in the *usr* folder.

### Kuyruklardan iletilerin seçilmesi

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

## Seçiciler

İleti seçici, bir uygulamanın ilgisini yalnızca, seçim dizgisinin temsil ettiği SQL (Yapılandırılmış Sorgu Dili) sorgusunu karşılayan özelliklere sahip iletilere kaydettirmek için kullanılan değişken uzunluklu bir dizilimdir.

## MQSUB ve MQOPEN işlev çağrılarını kullanarak seçim

MQSUB ve MQOP çağrılarını kullanarak seçim yapmak için MQCHARV tipinde bir yapı olan *SelectionString*' ı kullanıyorsunuz.

The *SelectionString* structure is used to pass a variable-length selection string to the queue manager.

Seçici dizgisiyle ilişkilendirilmiş CCSID, MQCHARV yapısının VSCCSID alanı aracılığıyla ayarlanır. Kullanılan değer, seçici dizgileri için desteklenen bir CCSID olmalıdır. Desteklenen kod sayfalarının listesi için [Kod sayfası dönüşümü](#) başlıklı konuya bakın.

Desteklenen bir IBM MQ Unicode dönüştürmesi olmayan bir CCSID ' nin belirtilmesi, MQRC\_SOURCE\_CCSID\_ERROR hatasına neden olur. Bu hata, seçici kuyruk yöneticisine (MQSUB, MQOPEN ya da MQPUT1 üzerinde) sunulmaya başlansa döndürülür.

VSCCSID alanının varsayılan değeri, seçim dizgisinin CCSID değerinin kuyruk yöneticisi CCSID değerine eşit olduğunu ya da bir istemci üzerinden bağlandıysa istemci CCSID değerinin eşit olduğunu gösterir. MQCCSI\_APPL değişmezi, derlemeden önce bir uygulama yeniden tanımlanarak geçersiz kılınabilir.

MQCHARV seçicisi bir NULL dizgisi gösteriyorsa, o ileti tüketicisi için seçim gerçekleşmez ve iletiler, bir seçici kullanılmamış gibi teslim edilir.

Bir seçim dizgisinin uzunluk üst sınırı yalnızca, *VSLength*MQCHARV alanı tarafından tanımlanabilen bir dizilimle sınırlandırılır.

Bir arabellek sağladıysanız ve VSBufSize içinde artı bir arabellek uzunluğu varsa, MQSO\_RESUME abone olma seçeneğini kullanarak bir MQSUB çağrısından çıkışta SelectionString döndürülür. Bir arabellek sağlamadıysanız, MQCHARV ' ın VSLecgth alanında yalnızca seçim dizgisinin uzunluğu döndürülür. Sağlanan arabellek, alanı döndürmek için gereken alandan küçükse, sağlanan arabellere yalnızca VSBufSize byte değeri döndürülür.

Bir uygulama, ilk olarak kuyruğun tanıtıcısını (MQOPEN için) ya da aboneliği kapatmaksızın (MQSUB için) bir seçim dizgisini değiştiremiyor. Daha sonra, sonraki bir MQOPED ya da MQSUB çağrısında yeni bir seçim dizgisi belirtilebilir.

### MQOPEN

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

### MQSUB

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

[Şekil 3 sayfa 30](#) , MQSUB çağrısını kullanarak seçim sürecini gösterir.

## MQOPEN

(APP 1)

ObjectName = "MyDestQ"  
hObj



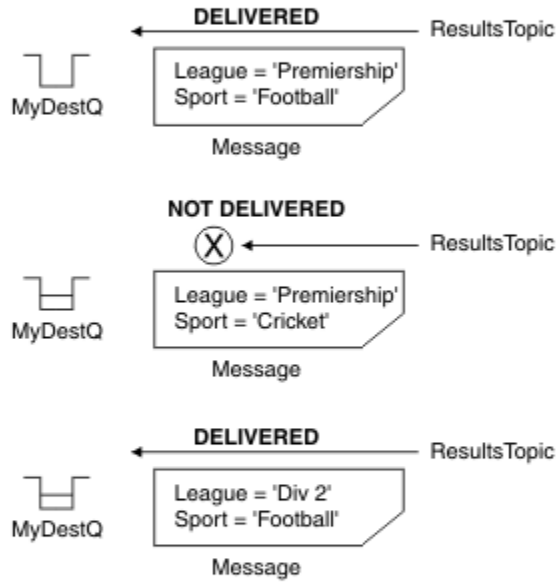
## MQSUB

(APP 1)

SelectionString = "Sport = 'Football'"  
hObj  
TopicString = "ResultsTopic"

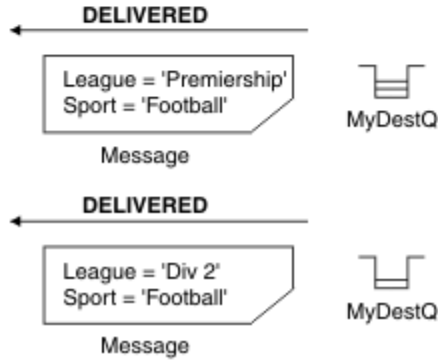


ResultsTopic



## MQGET

(APP 1) hObj



Şekil 3. MQSUB çağrısını kullanarak seçim

MQSD yapısındaki *SelectionString* alanı kullanılarak MQSUB çağrısına bir seçici aktarılabilir. MQSUB ' da bir seçicide geçirmenin etkisi, yalnızca sağlanan konuya abone olunan ve sağlanan seçim dizgisiyle eşleşen iletilerin hedef kuyrukta kullanılabilir kılındığı iletilerdir.

Şekil 4 sayfa 31 , MQOPEN çağrısını kullanarak seçim işlemini gösterir.

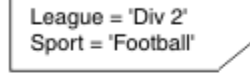
## MQOPEN

(APP 1)

SelectorString = "League = 'Premiership'"  
ObjectName = "SportQ"  
hObj

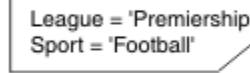


← MQPUT Application 2



Message

← MQPUT Application 2

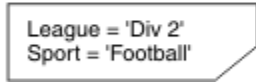


Message

## MQGET

(APP 1) hObj

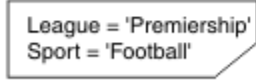
NOT DELIVERED



Message



← DELIVERED



Message



MQRC\_NO\_MSG\_AVAILABLE



Şekil 4. MQOPER çağrısını kullanarak seçim

MQOD yapısındaki *SelectionString* alanını kullanarak, MQOPEN çağrısına bir seçici aktarılabilir. MQOPER çağrısında bir seçicide geçirmenin etkisi, yalnızca açılan kuyrukta bulunan ve bir seçiciyle eşleşen iletilerin ileti tüketicisine teslim edilmeleridir.

Bir uygulamanın yalnızca bir seçiciyle eşleşen bir kuyruktaki iletileri almayı seçebileceği noktadan noktaya iletişim kutusu için, MQOPEN çağrısında seçici için ana kullanım kullanılır. Önceki örnek, iki iletinin MQOPER tarafından açılan bir kuyruğa yerleştirildiği, ancak bir seçiciyle eşleşen tek bir senaryonun yalnızca biri tarafından alındığı basit bir senaryoya yol açar.

İzleyen MQGET çağrılarının MQRC\_NO\_MSG\_AVAILABLE ile, belirtilen seçiciyle eşleşen kuyrukta başka ileti olmadığını belirtmesine dikkat edin.

### İlgili kavramlar

“Seçim dizgisi kuralları ve kısıtlamaları” sayfa 38

Seçim dizgilerinin, seçicileri kullanırken olası sorunları önlemek için nasıl yorumlanır ve karakter kısıtlamaları olduğunu öğrenmek için kendinizi bu kurallara uygun bir şekilde öğrenin.

## Seçim davranışı

IBM MQ seçim davranışına genel bakış.

MQMD yapıldıysa, bir MQMDE yapısındaki alanlar, MQMD ' ye karşılık gelen ileti tanımlayıcısı özelliklerine ilişkin ileti özellikleri olarak kabul edilir:

- MQFMT\_MD\_EXTENSION biçimi var
- Hemen ardından geçerli bir MQMDE yapısı var
- Sürüm bir ya da yalnızca iki alan için varsayılan sürüm olan iki alan içerir

Herhangi bir ileti özelliği gerçekleşmeden önce bir seçim dizisinin TRUE ya da FALSE olarak çözülebilmesi mümkündür. Örneğin, seçim dizisi "TRUE <> FALSE" olarak ayarlandıysa, bu durum söz sahibi olabilir. Bu tür erken değerlendirmenin, yalnızca seçim dizisinde ileti özelliği başvuruları olmadığında gerçekleşeceği garanti edilir.

Bir seçim dizisi, herhangi bir ileti özelliği dikkate alınmadan önce TRUE (Doğru) değerine çözümlerse, tüketici tarafından abone olunan konuya yayınlanan tüm iletiler teslim edilir. Bir seçim dizisi, herhangi bir ileti özelliği dikkate alınmadan önce FALSE değerine çözümlerse, seçiciyi sunan işlev çağrısında MQRC\_SELECTOR\_ALWAYS\_FALSE ve tamamlanma kodu MQCC\_FAILED bir neden kodu döndürülür.

Bir ileti ileti özelliği içermiyorsa (üstbilgi özellikleri dışında), seçim için uygun olabilir. Bir seçim dizisi var olmayan bir ileti özelliğine başvuruyorsa, bu özelliğin boş değer (NULL) ya da 'Unknown ' (Bilinmiyor) değerine sahip olduğu varsayılır.

Örneğin, bir ileti yine de 'Color IS NULL ' gibi bir seçim dizisini karşılayabilir; burada 'Color ' , iletide bir ileti özelliği olarak var olmaz.

Seçim yalnızca, genişletilmiş bir ileti seçimi sağlayıcısı yoksa, iletinin kendisi değil, yalnızca bir iletiyle ilişkili özelliklerde gerçekleştirilebilir. Yalnızca genişletilmiş bir ileti seçimi sağlayıcısı varsa, ileti bilgi yükünde seçim gerçekleştirilebilir.

Her ileti özelliğinin, kendisiyle ilişkilendirilmiş bir tipi vardır. Bir seçim gerçekleştirdiğinizde, ileti özelliklerini test etmek için ifadelerde kullanılan değerlerin doğru tipte olduğundan emin olmanız gerekir. Tip uyumsuzluğu ortaya çıkarsa, söz konusu ifade FALSE olarak çözümlüyor.

Seçim dizisinin ve ileti özelliklerinin uyumlu tipler kullandığından emin olmak sizin sorumluluğunuzda.

Seçim ölçütleri, etkin olmayan dayanıklı aboneler adına uygulanmaya devam eder, böylece yalnızca başlangıçta sağlanan seçim dizisiyle eşleşen iletiler saklanır.

Kalıcı abonelik alter (MQSO\_ALTER) ile sürdürüldüğünde seçim dizileri değiştirilemez. Bir kalıcı abone etkinliği devam ettiğinde farklı bir seçim dizisi sunulursa, uygulamaya MQRC\_SELECTOR\_NOT\_ALTERABLE değeri döndürülür.

Bir kuyruktan seçim ölçütlerine uygun bir ileti yoksa, uygulamalar MQRC\_NO\_MSG\_AVAILABLE dönüş kodunu alır.

Bir uygulama, özellik değerlerini içeren bir seçim dizisi belirttiyse, yalnızca eşleşen özellikleri içeren iletiler seçime uygun olur. Örneğin, bir abone "a = 3" seçim dizisini, hiçbir özellik içermeyen bir ileti yayınlanıyor ya da 'a' var olmayan ya da 3 'e eşit olmayan özellikler içeriyor. Abone, hedef kuyruğuna bu iletiyi almaz.

## İleti alışverişi başarımı

Selecting messages from a queue requires IBM MQ to sequentially inspect each message on the queue. İletiler, seçim ölçütleriyle eşleşen bir ileti bulununcaya ya da incelenecek başka ileti bulununcaya kadar incelenir. Bu nedenle, ileti seçimi derin kuyruklarda kullanılırsa, ileti alışverişi başarımı zarar görür.

Seçim JMSCorrelationID ya da JMSMessageID değerine dayalı olduğunda, ileti seçimini derin kuyruklarda eniyilemek için, formun bir seçim dizisini kullanın:

- JMSCorrelationID = 'ID:correlation\_id'
- JMSMessageID= 'ID:message\_id'



Burada:

- *correlation\_id* , standart IBM MQ ilinti tanıtıcısını içeren bir dizgidir.
- *message\_id* , standart bir IBM MQ ileti tanıtıcısı içeren bir dizgidir.

**Not:** Seçici yalnızca özelliklerden birine başvuruda bulunmalı. Bu biçimlerden birine sahip bir seçici kullanılması, JMSCorrelationID ' de seçilirken performansında önemli bir gelişme sunar ve JMSMessageID için marjinal bir başarımların artışı sağlar. Daha fazla bilgi için bkz [“JMS içindeki ileti seçicileri” sayfa 128.](#)

## Karmaşık seçicilerin kullanılması

Seçiciler birçok bileşen içerebilir; örneğin:

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

Bu tür karmaşık seçicilerin kullanımı ciddi performans etkileri ve aşırı kaynak gereksinimlerine sahip olabilir. Bu nedenle, IBM MQ sistemi, sistem kaynağı eksiklik sonuçlarıyla sonuçlanabilen aşırı karmaşık seçicileri işleyemeyerek koruyacaktır. Protection can occur on selection strings that contain more than 100 tests, or when IBM MQ detects that the limit on the size of the operating system stack is being approached. Koruma sınırlarına ulaşılmamasını sağlamak için, uygun altyapılarda birçok bileşenle seçim dizgileri kullanımını iyice denemeli ve sınamanız gerekir.

Seçicilerin performansı ve karmaşıklığı, bileşenleri birleştirmek için ek parantez kullanarak 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 38

Seçim dizgilerinin, seçicileri kullanırken olası sorunları önlemek için nasıl yorumlanır ve karakter kısıtlamaları olduğunu öğrenmek için kendinizi bu kurallara uygun bir şekilde öğrenin.

### İleti seçici sözdizimi

IBM MQ ileti seçici, SQL92 koşullu ifade sözdiziminin bir alt kümesini temel alan sözdizimine sahip bir dizgidir.

Bir ileti seçicinin değerlendirdiği sıra, öncelik düzeyi içinde soldan sağa doğru olur. Bu siparişi değiştirmek için ayraçları kullanabilirsiniz. Önceden tanımlı seçici hazır bilgileri ve işleç adları burada büyük harfle yazılır; ancak, büyük/küçük harf duyarlı değildir.

Seçici, API aracılığıyla sağlandıysa, IBM MQ , sunulma sırasında bir ileti seçicisinin sözdizimsel doğruluğunu doğrular. If the syntax of the selection string is incorrect or a property name is not valid, and an extended message selection provider is not available, MQRC\_SELECTION\_NOT\_AVAILABLE is returned to the application. Seçim dizgisinin sözdizimi yanlışsa ya da abonelik sürdürüldüğünde bir özellik adı geçerliyse, uygulamaya bir MQRC\_SELECTOR\_SYNTAX\_ERROR döndürülür. Özellik ayarlanırken özellik adı geçerlilik denetimi geçersiz kılındıysa ( MQCMHO\_VALIDATE yerine MQCMHO\_NONE belirlenerek ) ve bir uygulama daha sonra geçersiz özellik adına sahip bir ileti yerleştirdiyse, bu ileti hiçbir zaman seçilmez.

No error is returned at the time the selector is presented if IBM MQ determines that an administratively defined subscription selector is using extended message syntax, as indicated by the **DISPLAY SUB** parameter **SELTYPE** having the value UZATILDI. Bu durumda, seçim dizgisinin sözdiziminin denetlenmesi yayınlama saatine kadar ertelenir (bkz. [MQRC\\_SELECTION\\_NOT\\_AVAILABLE](#)).

Bir seçici şunları içerebilir:

- Hazır bilgiler:
  - Dizgi hazır bilgileri tek tırnak işareti içine alınır. Birbirini izleyen iki tek tırnak işareti tek tırnak işaretini temsil eder. 'Literal' ve 'literal' (hazır bilgi) örnekleri verilebilir. Java dizgi hazır bilgileri gibi, bunlar Unicode karakter kodlamasını kullanır. Bir dizgi hazır bilgisini kapatmak için çift tırnak işareti kullanamazsınız. Tek tırnak işaretleri arasında herhangi bir bayt dizisi kullanılabilir.

- Bir bayt dizilimi, çift tırnak işareti içine alınmış ve önünde 0x öneki bulunan bir ya da daha çok onaltılı karakter çiftidir. Örnekler: "0x2F1C" ya da "0XD43A". Bayt dizilimi uzunluğunun en az bir bayt olması gerekir. Bir seçici byte dizgisi MQTYPE\_BYTE\_STRING tipindeki bir ileti özelliğiyle eşleştirilirse, baştaki ya da sondaki sıfırda özel bir işlem yapılmamaktadır. Byte 'lar başka bir karakter olarak işlem görür. Endianness de dikkate alınmıyor. Seçici ve özellik byte dizgilerinin uzunluğu eşit olmalıdır ve byte sırası aynı olmalıdır.

Bayt dizilimi seçimlerine ilişkin örnekler ( *myBytes* = 0AFC23) aşağıdaki gibi olur:

- "myBytes = "0x0AFC23" " = TRUE

Aşağıdaki dizgi seçimleri eşleşmiyor:

- "myBytes = "0xAFC23" " = MQRC\_SELECTOR\_SYNTAX\_ERROR (çünkü byte sayısı birden çok iki byte değil)

- "myBytes = "0x0AFC2300" " = FALSE (karşılaştırmada sondaki sıfırın önemli olduğu için)

- "myBytes = "0x000AFC23" " = FALSE (karşılaştırmada sıfır değeri önemli olduğu için)

- "myBytes = "0x23FC0A" " = FALSE (endianness dikkate alınmadığı için)

- Onaltılı sayılar sıfır ile başlar ve ardından büyük ya da küçük harfli x olur. Hazır bilginin geri kalan kısmı bir ya da daha çok geçerli onaltılı karakter içeriyor. Örnekler: 0xA, 0xAF, 0X2020.
- 0-7 aralığındaki bir ya da daha çok basamak izleyen bir ya da daha çok basamak, her zaman sekizli bir sayının başlangıcı olarak yorumlanır. Sıfır önekli bir ondalık sayıyı bu gibi gösteremezsiniz; örneğin, 09 , 9 geçerli bir sekizli sayı olmadığı için bir sözdizimi hatası döndürür. Sekizli sayı örnekleri: 0177, 0713.
- An exact numeric literal is a numeric value without a decimal point, such as 57, -957, and +62. Tam sayısal hazır bilgi, sondaki büyük ya da küçük harflere sahip olabilir; bu, sayının nasıl depolanır ya da yorumlanabileceği etkilemez. IBM MQ supports exact numerals in the range -9, 223, 372, 036, 854, 775, 808 to 9, 223, 372, 036, 854, 775, 807.

- An approximate numeric literal is a numeric value in scientific notation, such as 7E3 or -57 . 9E2, or a numeric value with a decimal, such as 7 . , -95 . 7, or +6 . 2. IBM MQ supports numbers in the range -1 . 797693134862315E+308 to 1 . 797693134862315E+308.

Bu değer, isteğe bağlı bir işaret karakterinin ( + ya da - ) izlenmesi gerekir. Değer, tamsayı ya da kesir olmalıdır. Çok önemli bir kesirli kısım ve başında gelen bir basamak yok.

Büyük ya da küçük harfli E , isteğe bağlı bir üslubunun başlangıcını belirtir. Üstel bir ondalık radx ve üstel sayı parçası, isteğe bağlı bir işaret karakteriyle önek olarak öner olabilir.

Yaklaşık sayısal hazır bilgiler F ya da D karakteri tarafından sonlandırılabilir (büyük/küçük harfe duyarlı değildir). Bu sözdizimi, tek ya da çift duyarlıklı sayıların etiketlenilmesinin dil arası yöntemini desteklemesi için vardır. Bu karakterler isteğe bağlıdır ve yaklaşık sayısal hazır bilginin nasıl saklanmayacağını ya da işleneceğini etkilemez. Bu sayılar her zaman çift duyarlıklı olarak depolanır ve işlenir.

- Boole hazır bilgileri TRUE ve FALSE.

**Not:** Non-finite IEEE-754 representations such as NaN, +Infinity, -Infinity are not supported in selection strings. Bu nedenle, bu değerlerin bir ifadede işlenenler olarak kullanılması mümkün değildir. Negatif sıfır, matematiksel işlemler için pozitif sıfır olarak ele alınır.

- Tanıtıcılar:

Tanıtıcı, geçerli bir tanıtıcı başlangıç karakteriyle başlaması gereken, sıfır ya da daha fazla geçerli tanıtıcı parça karakteriyle başlaması gereken değişken uzunluklu bir karakter dizidir. Tanıtıcı adlarına ilişkin kurallar, ileti özelliği adlarıyla aynıdır, ek bilgi için "[Özellik adları](#)" sayfa 26 ve "[Özellik adı kısıtlamaları](#)" sayfa 27 ' a bakın.

**Not:** Yalnızca genişletilmiş bir ileti seçimi sağlayıcısı varsa, ileti bilgi yükünde seçim gerçekleştirilebilir.

Tanıtıcılar, üstbilgi alanı başvuruları ya da özellik başvurularıdır. Bir ileti seçicindeki bir özellik değerinin tipi, mümkün olduğu yerlerde sayısal promosyon gerçekleştirilmesine rağmen, özelliği ayarlamak için

kullanılan tipe karşılık gelmelidir. Tip uyumsuzluğu ortaya çıkarsa, ifadenin sonucu FALSE olur. İletide var olmayan bir özelliğe başvurulsa, değeri NULL olur.

Bir ileti seçici ifadesinde bir özellik kullanıldığında, özellikler için alma yöntemleri için geçerli olan tip dönüştürmeleri geçerli olmaz. Örneğin, bir özellik dizgi değeri olarak ayarlandıysa ve bunu sayısal değer olarak sorgulamak için bir seçici kullanırsanız, ifade FALSE değerini döndürür.

Özellik adlarıyla ya da MQMD alan adlarıyla eşlenen JMS alanı ve özellik adları, bir seçim dizgisinde de geçerli tanıtıcılardır. IBM MQ, tanınan JMS alanını ve özellik adlarını ileti özelliği değerleriyle eşler. Ek bilgi için "[JMS içindeki ileti seçicileri](#)" sayfa 128 başlıklı konuya bakın. As an example, the selection string "JMSPriority >=" selects on the Pri property found in the jms folder of the current message.

- Taşma/alt akış:

Hem ondalık hem de yaklaşık sayısal sayılar için, aşağıdaki koşullar tanımsız olur:

- Tanımlı aralık dışında bir sayı belirtme
- Taşmaya ya da alt akışa 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. Gerçek eşleşmeleri değerlendiren bir seçici; yanlış ya da bilinmeyen olarak değerlendirilen bir seçici eşleşmez.
- Aritmetik ifadeler, kendilerinden oluşur, aritmetik işlemler, tanıtıcılar (tanıtıcı değeri sayısal hazır bilgi olarak kabul edilir) ve sayısal hazır bilgiler.
- Koşullu ifadeler kendilerinden, karşılaştırma işlemlerinden ve mantıksal işlemlerden oluşur.

- İfadelerin değerlendirilen sıralamayı ayarlamak için standart destek pazarlama () desteklenir.

- Mantıksal işlemler öncelik sırasına göre: NOT, AND, OR.

- Karşılaştırma işlemleri: =, >, >=, <, <=, <> (eşit değil).

- İki baytlık dizgi yalnızca dizgiler aynı uzunlukdaysa ve byte sırası eşitse eşittir.
- Yalnızca aynı tipteki değerler karşılaştırılabilir. Bir kural dışı durum, tam sayısal değerleri ve yaklaşık sayısal değerleri karşılaştırmak için geçerlidir (gerekten tip dönüşümü, Java sayısal promosyonunun kurallarına göre tanımlanır). Farklı tipleri karşılaştırma girişiminde bulunulursa, seçici her zaman false olur.

- Dizgi ve boole karşılaştırması = ve <> ile sınırlandırılmıştır. İki dizgi, yalnızca aynı karakter dizisini içerirse eşittir.
- Aritmetik işlemler öncelik sırasına göre:
  - +, - birli.
  - \* çarpma ve / bölümü.
  - + ekleme ve - çıkarma.
  - Boş değer (NULL) olan aritmetik işlemler desteklenmez. Bunlar denenirse, tam seçici her zaman false olur.
  - Aritmetik işlemler Java sayısal promosyonu 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 (boş) ise, işlemin değeri false olur. Bir NOT BETWEEN işleminin ifadelerinden herhangi biri NULL olursa, işlemin değeri doğrudur.
- tanıtıcı [NOT] IN (string-literal1, string-literal2, ...) karşılaştırma işlecinde bir Dize ya da NULL değeri var.
  - Country IN ('UK', 'US', 'France'), 'UK' için true, 'Peru' için false. Bu, (Country = 'UK') OR (Country = 'US') OR (Country = 'France') ifadesiyle eşdeğerdir.
  - Country NOT IN ('UK', 'US', 'France') is false for 'UK' and true for 'Peru'. Bu, NOT ((Country = 'UK') OR (Country = 'US') OR (Country = 'France')) ifadesiyle eşdeğerdir.
  - Bir IN ya da NOT IN işleminin tanıtıcısı NULL ise, işlemin değeri bilinmez.
- identifier [NOT] LIKE *pattern-value* [ESCAPE *escape-character* ] karşılaştırma işleci, burada identifier bir dizgi değerine sahiptir. *pattern-value*, bir dizgi hazır bilgisidir; burada \_ herhangi bir tek karakter için, % ise herhangi bir karakter dizisi (boş sıra dahil) için durmaktadır. Diğer tüm karakterler kendileri için geçerli. The optional *çıkış karakteri* is a single character string literal that is used to escape the special meaning of the \_ and % in *örüntü-değer*. LIKE işlecinin yalnızca iki dizgi değerini karşılaştırmak için kullanılması gerekir.
  - phone LIKE '12%3', 123 ve 12993 için geçerlidir ve 1234 için false.
  - word LIKE 'l\_se', kaybedilen ve gevşek için false (yanlış) için geçerlidir.
  - underscored LIKE '\\\_%' ESCAPE '\\', \_foo için true, bariçin false.
  - phone NOT LIKE '12%3' is false for 123 and 12993 and true for 1234.
  - Bir LIKE ya da NOT LIKE işleminin tanıtıcısı NULL ise, işlemin değeri bilinmez.

**Not:** İki dizgi değerini karşılaştırmak için LIKE işleci kullanılmalıdır. Root.MQMD.CorrelId değeri, bir karakter dizisi değil, 24 byte 'lık bir bayt dizisidir. The selector string Root.MQMD.CorrelId LIKE 'ABC%' is accepted by the parser as syntactically valid, but it is evaluated to false. Bir bayt dizisini karakter dizisiyle karşılaştırırken, LIKE kullanılamaz.
- Bir NULL üstbilgisi alan değeri ya da eksik bir özellik değeri için identifier IS NULL karşılaştırma işleci testleri.
- identifier IS NOT NULL comparison operator tests for the existence of a non-null header field value or a property value.
- Boş değerler
 

NULL değerlerini içeren seçici ifadelerinin değerlendirilmesi, özet olarak SQL 92 NULL anlambilimi ile tanımlanır:

  - SQL, bir NULL değerini bilinmeyen olarak değerlendirir.
  - Bilinmeyen bir değere sahip karşılaştırma ya da aritmetik, her zaman bilinmeyen bir değer verir.
  - IS NULL ve IS NOT NULL işlemleri, bilinmeyen bir değeri TRUE ve FALSE değerlerine dönüştürür.

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

Çizelge 1. Mantık A AND Bolduğunda Boole işleci sonucunun değeri		
Operatör 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 Bolduğunda Boole işleci sonucunun değeri		
Operatör 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 Aolduğunda Boole işleci sonucunun değeri	
Operatör A	Sonuç ( NOT A)
T	F
F	T
U	U

Aşağıdaki ileti seçici, ileti tipi araba, mavi renk ve ağırlığı 2500 lbs 'den büyük olan iletileri seçer:

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

SQL, sabit ondalık karşılaştırma ve aritmetik desteklerini desteklese de, ileti seçicileri desteklemez. Bu nedenle, tam sayısal hazır bilgiler ondalıklı olmayan bu sayılarla sınırlandırılmıştır. Ayrıca, yaklaşık bir sayısal değer için alternatif bir temsil olarak bir ondalık ile sayısal değerler olmasının nedeni de bu.

SQL açıklamaları desteklenmez.

### İlgili kavramlar

[“İleti Özellikleri” sayfa 24](#)

Bir uygulamanın işlenecek iletileri seçmesine izin vermek ya da MQMD ya da MQRFH2 üstbilgilerine erişmeden bir iletiyle ilgili bilgileri almak için ileti özelliklerini kullanın. Ayrıca, IBM MQ ve JMS uygulamaları arasında iletişimi de kolaylaştırır.

## İlgili başvurular

[MsgHandle](#)

[MQBUFMMH-Arabelleği ileti tutamaçına dönüştür](#)

### *Seçim dizgisi kuralları ve kısıtlamaları*

Seçim dizgilerinin, seçicileri kullanırken olası sorunları önlemek için nasıl yorumlanır ve karakter kısıtlamaları olduğunu öğrenmek için kendinizi bu kurallara uygun bir şekilde öğrenin.

- Yayınlama/abone olma ileti alışverişi için ileti seçimi, yayıncı tarafından gönderilen iletiyle gerçekleşir. Bkz. [Seçim dizgileri](#).
- Eşdeğerlik, tek bir eşittir karakteri kullanılarak sınanmış; örneğin, a = b doğru, a == b yanlış.
- Birçok programlama dili tarafından 'eşit değil' temsil etmek üzere kullanılan bir işleç !=. Bu temsil, <> için geçerli bir eşanlamlı değildir; Örneğin a <> b geçerli; a != b geçerli değildir.
- Tek tırnak işaretleri yalnızca ' (U+0027) karakteri kullanılır. Benzer şekilde, yalnızca bayt dizilimlerini kapamak için kullanıldığında geçerli olmak üzere çift tırnak imi " (U+0022) karakteri.
- &, &&, | ve || simgeleri mantıksal bağlaç/çıkış için eşanlamlılar değildir; örneğin, a && b , a AND b olarak belirtilmelidir.
- The wildcard characters \* and ? are not synonyms for % and \_.
- 20 < b < 30 gibi bileşik ifadeler içeren seçiciler geçerli değil. Ayrıştırıcı, soldan sağa aynı önceliğe sahip işleçleri değerlendirir. Bu nedenle örnek, anlamlı gelmeyen (20 < b) < 30 olur. Bunun yerine ifade, (b > 20) AND (b < 30) olarak yazılmalıdır.
- Byte dizgileri çift tırnak içine alınmalıdır; tek tırnak işaretleri kullanılırsa, bayt dizilimi dizgi hazır bilgisi olarak alınır. 0x ' in ardından karakter sayısı (karakterlerin gösterdiği sayı değil), iki karakter olmalıdır.
- IS anahtar sözcüğü, eşittir karakteriyle eşanlamlı değil. Thus the selection strings a IS 3 and b IS 'red' are not valid. IS anahtar sözcüğü yalnızca IS NULL ve IS NOT NULL vakalarını desteklemek için vardır.

## İlgili kavramlar

[Seçim dizgileri](#)

“Seçim davranışı” sayfa 32

IBM MQ seçim davranışına genel bakış.

### *İleti seçicileri kullanırken UTF-8 ve Unicode ile ilgili dikkat edilmesi gereken noktalar*

Tek tırnak işareti içine alınmamış, bir seçim dizgisinin ayrılmış anahtar sözcüklerini oluşturan karakterler, Basic Latin Unicode 'da ( U+0000 karakterinden U+0007Farasında değişir) girilmelidir. Alfasayısal karakterlerin diğer kod noktası gösterimlerini kullanmak için geçerli değildir. Örneğin, 1 numaralı sayının Unicode 'da U+0031 olarak ifade edilmesi gerekir, Tam Genişlikli Basamak eşdeğer U+FF11 ya da Arapça eşdeğer U+0661' in kullanılması geçerli değildir.

İleti özelliği adları, herhangi bir Unicode karakter dizisi kullanılarak belirlenebilir. Message property names contained within selection strings that are encoded in UTF-8 will be validated even if they contain multi-byte characters. Çok baytlı UTF-8 için geçerlilik denetimi katıdır ve ileti özelliği adları için geçerli UTF-8 sıralarının kullanıldığından emin olmanız gerekir. Characters beyond the Unicode Basic Multilingual Plane (those above U+FFFF), represented in UTF-16 by surrogate code points (X'D800' through X'DFFF'), or four bytes in UTF-8, are not supported in message property names.

Eşitlik için karşılaştırılırken özellik adlarında ya da değerlerde fazladan işlem gerçekleştirilmez. Bu, örneğin, hiçbir kompozisyonun gerçekleşmediği ve bağlarının hiçbir özel anlam ifade etmediğine dair bir anlam ifade ediyor. Örneğin, önceden oluşturulmuş umlaut karakteri U+00FC , U+0075 + U+0308 karakterlerine eşdeğer olarak kabul edilmiyor ve karakter sırası ff değerinin Unicode U+FB00 (LATIN Küçük BAĞLANTI FF) ile eşdeğer olduğu kabul edilmez.

Tek tırnak işaretleri içindeki özellik verileri, herhangi bir bayt dizisi ile gösterilebilir ve geçerliliği denetlenmez.

### ***İletinin içeriğini seçme***

Bir ileti bilgi yükü içeriğine (içerik süzgeci olarak da bilinir) dayalı olarak abone olmak mümkündür; ancak, iletilerin bu tür bir aboneliğe teslim edilmesi gereken karar doğrudan IBM MQ tarafından gerçekleştirilemez; bunun yerine iletilerin işlenmesi için genişletilmiş bir ileti seçimi sağlayıcısı (örneğin, IBM Integration Bus) gereklidir.

When an application publishes on a topic string, where one or more subscribers have a selection string selecting on the content of the message, IBM MQ will request that the extended message selection provider parse the publication and inform IBM MQ whether the publication matches the selection criteria specified by each subscriber with a content filter.

Genişletilmiş ileti seçimi sağlayıcısı, yayının abonenin seçim dizisiyle eşleştiğini belirtiyorsa, ileti aboneyeye teslim edilmeye devam eder.

Genişletilmiş ileti seçimi sağlayıcısı, yayının eşleşmediğini belirlerse, ileti aboneyeye teslim edilmez. Bu, MQPUT ya da MQPUT1 çağrısının, MQRC\_PUBLICATION\_FAILURE neden kodu ile başarısız olmasına neden olabilir. Genişletilmiş ileti seçimi sağlayıcısı yayını ayırtıramıyorsa, neden kodu MQRC\_CONTENT\_ERROR döndürülür ve MQPUT ya da MQPUT1 çağrısı başarısız olur.

Genişletilmiş ileti seçimi sağlayıcısı kullanılamıyorsa ya da abonenin yayını alması gerekirken gerekmediğini belirleyemezse, MQRC\_SELECTION\_NOT\_AVAILABLE neden kodu döndürülür; MQPUT ya da MQPUT1 çağrısı başarısız olur.

Bir içerik süzgeci ile bir abonelik yaratılırken ve genişletilmiş ileti seçimi sağlayıcısı yoksa, MQSUB çağrısı MQRC\_SELECTION\_NOT\_AVAILABLE neden kodu ile başarısız olur. İçerik süzgeci olan bir abonelik sürdürülüyorsa ve genişletilmiş ileti seçimi sağlayıcısı yoksa, MQSUB çağrısı MQRC\_SELECTION\_NOT\_AVAILABLE uyarısını döndürür, ancak aboneliğin sürdürülmesine izin verilir.

### **İlgili kavramlar**

[Seçim dizgileri](#)

## **IBM MQ iletilerinin zamanuyumsuz tüketimi**

Zamanuyumsuz tüketim, bir MQI (Message Queue Interface; İleti Kuyruğu Arabirimi) uzantıları kümesini kullanır; MQI, MQCB ve MQCTL 'yi çağırır. Bu, bir MQI uygulamasının iletileri bir kuyruk kümesinden tüketmek üzere yazılmasına olanak sağlar. İletiler, uygulamanın iletiyi ileterek ya da iletiyi gösteren bir belirteç tarafından tanımlanan 'kod birimini' çağırarak uygulamaya teslim edilir.

Uygulama ortamlarında en basit şekilde, kod birimi bir işlev işaretçisi tarafından tanımlanır; ancak, diğer ortamlarda kod birimi bir program ya da modül adıyla tanımlanabilir.

İletilerin zamanuyumsuz olarak tüketilmesinde, aşağıdaki terimler kullanılır:

### **İleti tüketicisi**

Uygulamalar gereksinimle eşleşen bir program olduğunda bir ileti ile çağrılacak bir program ya da işlev tanımlamanıza olanak sağlayan bir programlama yapısı.

### **Olay işleyici**

Kuyruk yöneticisi susturulması gibi zamanuyumsuz bir olay gerçekleştiğinde çağrılacak bir program ya da işlev tanımlamanıza olanak sağlayan bir programlama yapısı.

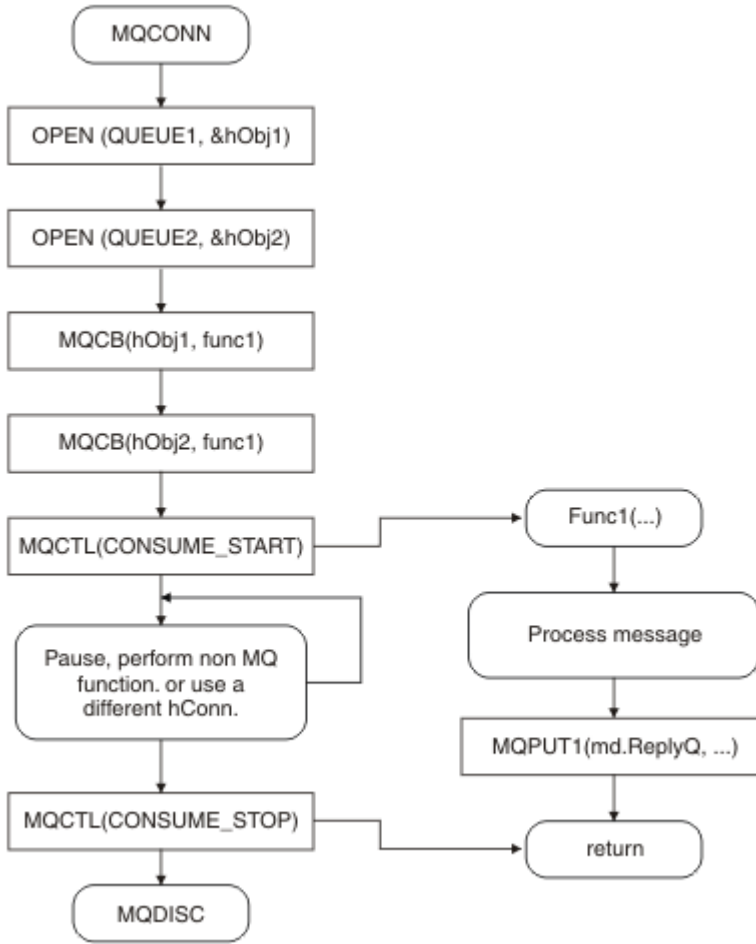
### **Geri çağırma**

Bir Message Consumer ya da Event Handler yordamlarına gönderme yapmak için kullanılan soysal terim.

Zamanuyumsuz tüketim, özellikle birden çok giriş kuyruğu ya da aboneliği işleyen yeni uygulamaların tasarlanmasını ve uygulanmasını basitleştirebilir. Ancak, birden fazla giriş kuyruğu kullanıyorsanız ve iletileri öncelik sırasına göre işleyorsanız, her bir kuyruk içinde öncelik sırası bağımsız olarak görülür: Yüksek öncelikli iletilerin bir kuyruğundan başka bir kuyruktan düşük öncelikli iletiler alabilirsiniz. Birden çok kuyrukta ileti sırası garanti edilmez. Ayrıca, API çıkışlarını kullanırsanız, MQCB ve MQCTL çağrılarını içermek için bunları değiştirmeniz gerekebilir.

Aşağıdaki şekillerde, bu işlevi nasıl kullanabileceğinin bir örneği yer almanıza yardımcı olur.

Şekil 5 sayfa 40 , iki kuyruktan gelen iletileri tüketen çok iş parçacıklı bir uygulamayı gösterir. Örnekte, tek bir işleve teslim edilen tüm iletiler gösterilir.



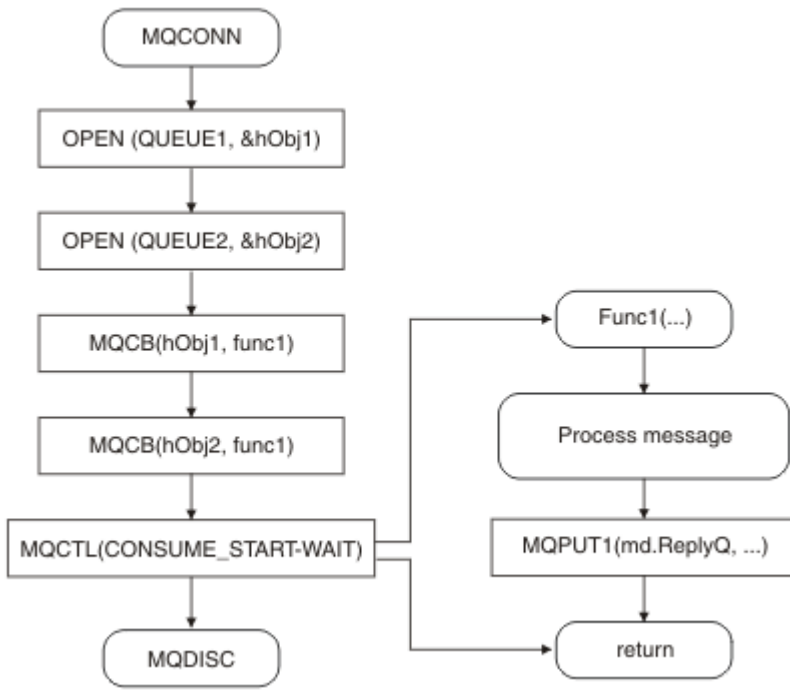
Şekil 5. İki kuyruktan alan standart ileti odaklı uygulama

**z/OS** z/OS' ta, ana denetim iş parçacığının sona erdirilmeden önce bir MQDISC çağrısı yayınlaması gerekir. Bu, geri bildirme iş parçacıklarının sistem kaynaklarını sona erdirmesini ve serbest bırakmasını sağlar

Şekil 6 sayfa 41 Bu örnek akış, iki kuyruktan gelen iletileri tüketen tek bir iş parçacıklı uygulamayı gösterir. Örnekte, tek bir işleve teslim edilen tüm iletiler gösterilir.

Zamanuyumsuz vakadan farkı, tüm tüketicilerin kendilerini devre dışı bırakılana kadar, denetim MQCTL ' nin yayınına dönmemesi; bu bir tüketici, bir MQCTL STOP isteği ya da kuyruk yöneticisi susturma işlemi yayınlamalıdır.





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

## İleti grupları

İletiler, ileti sıralamasına izin vermek için gruplar içinde oluşabilir.

İleti grupları, birden çok iletinin birbiriyle ilişkili olarak imlenmesine ve gruba uygulanmasına ilişkin mantıksal bir sıralamaya izin verir (bkz. “Mantıksal ve fiziksel sıralama” sayfa 811). Multiplatformsüzerinde, ileti bölümlenmesi büyük iletilerin daha küçük kesimlere ayrılmasını sağlar. Bir konuya yerleştirilirken gruplanmış ya da bölümlenmiş iletileri kullanamazsınız.

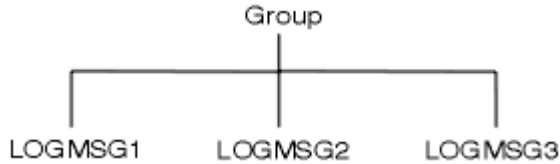
Bir grup içindeki sıradüzeni aşağıdaki gibidir:

### Grup

Bu, sıradüzendeki en yüksek düzeydir ve bir *GroupId* ile tanımlanır. Aynı *GroupId*' i içeren bir ya da daha fazla iletinden oluşur. Bu iletiler kuyruğun herhangi bir yerinde saklanabilir.

**Not:** *ileti* terimi, kuyruklardaki bir öğeyi göstermek için burada kullanılır; örneğin, MQGMO\_COMPLETE\_MSG belirtmeyen tek bir MQGET tarafından döndürülecektir.

Şekil 7 sayfa 41 , bir mantıksal ileti grubunu gösterir:



Şekil 7. Mantıksal ileti grubu

Bir kuyruk açık MQOO\_BIND\_ON\_GROUP belirtilerek, bu kuyruğa gönderilen bir gruptaki tüm iletileri, kuyruğun aynı örneğine gönderilmek üzere zorlayın. BIND\_ON\_GROUP seçeneği ile ilgili ek bilgi için [İleti zenginliklerine işlemebaşlıklı konuya](#) bakın.

### Mantıksal ileti

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

Bir grup içindeki mantıksal iletileri aşağıdaki gibi kullanın:

- Siparişin verildiğinden emin olun (bu, iletinin iletilmekte olduğu koşullar altında garanti verilmezse).
- Uygulamaların benzer iletileri gruplamasına izin ver (örneğin, tümü aynı sunucu eşgörünümü tarafından işlenmek zorunda olanlar).

Bir grup içindeki her ileti, kesimlere bölünmediği sürece, tek bir fiziksel iletten oluşur. Her ileti mantıksal olarak ayrı bir iletidir ve yalnızca, MQMD 'deki *GroupId* ve *MsgSeqNumber* alanlarının gruptaki diğer iletilerle herhangi bir ilişkiyi taşıması gerekir. MQMD 'deki diğer alanlar bağımsızdır; bazıları, gruptaki tüm iletiler için aynı olabilir, ancak diğerleri farklı olabilir. Örneğin, bir gruptaki iletiler farklı biçim adları, CCSID 'ler ve kodlamalar olabilir.

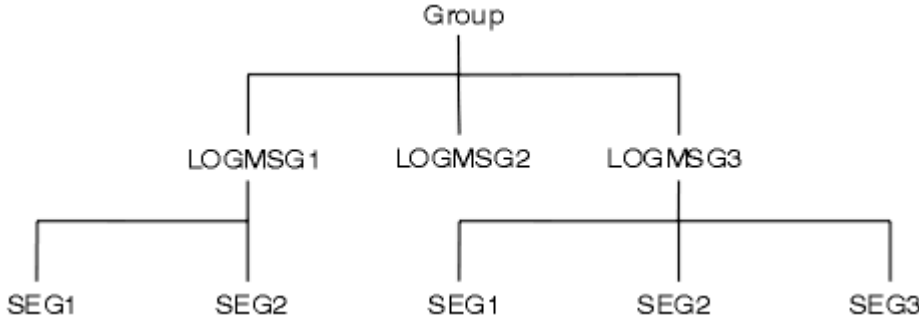
## Bölüm

Kesimler, uygulama 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 bölümlenmesi” sayfa 829 başlıklı konuya bakın.

Bir ileti, *kesimler* adı verilen daha küçük iletilere bölünmesi halinde ayrılır. A segment of a message is identified by the *GroupId*, *MsgSeqNumber*, and *Offset* fields. *Offset* alanı, bir ileti içindeki ilk bölüm için sıfır düzeyinde başlar.

Her bir kesim, bir gruba ait olabilecek bir fiziksel iletinden oluşur (Şekil 8 sayfa 42 , bir grup içindeki iletilere bir örnek gösterir). Bölüm, mantıksal olarak tek bir iletinin bir parçasıdır; bu nedenle, MQMD 'deki *MsgId*, *Offset* ve *MsgFlags* alanlarının aynı iletinin ayrı kesimleri arasında farklılık göstermeleri gerekir. Bir kesim gelmezse, neden kodu MQRC\_INCOMPLE\_GROUP ya da MQRC\_INCOMPLE\_MSG uygun olarak döndürülür.

Şekil 8 sayfa 42 , bir grup mantıksal ileti grubunu, bazıları bölümlenmiş bir grup iletisi gösterir:



Şekil 8. Kesimlere ayrılmış iletiler

**z/OS** IBM MQ for z/OS üzerinde bölümlenme desteklenmiyor.

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

## İlgili kavramlar

“İleti bölümlenmesi” sayfa 829

Bölüm iletileri hakkında bilgi edinmek için bu bilgileri kullanın. Bu özellik IBM MQ for z/OS üzerinde ya da IBM MQ classes for JMS kullanan uygulamalar tarafından desteklenmez.

## İlgili başvurular

“Mantıksal ve fiziksel sıralama” sayfa 811

Kuyruklardaki iletiler *fiziksel* ya da *mantıksal* düzende (her bir öncelik düzeyi içinde) gerçekleştirilebilir.

MQMD-İleti tanımlayıcısı

## İleti kalıcılığı

Kalıcı iletiler günlüklere ve kuyruk veri dosyalarına yazılır. Bir kuyruk yöneticisi bir hatadan sonra yeniden başlatılırsa, günlüğe kaydedilen verilerden gerekli olan bu kalıcı iletileri kurtarır. Kuyruk yöneticisi durursa, kalıcı olmayan iletiler atılır; durdurma sayfası bir işletmen komutunun sonucu olarak mı, yoksa sisteminizin bir kısmının başarısızlığı nedeniyle mi atılır.

**z/OS** z/OS üzerindeki bir bağlaşım tesisinde (CF) saklanan kalıcı olmayan iletiler, bu işlem için bir kural dışı durum. CF kullanılabilir olmaya devam ettiği sürece devam ederler.

Bir ileti yarattığınızda, varsayılan değerleri kullanarak ileti tanımlayıcısını (MQMD) kullanıma hazırlarken, iletiye ilişkin kalıcılık, MQOPED komutunda belirtilen kuyruğun **DefPersistence** özneliğinden alınır. Diğer bir seçenek olarak, iletiyi kalıcı ya da kalıcı olmayan bir ileti olarak tanımlamak için MQMD yapısının *Persistence* alanını kullanarak iletinin kalıcısını ayarlayabilirsiniz.

Kalıcı iletiler kullandığınızda uygulamanızın performansı etkilenir; etkinin kapsamı, makinenin G/Ç altsisteminin başarımlı özelliklerine ve her altyapıda eşitleme noktası seçeneklerini nasıl kullandığınızı gösterir:

- Geçerli iş biriminin dışında, sürekli bir ileti, her put ve get işleminde diske yazılıdır. Bkz. [“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888.](#)
- **ULW** **z/OS** IBM dışındaki tüm altyapılar için, yürürlükteki iş birimi içindeki kalıcı bir ileti, iş birimi kesinleştirildiğinde günlüğe kaydedilir ve iş birimi birçok kuyruk işlemi içerebilir.

Hızlı ileti sistemi için kalıcı olmayan iletiler kullanılabilir. Hızlı iletiler hakkında daha fazla bilgi için bkz. [İletilerin güvenliği](#).

**Not:** Bir iş birimi içindeki kalıcı iletilerin yazılması ve kalıcı iletilerin bir birimin ya da işin dışında yazılması, uygulamalarınız için ciddi başarımlı sorunlarına neden olabilir. Bu durum özellikle, her iki işlem için de aynı hedef kuyruk kullanıldığında geçerlidir.

## Teslim edilememiş olan iletiler

Kuyruk yöneticisi bir iletiyi kuyruğa koyamadığında, çeşitli seçenekleriniz vardır.

Yapabilecekleriniz:

- İletiyi yeniden kuyruğa koyma girişiminde bulunuldu.
- İletinin gönderene döndürülemediğini isteyin.
- Mesajı ölü mektup kuyruğuna koyun.

Ek bilgi için [“Yordamsal program hatalarının işlenmesi” sayfa 1073](#) başlıklı konuya bakın.

## Yedeklenen iletiler

Bir iş biriminin denetimi altındaki bir kuyruktan ileti işlenirken, iş birimi bir ya da daha çok iletiden oluşabilir. Bir geriletme ortaya çıkarsa, kuyruktan alınan iletiler kuyrukta yeniden başlatılır ve başka bir iş biriminde işlenebilir. Belirli bir iletinin işlenmesi soruna neden oluyorsa, iş birimi yeniden yedeklenir. Bu işlem bir işleme döngülerine neden olabilir. Kuyruğa yerleştirilecek iletiler kuyruktan kaldırılır.

Uygulama, MQMD 'nin *BackoutCount* alanını sınavarak böyle bir döngüye yakalanan iletileri saptayabilir. Uygulama durumu düzeltebilir ya da bir operatöre uyarı yayınlayabilir.

**Multi** Geriletme sayısı her zaman kuyruk yöneticisinin yeniden başlatılabilmesine neden olur. **HardenGetBackout** özneliğe yapılan herhangi bir değişiklik yoksayılr.

**z/OS** Paylaşılan kuyruklar için, geriletme sayısı her zaman kuyruk yöneticisinin yeniden başlatılmasına neden olur. z/OS üzerindeki diğer tüm yapılandırmalar için, özel kuyruklara ilişkin geri çıkış sayısının kuyruk yöneticisinin yeniden başlatılmasına neden olduğundan emin olmak için, *HardenGetBackout* özneliğini MQQA\_BACKUT\_HENDENED; olarak ayarlayın; tersi durumda, kuyruk yöneticisi yeniden başlatmak zorunda kalırsa, her ileti için doğru bir geri alma sayısı tutmaz. Özneliğin bu şekilde ayarlanması, fazladan işleme maliyetini ekler.

İletilerin kesinleştirilmesi ve yedeklenmesi hakkında daha fazla bilgi için bkz. [“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888.](#)

## Yanıtın gönderileceği kuyruk ve kuyruk yöneticisi

Gönderdiğiniz bir iletiye yanıt olarak ileti alabileceğiniz durumlar da vardır:

- Bir istek iletilisinde yanıt olarak yanıt iletilisi
- Beklenmeyen bir olay ya da süre bitimi ile ilgili bir rapor iletilisi
- Bir COA (rakip Onaylama Onayı) veya bir COD (Teslim Onayı) olayı hakkında bir rapor iletilisi
- Bir PAN (Pozitif İşlem Bildirimi) ya da bir NAN (Negatif İşlem Bildirimi) olayı ile ilgili bir rapor iletilisi

Using the MQMD structure, specify the name of the queue to which you want reply and report messages sent, in the *ReplyToQ* field. *ReplyToQMGr* alanında, yanıtın gönderileceği kuyruk yöneticisinin adını belirtin.

*ReplyToQMGr* alanını boş bırakırsanız, kuyruk yöneticisi, kuyrukta bulunan ileti tanımlayıcısında aşağıdaki alanların içeriğini ayarlar:

### **ReplyToQ**

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

### **ReplyToQMGr**

*ReplyToQ* uzak bir kuyruğun yerel tanığıysa, *ReplyToQMGr* alanı, uzak kuyruğa sahip 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 fazla girişimde bulunmasını isteyebilirsiniz ve hata başarısız olursa iletinin atıldığını isteyebilirsiniz. İleti, teslim edilemedikten sonra atılmayacaksa, uzak kuyruk yöneticisi iletiyi ölüme ilişkin ileti (teslim edilmemiş ileti) kuyruğuna koyar (bkz. [“Ölü harf \(teslim edilmemiş ileti\) kuyruğunun kullanılması” sayfa 1077](#)).

## **İleti bağılamı**

*İleti bağılamı* bilgileri, iletiyi alan uygulamanın, iletiyi oluşturan iletiyi bulmasını sağlayan bir uygulamaya olanak tanır.

Alma uygulaması aşağıdakileri yapmak isteyebilirler:

- Gönderme uygulamasının doğru yetki düzeyine sahip olup olmadığını denetleyin
- Bazı muhasebe işlevini, gerçekleştirmesi gereken işler için gönderme uygulamasını şarj edebilmesini sağlar.
- Üzerinde çalıştığı tüm iletilerin bir denetim izini alıyoy

Bir iletiyi kuyruğa koymak için MQPUT ya da MQPUT1 çağrısını kullandığınızda, kuyruk yöneticisinin ileti tanımlayıcısına varsayılan bağlam bilgileri eklemesini belirleyebilirsiniz. Uygun yetki düzeyine sahip uygulamalar ek bağlam bilgileri ekleyebilir. Bağlam bilgilerinin nasıl belirtileceği hakkında daha fazla bilgi için bkz. [“İleti bağılamı bilgilerinin denetlenmesi” sayfa 797](#).

Kullanıcı bağılamı, aşağıdaki rapor iletilisi türleri oluşturulurken kuyruk yöneticisi tarafından kullanılır:

- Teslim edilmeyi onayla
- Son kullanma tarihi

Bu rapor iletileri oluşturulduğunda, kullanıcı bağılamı, raporun hedefi üzerinde + put ve + passid yetkisi olup olmadığını denetlenir. Kullanıcı bağılamının yetkisi yetersiz olduğu durumlarda, rapor iletilisi, tanımlanmış olan ileti kuyruğuna yerleştirilir. Ölü-mektup kuyruğu olmadığı durumlarda, rapor iletilisi atılır.

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

## **Kimlik bağılamı**

*Tanıtıcı bağılamı* bilgileri, iletiyi ilk olarak kuyruğa koyan uygulamanın kullanıcılarını tanımlar. Uygun şekilde yetkili uygulamalar aşağıdaki alanları ayarlayabilir:

- Kuyruk yöneticisi, *UserIdentifier* alanını, kullanıcıyı tanımlayan bir adla doldurur; kuyruk yöneticisinin bunu yapabilme biçimi, uygulamanın çalışmakta olduğu ortama bağlıdır.
- Kuyruk yöneticisi, *AccountingToken* alanını, iletiyi koyan uygulamadan saptadığı bir simgeyle ya da sayıyla doldurur.
- Uygulamalar, kullanıcı hakkında içermek istedikleri ek bilgiler (örneğin, şifrelenmiş bir parola) için *AppIdentityData* alanını kullanabilir.

A Windows systems security identifier (SID) is stored in the *AccountingToken* field when a message is created under IBM MQ for Windows. SID, *UserIdentifier* alanını tamamlamak ve bir kullanıcının kimlik bilgilerini oluşturmak için kullanılabilir.

Kuyruk yöneticisinin *UserIdentifier* ve *AccountingToken* alanlarını doldurduyla ilgili bilgi için [UserIdentifier](#) ve [AccountingToken](#) içindeki bu alanların tanımlarına bakın.

Bir kuyruk yöneticisinden diğerine ileti geçiren uygulamalar, diğer uygulamaların iletinin kökeninin kimliğini bilmesi için kimlik bağlamı bilgilerini de iletmelidir.

## Kaynak bağlamı

*Köken bağlamı* bilgileri, iletiyi iletinin bulunduğu kuyruğa koyan uygulamayı açıklar. İleti tanımlayıcısı, kaynak bağlamı bilgileri için aşağıdaki alanları içerir:

- *PutAppType* , iletiyi (örneğin, bir CICS hareketi) koyan uygulama tipini tanımlar.
- *PutAppName* , iletiyi (örneğin, bir iş ya da hareketin adı) koyan uygulamanın adını tanımlar.
- *PutDate* , iletinin kuyruğa konacağı tarihi tanımlar.
- *PutTime* , iletinin kuyrukta ne zaman konacağı tanımlıyor.
- *AppOriginData* , bir uygulamanın iletinin kökeniyle ilgili olarak eklemek istediği fazladan bilgileri tanımlar. Örneğin, kimlik verilerinin güvenilir olup olmadığını göstermek için uygun yetkili uygulamalar tarafından ayarlanabilirdi.

Kaynak bağlamı bilgileri genellikle kuyruk yöneticisi tarafından sağlanır. Greenwich Ortalama Saati (GMT), *PutDate* ve *PutTime* alanları için kullanılır. [PutDate](#) ve [PutTime](#) içinde bu alanların açıklamalarını görebilirsiniz.

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

## IBM MQ nesnelere

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

Kuyruk yöneticileri, bu nesnelere özelliklerini (öznitelikler olarak bilinir) tanımlar. Bu özneliklerin değerleri, IBM MQ ' in bu nesnelere işleminin yolunu etkiler. Uygulamalarınızdan bu nesnelere denetlemek için Message Queue Interface (MQI) olanağını kullanıyorsunuz. Nesnelere, bir programdan adreslendiklerinde bir *nesne tanımlayıcısı* (MQOD) ile tanımlanır.

Nesnelere 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 çağrısını kullandığında, kuyruk yöneticisi, uygulamanın o nesneye erişime izin verebilmesi için gerekli düzeyde yetki olduğunu denetler. Çekler, açılmakta olan nesnenin adı üzerinde yapılır.

### İlgili kavramlar

[“İleti bağlamı bilgilerinin denetlenmesi” sayfa 797](#)

Bir iletiyi kuyruğa koymak için MQPUT ya da MQPUT1 çağrısını kullandığınızda, kuyruk yöneticisinin ileti tanımlayıcısına varsayılan bağlam bilgileri eklemesini belirleyebilirsiniz. Uygun yetki düzeyine sahip

uygulamalar ek bağlam bilgileri ekleyebilir. Bağlam bilgilerini denetlemek için, MQPMO yapısındaki seçenekler alanını kullanabilirsiniz.

### İlgili başvurular

[“İleti bağlamına ilişkin MQAç seçenekleri” sayfa 789](#)

Bir iletiyi bir kuyruğa koyduğunuzda bağlam bilgilerini bir iletiyle ilişkilendirebilmek istiyorsanız, kuyruğu açtığınızda ileti bağlamı seçeneklerinden birini kullanmanız gerekir.

## Windows Microsoft Transaction Server uygulamalarının hazırlanması ve çalıştırılması

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

IBM MQ kaynaklarına erişen Microsoft Transaction Server (MTS) uygulamalarının nasıl geliştirileceği hakkında genel bilgi edinmek için IBM MQ Yardım Merkezi 'nde MTS ' nin (MTS) bölümüne bakın.

Bir MTS uygulamasını IBM MQ MQI client uygulaması olarak çalıştırmak üzere hazırlamak için, uygulamanın her bileşeni için aşağıdakilerden birini yapın:

- Bileşen, MQI için C dili bağ tanımlarını kullanıyorsa, [“C programlarını Windows’inde hazırlama” sayfa 1052](#) içindeki yönergeleri izleyin, ancak bileşeni mqic.libyerine mqicxa.lib kitaplığıyla bağlantılayın.
- Bileşen IBM MQ C++ sınıflarını kullanıyorsa, [“Building C++ programs on Windows” sayfa 523](#) içindeki yönergeleri izleyin, ancak bileşeni imqc23vn.libyerine imqx23vn.lib kitaplığıyla bağlantılayın.
- Bileşen, MQI için Visual Basic dili bağ tanımlarını kullanıyorsa, [“Preparing Visual Basic programs in Windows” sayfa 1056](#) içindeki yönergeleri izleyin, ancak Visual Basic projesini tanımlarken **Koşullu Derleme Bağımsız Değişkenleri** alanında MqType=3 yazın.
- If the component uses the IBM MQ Automation Classes for ActiveX (MQAX), define an environment variable, GMQ\_MQ\_LIB, with the value mqic32xa.dll.

Ortam değişkenini uygulamanızın içinden tanımlayabilir ya da kapsamı sistem çapında olacak şekilde tanımlayabilirsiniz. Ancak, sistemi geniş olarak tanımlamak, var olan MQAX uygulamasının, uygulama içinden ortam değişkenini tanımlamamasına, yanlış şekilde davranmasına neden olabilir.

## IBM MQ uygulamaları için dikkat edilmesi gereken noktalar

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğinize karar verdiğinizde, IBM MQ tarafından sunulan özelliklerin nasıl kullanılacağına karar vermeniz gerekir.

Bir IBM MQ uygulaması tasarlarken aşağıdaki sorular ve seçenekler göz önünde bulundurulur:

### Uygulama tipi

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

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

Buna ek olarak, IBM MQ yönetimini otomatikleştirmek için kendi uygulamalarınızı da yazabilirsiniz. Daha fazla bilgi için bakınız: [The IBM MQ Administration Interface \(MQAI\)](#) ve [Automating admining tasks](#).

### Programlama dili

IBM MQ , uygulama yazmak için çeşitli programlama dillerini destekler. Daha fazla bilgi için bkz. [“IBM MQ için uygulama geliştirilmesi” sayfa 5](#).

## Birden fazla platform için uygulamalar

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

Örneğin, ANSI standart C ' de C kodu kullanıyorsanız, Platforma özgü işlev daha hızlı ya da daha verimli olsa bile, eşdeğer bir platforma özgü işlev yerine standart bir C kitaplığı işlevi kullanın. Bu kural dışı durum, #ifdef kullanılarak her iki durum için kodladığınızda, koddaki verimliliğin ayrıştırılması durumunda olur. Örneğin:

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

## Kuyruk tipleri

Her gereksinim duyarsanız bir kuyruk yaratmak istiyor musunuz, yoksa önceden ayarlanmış kuyrukları kullanmak mı istiyorsunuz? Kullanmayı bitirdiğinizde bir kuyruğu silmek istiyor musunuz, yoksa yeniden kullanılacak bir kuyruk mu? Uygulama bağımsızlığı için diğer ad kuyruklarını kullanmak istiyor musunuz? Hangi kuyrukların desteklendiğini görmek için [Kuyruklarkonusuna](#) bakın.

## Paylaşılan kuyruklar, kuyruk paylaşım grupları ve kuyruk paylaşım grubu kümeleri kullanma (yalnızca IBM MQ for z/OS )

Kuyruk paylaşım gruplarıyla paylaşılan kuyruklar kullandığınızda, olası kullanılabilirlik, ölçeklenebilirlik ve iş yükü dengelemesinin avantajlarından yararlanmak isteyebilirsiniz. Ek bilgi için [Paylaşılan kuyruklar ve kuyruk paylaşım grupları](#) başlıklı konuya bakın.

Ayrıca, ortalama ve en yoğun ileti akışlarını tahmin etmek ve iş yükünü dağıtmak için kuyruk paylaşım grubu kümelerini kullanmayı düşünebilirsiniz. Ek bilgi için [Paylaşılan kuyruklar ve kuyruk paylaşım grupları](#) başlıklı konuya bakın.

## Kuyruk yöneticisi kümelerini kullanma

Kümeleri kullanırken mümkün olan basitleştirilmiş sistem yönetiminden ve artırılabilir düzeyde kullanılabilirlik, ölçeklenebilirlik ve iş yükü dengelemesi avantajlarından yararlanmak isteyebilirsiniz.

## İleti tipleri

Basit iletiler için veri paketleri kullanmak isteyebilirsiniz, ancak diğer durumlar için istek iletileri (yanıt beklediğiniz) istemeniz gerekebilir. İletilerinizin bazılarını farklı öncelikler atamak isteyebilirsiniz. İletileri tasarlamaya ilişkin daha fazla bilgi için bkz. [“İletilere ilişkin tasarım teknikleri” sayfa 54.](#)

## Yayınlama/abone olma ya da noktadan noktaya ileti alışverişi kullanma


Yayınlama/abone olma ileti alışverişi kullanılarak, bir gönderme uygulaması, IBM MQ iletilerinde paylaşmak istediği bilgileri IBM MQ yayınlama yönetilen standart bir hedefe gönderiyor? Abone olun ve IBM MQ ' in bu bilgilerin dağıtımını işlemlerini sağlar. Hedef uygulamanın aldığı bilgilerin kaynağı hakkında herhangi bir bilgi sahibi olması gerekmez, yalnızca bir ya da daha fazla konuya ilgi kaydeder ve bu bilgileri kullanılabilir olduğunda alır. Yayınlama/abone olma ileti alışverişi hakkında daha fazla bilgi için bkz. [Yayınlama/abone olma ileti alışverişi.](#)

Bir gönderme uygulaması, noktadan noktaya ileti sistemini kullanarak, belirli bir kuyruğa ileti gönderir. Bu ileti, bir alma uygulamasının onu alabileceği bir kuyruğa gönderir. Alma uygulaması, belirli bir kuyruktan iletileri alır ve içerikleri üzerinde işlem yapar. Uygulama genellikle hem gönderici, hem de alıcı olarak işlev görecektir, başka bir uygulamaya sorgu gönderip yanıt alacaktır.

## IBM MQ programlarınızı denetleme

Bazı programları otomatik olarak başlatmak ya da belirli bir ileti kuyruğa gelene kadar ( IBM MQ *tetikleme* özelliğini kullanarak) program beklemeniz gerekebilir. Bkz. [“Starting IBM MQ applications using triggers” sayfa 899](#) ). Diğer bir seçenek olarak, bir kuyruktaki bulunan iletiler yeterince hızlı işlenmediklerinde, uygulamanın başka bir eşgörünümünü başlatmak isteyebilirsiniz ( *Özel işlemden geçirme olayları* içinde açıklandığı gibi IBM MQ *izleme kodu ekleme olayları* özelliğini kullanarak).


## Uygulamanızın bir IBM MQ istemcisinde çalıştırılması

İstemci ortamında tam MQI desteklenir ve bir yordamsal dille yazılmış hemen hemen her IBM MQ uygulaması bir IBM MQ MQI client' ta çalışmak üzere yeniden bağlantılandırılabilir. Link the application on the IBM MQ MQI client to the MQIC library, rather than to the MQI library.  Get(signal) on z/OS is not supported.

**Not:** Bir IBM MQ istemcisinde çalışan bir uygulama, koşut zamanlı olarak birden çok kuyruk yöneticisine bağlanabilir ya da bir MQCONN ya da MQCONNX çağrısında yıldız (\*) ile bir kuyruk yöneticisi adı kullanabilir. İstemci kitaplıkları yerine kuyruk yöneticisi kitaplıklarına bağlantı oluşturmak istiyorsanız, bu işlev kullanılamayacağını uygulamayı değiştirin.

Ek bilgi için [“IBM MQ MQI client ortamında uygulamaları çalıştırma” sayfa 953](#) başlıklı konuya bakın.

## Uygulama performansı

Design decisions can impact your application performance, for suggestions for enhancing performance of IBM MQ applications, see [“Uygulama tasarımı ve performansı ile ilgili önemli noktalar” sayfa 56](#)  ve [“IBM i uygulamaları için tasarım ve performans konuları” sayfa 59](#).

## Gelişmiş IBM MQ teknikleri


Daha ileri düzey uygulamalar için, yanıtları ilintilendirmek ve IBM MQ bağlam bilgilerini oluşturmak ve göndermek 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 bütünlüğünün korunması

İletinin kabul edilebilir bir kaynaktan gönderildiğini test etmek için iletiyle geçirilen bağlam bilgilerini kullanabilirsiniz. Verilerinizin diğer kaynaklarla tutarlı olmasını sağlamak için IBM MQ 'in ya da işletim sisteminizin sağladığı uyumluluk olanaklarını kullanabilirsiniz (ek ayrıntılar için [“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#) ' a bakın). Önemli iletilerin sunulmasını sağlamak için IBM MQ iletilerinin *persistence* özelliğini kullanabilirsiniz.

## IBM MQ uygulamalarının test edilmesi

IBM MQ programları için uygulama geliştirme ortamı, başka bir uygulama için bundan farklı değildir; bu nedenle, IBM MQ izleme olanaklarının yanı sıra aynı geliştirme araçlarını da kullanabilirsiniz.

 When testing CICS applications with IBM MQ for z/OS, you can use the CICS Execution Diagnostic Facility (CEDF). CEDF, tüm CICS hizmetlerine çağrılarının yanı sıra, her MQI çağrısının girişini ve çıkışını tuzaklıyor. Ayrıca, CICS ortamında, tanılama bilgilerini her MQI çağrısından önce ve sonra sağlamak için API geçiş çıkış programı yazabilirsiniz. Bunun nasıl yapacağına ilişkin bilgi için bkz. [“Uygulamaların IBM MQ for z/OS üzerinde kullanılması ve yazılması” sayfa 922](#).

 IBM i uygulamalarını test ederken standart Hata Ayıklayıcısı 'nı kullanabilirsiniz. Bunu başlatmak için STRDBG komutunu kullanın.

## Kural dışı durumlar ve hatalar işleniyor

Teslim edilemeyen iletilerin nasıl işleneceğini ve kuyruk yöneticisi tarafından size bildirilen hata durumlarının nasıl çözümleneceğini göz önünde bulundurmanız gerekir. Bazı raporlar için, rapor seçeneklerini MQPUT üzerine ayarlamanız gerekir.

## İlgili kavramlar

[IBM MQ teknik genel bakış](#)

[“z/OS uygulamaları için tasarım ve performans konuları” sayfa 61](#)

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

[“IBM MQ için uygulama geliştirilmesi” sayfa 5](#)

İletileri göndermek ve almak, kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için uygulamalar geliştirebilirsiniz. IBM MQ , birçok farklı dil ve çerçeve içinde yazılmış uygulamaları destekler.

[“Uygulama geliştirme kavramları” sayfa 7](#)

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz.

IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.



[“Kuyruğa alma için bir yordamsal uygulama yazma” sayfa 759](#)

Kuyruk yöneticisi, yayınlama/abone olma, nesneleri açma ve kapama nesnelere bağlanma ve bağlantıyı kesme, kuyruğa alma ve bağlantı kesme hakkında bilgi edinmek için bu bilgileri kullanın.

[“İstemci yordamsal uygulamaları yazılıyor” sayfa 945](#)

Bir yordamsal dil kullanarak IBM MQ üzerinde istemci uygulamaları yazmak için bilmeniz gerekenler.

[“.NET uygulamalarının geliştirilmesi” sayfa 528](#)

IBM MQ classes for .NET , .NET programlama çerçevesinde yazılmış bir programın IBM MQ ile IBM MQ MQI client arasında bağlantı kurmasını ya da bir IBM MQ sunucusuna doğrudan bağlanmasını sağlar.

[“C++ uygulamaları geliştirilmesi” sayfa 500](#)

IBM MQ provides C++ classes equivalent to IBM MQ objects and some additional classes equivalent to the array data types. Bu, MQI aracılığıyla olmayan bazı özellikleri sağlar.

[“kullanmaIBM MQ classes for JMS” sayfa 77](#)

IBM MQ classes for Java Message Service (IBM MQ classes for JMS), IBM MQ ile birlikte verilen JMS sağlayıcısıdır. javax.jms paketinde tanımlanan arabirimlerin yanı sıra, IBM MQ classes for JMS , JMS API ' ye iki uzantı kümesi sağlar.

[“kullanmaIBM MQ classes for Java” sayfa 321](#)

Java ortamında IBM MQ kullanın. IBM MQ classes for Java allow a Java application to connect to IBM MQ as an IBM MQ client, or connect directly to an IBM MQ queue manager.

[“Component Object Model Interface olanağının kullanılması \( ActiveX için IBM MQ Otomasyon Sınıfları\)” sayfa 640](#)

The IBM MQ Automation Classes for ActiveX (MQAX) are ActiveX components that provide classes that you can use in your application to access IBM MQ.

## V 9.1.2 Desteklenen programlama dillerinde uygulama adının belirtilmesi

IBM MQ 9.1.2' öncesinde, Java ya da JMS istemci uygulamalarında bir uygulama adı zaten belirtebilirsiniz. IBM MQ 9.1.2 ' dan bu özellik, IBM MQ for Multiplatforms üzerindeki diğer programlama dillerine uzatılır.

### Uygulama adının nasıl kullanıldığı

Uygulama adının çıkışı şöyledir:

- 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 olmayan uygulamalar için varsayılan uygulama adı, Windows dışında yürütülebilir dosyanın kısaltılmış adıdır.

**Windows** Windows' ta, varsayılan ad tam olarak nitelenmiş yürütülür dosya adıdır, soldaki 28 karakter olarak kısaltılır.

Java uygulamaları için, örnek olarak 28 karaktere kadar kesilen paket adının başına örnek olarak örnek olarak gelen sınıf adıdır.

Ek bilgi için bkz. [PutApplAd](#).

IBM MQ 9.1.2' tan, IBM MQ for Multiplatforms üzerindeki uygulamalar uygulama adlarını yönetimsel olarak ya da çeşitli programlama yöntemlerini kullanarak ayarlayabilir. Bu, uygulama etkinliği izlemeyi yapılandırdığınızda ya da çeşitli **runmqsc** komutlarından çıkış yaparken, uygulamaların daha anlamlı bir platforma bağımsız ad sağlamasına olanak sağlar.

IBM MQ 9.1.2 , uygulamaları tek bir küme içinde yeniden dengeleme yeteneği ekler. Bunu başarmak için anlamlı uygulama adları kullanılır.

## Desteklenen karakterler

Uygulama adını nasıl belirtmenize ilişkin daha fazla bilgi için bkz. [“Önerilen uygulama adı karakterleri” sayfa 50](#).

## Programlama dilleri

Uygulamaların C içindeki IBM MQ kitaplıklarına ve diğer programlama dillerine nasıl çözülmesine ilişkin ek bilgi edinmek için [“Programlama dili bağlantıları” sayfa 52](#) ' e bakın.

## Yönetilen .NET uygulamaları

Yönetilen .NET uygulamalarının uygulama adını nasıl sağlayabileceğiyle ilgili bilgi için bkz. [“Yönetilen .NET uygulamaları” sayfa 53](#).

## XMS uygulamaları

See [“XMS uygulamaları” sayfa 53](#) for information on how XMS applications can provide the application name.

## Java ve JMS bağ tanımları uygulamaları



See [“Java ve JMS bağ tanımları uygulamaları” sayfa 54](#) for information on how Java and JMS applications can provide the application name.

### İlgili kavramlar

[Uygulama etkinliği izlemesi](#)

[Tek biçimli kümeler](#)

### İlgili başvurular

[MQCNO](#)

## V 9.1.2 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 özniteliğin ayrıntıları için [kuyruk yöneticisine ilişkin öznitelikler](#) konusuna bakın.

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

Uygulama adının kuyruk yöneticileri arasında sorunsuz bir şekilde geçmesini sağlamak için **V 9.1.5** ve kaynak izleme konuları aracılığıyla uygulama kaynağı izlemesine izin vermek için, uygulama adları yalnızca tek baytlık yazdırılabilir karakterler içermelidir.

**Not:** **V 9.1.5** Uygulama adlarında eğik çizgi ve ve ve imi karakterlerinin kullanılmasını da önlemelisiniz.

Bu, adı aşağıdaki şekilde sınırlar:

- Alfayısal karakterler: A-Z, a-z ve 0-9

**Not:** EBCDIC Katakana kullanan sistemlerde uygulama adlarında küçük harfli a-z karakterlerini kullanmamalısınız.

- Boşluk karakteri
- V 9.1.5 EBCDIC ' de değişmeyen yazdırılabilir karakterler: + < = > % \* ' ( ) , \_  
- . : ; ?

## Karakterleri ayarlama

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

	C bağ tanımları ve istemci	Java bağ tanımları ve istemci	JMS bağ tanımları ve istemci	Yönetilen .NET istemci si	Yönetilmeyen .NET bağ tanımları ve istemci si	Yönetilen XMS istemci si	Yönetilmeyen XMS bağ tanımları ve istemci si
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 özellik	.NET geçersiz kılınmış özellik		
MQOrtamı		Java MQOrtamı		.NET MQOrtamı	.NET MQOrtamı		
Bağlantı üreticisi özelliği			Bağlantı üreticisi özelliği			Bağlantı üreticisi özelliği	Bağlantı üreticisi özelliği
JMSAdmin			JMSAdmin			JMSAdmin	JMSAdmin
MQCNO	Bağlantı 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				

	C bağ tanımları ve istemci	Java bağ tanımları ve istemci	JMS bağ tanımları ve istemci	Yönetilen .NET istemci si	Yönetilmeyen .NET bağ tanımları ve istemci si	Yönetilen XMS istemci si	Yönetilmeyen XMS bağ tanımları ve istemci si
Varsayılan ad	<u>Varsayılan ad</u>			<u>.NET Varsayılan adı</u>	<u>.NET Varsayılan adı</u>	<u>.NET Varsayılan adı</u>	<u>.NET Varsayılan adı</u>

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

- COBOL
- Çevirici
- Visual Basic

## 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

**ULW** MQCNO ' ya iki yeni alan eklendi ve **Version** sayısı 7olarak artırıldı. Daha fazla bilgi için bkz. [MQCNO](#).

**Not:** **z/OS** Bir IBM MQ for z/OS kuyruk yöneticisine bağlanırken yalnızca IBM MQ classes for JMSya da IBM MQ classes for Javave istemci kipi bağlantılarını kullanan uygulamaların uygulama adını ayarlamasına izin verilir.

### ULW Ortam değişkenleri

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

```
export MQAPPLNAME=ExampleAppName
```

Ek bilgi için [Ortam değişkenleri açıklamaları](#) başlıklı konuya bakın.

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

**Not:** Öznitelik, yalnızca desteklenen programlama dilleri, yönetilmeyen .NET ve yönetilmeyen XMS bağlantıları için geçerlidir.

### ULW İstemci yapılandırma kütüğü

Henüz bir uygulama adı seçmediyseniz ve bağlantı bir istemci bağlantıysa, kuyruk yöneticisiyle bağlantıyı tanımlamak için istemci yapılandırma dosyasında (örneğin, *mqclient.ini*) aşağıdakileri belirtebilirsiniz.

```
Connection:
  App1Name=ExampleAppName
```

## Notlar:

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

Örnek bir yapılandırma dosyası için [Yapılandırma kütüğü kullanılarak istemci yapılandırılması](#) konusuna bakın.

### 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. [PutAppAdı](#).

## 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, önceki 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=ExampleAppName
```

### 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, "ExampleAppName" );
MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
```

Ek bilgi için bkz. [“.NET'teki yönetilen ve yönetilmeyen işlemler” sayfa 617](#).

### MQEnvironment

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

```
MQEnvironment.AppName = "ExampleAppName";
```

### Varsayılan ad

Önceki metinde uygulama adını herhangi bir şekilde sağlamadıysanız, uygulama adı otomatik olarak yürütülebilir ad (ve sıgacak yolun çoğu) olarak ayarlanır.

## XMS uygulamaları

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

## Bağlantı üreticisi özelliği

XMS uygulamaları, *XMSC.WMQ\_APPLICATIONNAME* özelliği ("*XMSC\_WMQ\_APPNAME*") JMS ile benzer şekilde. En çok 28 karakter belirtebilirsiniz.


Daha fazla bilgi için bkz. "[XMS .NET creating administered objects](#)" sayfa 624 ve "[XMS iletisinin özellikleri](#)" sayfa 632 .

## 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.

 Java ve JMS istemci uygulamaları zaten bir uygulama adı belirtebilir ve bu IBM MQ for Multiplatforms üzerinde MQCNO **App1Name** alanını kullanarak bağ tanımlama uygulamalarına genişletildi.

## Bağlantı özelliği geçersiz kılma değeri

**Application name** özelliği, geçersiz kılabileceğiniz bağlantı özellikleri listesine eklendi. Ek bilgi için [IBM MQ bağlantı özelliği geçersiz kılma özelliğinin kullanılması](#) başlıklı konuya bakın.



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

## Geçersiz Kılınan özellik

**MQC.APPNAME\_PROPERTY** değişmezi *APPNAME* değeriyle tanımlandı. Artık bu özelliği, adın ilk 28 karakterini kullanarak **MQQueueManager** oluşturucusuna geçirebilirsiniz. Ek bilgi için bkz. [IBM MQ classes for Java içinde bağlantı özelliği geçersiz kılma değeri kullanılıyor](#) .

## MQEnvironment

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

Ek bilgi için bkz. "[IBM MQ classes for Java için IBM MQ ortamını ayarlama](#)" sayfa 346 .

## Java sınıf adı

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

Ek bilgi için bkz. "[IBM MQ classes for Java için IBM MQ ortamını ayarlama](#)" sayfa 346 .

## İlgili kavramlar

"[IBM MQ classes for Java için IBM MQ ortamını ayarlama](#)" sayfa 346

Bir uygulamanın istemci kipinde bir kuyruk yöneticisine bağlanmasını sağlamak için, uygulamanın kanal adını, anasistem adını ve kapı numarasını belirtmesi gerekir.

## İlgili başvurular

[MQCNO](#)

## İletilere ilişkin tasarım teknikleri

Seçiciler ve ileti özellikleri için dikkat edilmesi gereken noktalar da dahil olmak üzere, iletileri tasarlamaya yardımcı olacak dikkat edilecek noktalar.

## Tasarım aşamasında göz önünde bulundurulması gereken şeyler

İletiyi kuyruğa koymak için bir MQI çağrısı kullandığınızda bir ileti oluşturursunuz. Aramaya giriş olarak, bir *ileti tanımlayıcısı* (MQMD) ve başka bir programa göndermek istediğiniz veriler için bazı denetim bilgileri

sağlanmanız gerekir. Ama tasarım aşamasında, aşağıdakileri göz önünde bulundurmanız gerekir, çünkü onlar sizin mesajlarınızı nasıl oluşturmanızı etkiler:

### **Kullanılacak ileti tipi**

Bir ileti gönderebileceğiniz basit bir uygulama tasarlıyor musunuz, daha sonra başka bir işlem yapmak zorunda kalmadınız mı? Yoksa bir soruyla cevap mı istiyorsun? Bir soru soruyorsanız, iletiyi almak istediğiniz kuyruğun adını ileti tanımlayıcısına dahil edebilirsiniz.

İsteğinizin ve yanıt iletilerinin zamanuyumlu olmasını istiyor musunuz? Bu, yanıtta yanıtlamak için bir zaman aşımı süresi ayarladığınızı belirtir; yanıtı bu süre içinde almezseniz, yanıt bir hata olarak işlenir.

Ya da işlemlerinizin ortak zamanlama sinyalleri gibi belirli olayların ortaya çıkmalarına bağlı kalmaması için zamanuyumsuz olarak çalışmayı tercih eder misiniz?

Diğer bir dikkat, tüm mesajlarınızın bir çalışma birimi içinde olup olmamasıdır.

### **İletilere farklı öncelikler atama**

Her iletiye bir öncelik değeri atayabilir ve kuyruk tanımlayabilir ve böylece, iletilerin öncelikleri sırasına göre iletileceğini belirler. Bunu yapmazsanız, başka bir program kuyruktan bir ileti aldıklarında, her zaman en yüksek önceliğe sahip iletiyi alır. Kuyruk, iletilerini öncelik sırasına göre korumuyorsa, kuyruktan ileti alan bir program, iletileri kuyruğa eklendikleri sırayla alır.

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

### **Kuyruklardaki kuyruk yöneticisinin yeniden başlatılmasına etkisi**

The queue manager preserves all persistent messages, recovering them when necessary from the IBM MQ log files, when it is restarted. Kalıcı olmayan iletiler ve geçici dinamik kuyruklar korunmaz. Atılmamasını istemediğiniz iletiler, yaratıldıklarında kalıcı olarak tanımlanmalıdır. When writing an application for IBM MQ for Windows or IBM MQ on UNIX and Linux systems, make sure that you know how your system has been set up in respect of log file allocation to reduce the risk of designing an application that will run to the log file limits.

**z/OS** Paylaşılan kuyruklardaki iletiler (yalnızca IBM MQ for z/OS üzerinde kullanılabilir) bağlaşım tesisinde tutulur (CF), kalıcı olmayan iletiler, CF kullanılabilir kaldığı sürece bir kuyruk yöneticisinin yeniden başlatma işlemlerinde korunur. CF başarısız olursa, kalıcı olmayan iletiler kaybedilir.

### **İletilerin alıcısına kendinizle ilgili bilgiler verme**

Genellikle, kuyruk yöneticisi kullanıcı kimliğini ayarlar, ancak uygun şekilde yetkili uygulamalar bu alanı da ayarlayabilir; böylece, kendi kullanıcı kimliğinizi ve alma programının hesap ya da güvenlik amacıyla kullanabileceği diğer bilgileri de ekleyebilirsiniz.

### **Alma kuyruklarının miktarı**

**Multi** Bir iletinin birkaç kuyruğa konması gerekiyorsa, bir konu ya da dağıtım listesine yayınlayabilirsiniz.

**z/OS** Bir iletinin birkaç kuyruğa konması gerekiyorsa, bir konuya yayınlayabilirsiniz.

### **Seçiciler ve ileti özellikleri**

İletiler, ana ileti bilgi yükünün yanında kendileriyle ilişkili meta verileri olabilir. Bu ileti özellikleri, ek veriler sağlanmasında yararlı olabilir.

Bu ek verilerin iki yönü vardır; bu bilgiler aşağıdakileri bilmeniz açısından önemlidir:

- Özellikler, Advanced Message Security (AMS) korumasının konusu değildir. Verilerinizi korumak için AMS 'yi kullanmak istiyorsanız, ileti özelliklerini değil, bilgi yükünün içine yerleştirin.
- Bu özellikler, iletilerin seçilmesini gerçekleştirmek için kullanılabilir.

Seçiciler kullanıldığında ilk olarak ilk olarak standart ileti konvansiyonunun kırıldığını belirtmek önemlidir. Kuyruk yöneticisi bu iş yükü için en iyi duruma getirildiğinden, performans nedenleri için karmaşık seçiciler önerilmiyor. Kuyruk yöneticisi, ileti özelliklerinin dizinlerini saklamaz, bu nedenle bir ileti

aranması doğrusal bir arama olmalıdır. Kuyruk ne kadar derinse, seçici o kadar karmaşık ve seçicinin bir iletiyle eşleşen daha düşük olasılık performansı olumsuz yönde etkileyecektir.

If complex selection is required, it is suggested to filter the messages by using any application or processing engine, such as IBM Integration Bus, to different destinations. Diğer bir seçenek olarak, bir konu sıradüzeninin kullanılması yararlı olabilir.

**Not:** IBM MQ classes for Java do not support the use of selectors, if you do wish to use selectors these should be done via the JMS API.

## Uygulama tasarımı ve performansı ile ilgili önemli noktalar

Düşük program tasarımının performansı etkileyebileceği bir dizi yol vardır. Bu durum, programın kendisini iyi bir şekilde gerçekleştirebileceği, ancak diğer görevlerin başarımını olumsuz yönde etkileyebileceği için algılaması zor olabilir. Bu konuda IBM MQ çağrılarını yapan programlara özgü bazı sorunlar açıklanmaktadır.

Verimli uygulamaları tasarlamaya yardımcı olacak birkaç fikir aşağıda bulunur:


- Uygulamanızı, bir kullanıcının düşünme süresiyle paralel olarak işlenmek üzere tasarlayın:
  - Bir pano görüntüleyin ve uygulamanın başlatılmasına devam ederken kullanıcının yazmaya başlamasını sağlayın.
  - Gereksinim duyduğunuz verileri farklı sunuculardan elde edin.
- Sürekli olarak açma ve kapatma, bağlama ve bağlantı kesme yerine bunları yeniden kullanacaklarınız, bağlantıların ve kuyrukların açık kalmasını sağlar.
- Ancak, yalnızca bir ileti yerleştiren bir sunucu uygulaması MQPUT1' i kullanmalıdır.
- Kuyruk yöneticileri, 4 KB ile 100 KB arasındaki iletiler için eniyilenir. Çok büyük iletiler verimsizdir; tek bir 100 MB 'lık iletiden her biri 1 MB' lik 100 ileti göndermek daha iyi olur. Çok küçük mesajlar da verimsiz. Kuyruk yöneticisi, 4 KB ' lik bir iletide olduğu gibi tek byte 'lık bir ileti için de aynı iş miktarını gerçekleştirir.
- İletilerinizi, eşzamanlı olarak kesinleştirilebilecek ya da yedekleyebilmeleri için bir çalışma birimi içinde saklayın.
- Kurtarılabilir olması gerekmeyen iletiler için kalıcı olmayan seçeneği kullanın.
- Bir iletiyi hedef kuyruklara göndermeniz gerekiyorsa, bir dağıtım listesi kullanmayı düşünün.

### İleti uzunluğunun etkisi

Bir iletteki veri miktarı, iletiyi işleyen uygulamanın performansını etkileyebilir. Uygulamanızın en iyi performansını elde etmek için yalnızca bir iletiyle temel verileri gönderin. Örneğin, bir banka hesabını borç istemek için, istemciden sunucu uygulamasına geçilmesi gerekebilecek tek bilgi, hesap numarası ve borç tutarının olduğu anlamına gelir.

### İleti kalıcılığın etkisi

Kalıcı iletiler genellikle günlüğe kaydedilir. Günlüğe kaydetme iletileri, uygulamanızın performansını azaltır, bu nedenle yalnızca temel veriler için kalıcı iletiler kullanın. Kuyruk yöneticisi durdurursa ya da başarısız olursa, bir iletteki veriler atılabilir ise, kalıcı olmayan bir ileti kullanın.

 Kalıcı iletiler için MQPUT ve MQGET işlemleri, işlemleri kaydetmek için yeterli kurtarma günlüğü alanı olmadığında bloke olur. Such a condition is indicated in the queue manager job log by messages CSQJ110E and CSQJ111A. Bu tür koşulların yönetilmesini ve önlenmelerini sağlamak için izleme süreçlerinin yerinde olduğundan emin olun.

### Belirli bir iletiyi arama

MQGET çağrısı genellikle bir kuyruktan ilk iletiyi alır. Belirli bir iletiyi belirtmek için ileti tanımlayıcısındaki ileti ve ilinti tanıtıcılarını (*MsgId* ve *CorrelId*) kullanırsanız, kuyruk yöneticisi bu iletiyi buluncaya kadar



kuyrukta arama yapmak zorundadır. MQGET çağrısının bu şekilde kullanılması, uygulamanızın başarımını etkiler.

## Farklı uzunluklara ilişkin iletileri içeren kuyruklar

Uygulamanız sabit bir uzunluğa ilişkin iletileri kullanamıyorsa, arabellekleri büyüyüp küçültmeyi dinamik olarak tipik ileti boyutuna uygun olarak küçültür. Uygulama, arabellek çok küçük olduğu için başarısız olan bir MQGET çağrısını yayınlarsa, ileti verilerinin boyutu döndürülür. Arabelleğin uygun bir şekilde yeniden boyutlandırılması ve MQGET çağrısının yeniden yayınlanması için uygulamanızı kod ekleyin.

**Not: MaxMsgLength** özneteliğini açık bir şekilde ayarlamadıysanız, varsayılan olarak 4 MB ' dir. Bu, uygulama arabelleği boyutunu etkilemek için kullanılırsa çok verimsiz olabilir.


## Eşitleme noktalarının sıklığı

Eşitleme noktası içinde çok büyük sayıda MQPUT ya da MQGET komutu veren programlar, bunları kesinleştirmeden, performans sorunlarına neden olabilir. Etkilenen kuyruklar, şu anda erişilemez olan iletilerle veri doldurabilir; diğer görevler bu iletileri almak için bekliyor olabilir. Bu, depolama açısından ve ileti almaya çalışan görevlerle bağlantılı iş parçacıkları açısından çıkarımlar içerir.

## MQPUT1 çağrısının kullanılması

Use the MQPUT1 call only if you have a single message to put on a queue. Birden çok ileti koymak istiyorsanız, MQOPEN çağrısını kullanın ve ardından bir dizi MQPUT çağrısını ve tek bir MQCLOSE çağrısını kullanın.

## Kullanıdaki iş parçacıklarının sayısı

 IBM MQ for Windows için, bir uygulama çok sayıda iş parçacığını gerektirebilir. Her kuyruk yöneticisi işlemi, izin verilen uygulama iş parçacığı sayısı üst sınırı olarak ayrılır.

Uygulamalar çok fazla iş parçacığı kullanabilir. Uygulamanın bu olasılığı dikkate alıp almadığını ve bu tür oluş tipini raporlamak veya raporlamak için gerekli önlemleri aldığını göz önünde bulundurun.

## Kalıcı iletileri eşitleme noktası altına koy

Kalıcı iletiler, uyumluluk noktasının altına konulmalı ve altına girmelidir. Bunun nedeni, syncpoint dışında kalıcı bir ileti alınırken, alma işlemi başarısız olursa, iletinin kuyruktan alıp almadığını ve iletinin verilir vermediğini bilmesinin hiçbir yolu yoktur ve ileti de kaybolursa, bu ileti de kaybedilir. Syncpoint altında kalıcı iletiler alınırken, herhangi bir işlem başarısız olursa, hareket geriye işlenir ve kalıcı ileti kuyruğun üzerinde olduğu için kaybolmaz.

Benzer şekilde, kalıcı iletiler yerleştirilirken, bunları eşitleme noktası altına koyun. Syncpoint altında kalıcı iletiler koymanın ve kalıcı iletilerin alınması için başka bir neden de, IBM MQ içindeki kalıcı ileti kodunun uyumluluk noktası için yoğun olarak eniyelenmiş olması. Bu nedenle, eşitleme noktası altındaki kalıcı iletilerin yerleştirilip alınması, uyumluluk noktası dışında kalıcı iletiler koymaktan ve alkoymaktan daha hızlıdır.

Uygulamanız syncpoint dışında kalıcı iletiler ortaya koyarsa, kuyruk yöneticisi uygulama adına bir örtük eşitleme noktası yaratıp yaramadığını denetler. Kuyruk yöneticisi bunu yapabiliyorsa, o eşitleme noktasının içine yerleştirmeyi içerir ve otomatik olarak kesinleştirebilir. Daha ayrıntılı bir açıklama için [“Çoklu Platformlar üzerindeki örtük Syncpoint” sayfa 896 ' e bakın.](#)

Ancak, IBM MQ içindeki kalıcı olmayan kod, syncpoint dışında olmak üzere eniyelendiği için, syncpoint dışında kalıcı olmayan iletiler koymak ve bu iletileri almak daha hızlıdır. Kalıcı ileti diske kalıcı kılındığından, kalıcı iletilerin disk hızlarına alınması ve disk hızlarında alınması. Bununla birlikte, kalıcı olmayan iletilerin yerleştirilip alınması ve ilgili disk yazma işlemi olmadığı için, syncpoint kullanılırken bile işlem yapılmadığı için CPU hızlarına geçilir.

Bir uygulama iletileri alıyorsa ve bunların kalıcı olup olmadığını önceden bilmiyorsa, GMO seçeneği MQGMO\_SYNCPOINT\_IF\_PERSISTENT seçeneği kullanılabilir.


## Gelişmiş uygulamalar için tasarım teknikleri

Daha gelişmiş uygulamalar tasarlarken, iletileri beklemek, yanıtları ilintilendirmek, bağlam bilgilerini ayarlamak ve kullanmak, uygulamaları otomatik olarak başlatmak, raporlar oluşturmak ve kümeleme kullanırken ileti yakınlıkları kaldırmak gibi bazı teknikler vardır.

Basit bir IBM MQ uygulaması için, uygulamanıza hangi IBM MQ nesnelere ve hangi ileti türlerini 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 işlemi yaparak iletileri bekleyebilirler:

- Bir ileti gelene kadar bekleme ya da belirtilen bir zaman aralığı kullanım süresi doluyor (bkz. [“İleti bekleniyor” sayfa 833](#)).
-  Yalnızca IBM MQ for z/OS üzerinde, bir ileti geldiğinde programın bilgilendirilebilmesi için bir sinyal ayarlayın. Daha fazla bilgi için bkz [“Sinyalizasyon” sayfa 834](#).
- Bir ileti geldiğinde yönlendirilecek bir geri bildirim çıkışı oluşturma; bkz. [“IBM MQ iletilerinin zamanuyumsuz tüketimi” sayfa 39](#).
- Bir iletinin geldiğini görmek için kuyruğun düzenli olarak çağrılması (*yoklama*). Bu, tipik olarak önerilmez, çünkü performans sonuçları olabilir.

### Yanıt ilintilendirme

IBM MQ uygulamalarında, bir program bunu iş yapmak için istekte bulunan bir ileti aldığı anda, program genellikle istekte bulunana bir ya da daha fazla yanıt iletisi gönderir.

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

### Bağlam bilgilerinin ayarlanması ve kullanılması

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

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

Bağlam bilgilerini kullanma ve ayarlama hakkında daha fazla bilgi için bkz. [“İleti bağlamı” sayfa 44](#).

### IBM MQ programlarını otomatik olarak başlatma

Bir kuyruğa ileti geldiğinde otomatik olarak bir program başlatmak için IBM MQ *tetikleme* seçeneğini kullanın.

Bir programın bu kuyruğu işlemeye başlaması için, tetikleme koşullarını bir kuyruğun üzerinde ayarlayabilirsiniz:

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

Tetikleme hakkında daha fazla bilgi için bkz. [“Starting IBM MQ applications using triggers” sayfa 899](#).

Tetikleme, bir programı otomatik olarak başlatmanın tek bir yolu. Örneğin, IBM MQ dışı olanakları kullanan bir süreölçerin üzerinde otomatik olarak bir program başlatabilirsiniz.

Multiplatforms üzerinde IBM MQ , kuyruk yöneticisi başlatıldığında IBM MQ programlarını başlatmak için hizmet nesnelere tanımlayabilir; bkz. [Hizmet nesnelere](#).

## IBM MQ raporları oluşturma

Bir uygulama içinde aşağıdaki raporları isteyebilirsiniz:

- Özel durum raporları
- Süre bitimi raporları
- Varışta doğrulama (COA) raporları
- Teslim edilme (COD) raporları
- Pozitif işlem bildirim (PAN) raporları
- Negatif işlem bildirim (NAN) raporları

Bunlar [“Rapor iletileri” sayfa 18](#) içinde açıklanmaktadır.

## Kümeler ve ileti zenginlikleri

Aynı kuyruk için birden çok tanımı olan kümeleri kullanmaya başlamadan önce, ilgili ileti alışverişi gerektiren herhangi bir uygulama olup olmadığını görmek için uygulamalarınızı inceleyin.

Bir küme içinde, bir ileti, uygun kuyruğun bir örneğini barındıran herhangi bir kuyruk yöneticisine yönlendirilebilir. Bu nedenle, ileti zenginlikleri olan uygulamaların mantığı bozunabilir.

Örneğin, soru ve yanıt formlarında aralarında akan bir dizi ileti dizisine dayanan iki uygulamanız olabilir. Tüm soruların aynı kuyruk yöneticisine gönderilmesi ve tüm yanıtların diğer kuyruk yöneticisine geri gönderildiği önemli olabilir. Bu durumda, iş yükü yönetimi yordamında, iletileri uygun kuyruğun bir eşgörünümüne ev sahipliği yapmak üzere olan herhangi bir kuyruk yöneticisine göndermemesi önemlidir.

Mümkün olan yerlerde, kötülerini ortadan kaldırın. İleti zenginliğinin kaldırılması, uygulamaların kullanılabilirliğini ve ölçeklenebilirliğini artırır.

Daha fazla bilgi için bakınız: [Handling message affinities](#).

## IBM i uygulamaları için tasarım ve performans konuları

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

Bu bilgiler iki bölüme ayrılır:

- [“Uygulama tasarımı konuları” sayfa 59](#)
- [“Belirli başarımlar sorunları” sayfa 60](#)

## Uygulama tasarımı konuları

Düşük program tasarımının performansı etkileyebileceği bir dizi yol vardır. Bu sorunlar, programın düzgün bir şekilde, diğer görevlerin başarımını olumsuz yönde etkileyebileceği için algılaması zor olabilir. IBM MQ for IBM i çağrılarını yapan programlara özgü bazı sorunlar aşağıdaki bölümlerde açıklanır.

Uygulama tasarımıyla ilgili daha fazla bilgi için bkz. [“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#).

## İleti uzunluğunun etkisi

IBM MQ for IBM i , iletilerin 100 MB ' ye kadar veri tutmasına olanak tanısa da, bir iletteki veri miktarı, iletiyi işleyen uygulamanın performansını etkiler. Uygulamalarınızdan en iyi performansı elde etmek için, yalnızca temel verileri bir iletiye gönderin; örneğin, bir banka hesabını borç isteğine göre, istemciden sunucu uygulamasına geçilmesi gerekebilecek tek bilgi, hesap numarası ve borç tutarı.

## İleti kalıcılığın etkisi

Kalıcı iletiler günlüğe kaydedildi. Günlük kaydı iletileri, uygulamanızın performansını azaltır, bu nedenle yalnızca temel veriler için kalıcı iletiler kullanın. Kuyruk yöneticisi durdurursa ya da başarısız olursa, bir iletteki veriler atılabilir ise, kalıcı olmayan bir ileti kullanın.

## Belirli bir iletiyi arama

MQGET çağrısı genellikle bir kuyruktan ilk iletiyi alır. Belirli bir iletiyi belirtmek için ileti tanımlayıcısındaki ileti ve ilinti tanıtıcılarını (*MsgId* ve *CorrelId*) kullanırsanız, kuyruk yöneticisinin bu iletiyi buluncaya kadar kuyrukta arama olması gerekir. MQGET çağrısının bu şekilde kullanılması, uygulamanızın başarımını etkiler.

## Farklı uzunluklara ilişkin iletileri içeren kuyruklar

If the messages on a queue are of different lengths, to determine the size of a message, your application can use the MQGET call with the *BufferLength* field set to zero so that, even though the call fails, it returns the size of the message data. Daha sonra, uygulama, ilk çağrısında ölçülen iletinin tanıtıcısını ve doğru büyüklüğün arabelleğiyle belirtilen tanıtıcıyı belirterek çağrıyı yineleyebilir. Ancak, aynı kuyruğa hizmet eden başka uygulamalar varsa, ikinci MQGET çağrısı, iki aramanızın arasında başka bir uygulamanın aldığı bir iletiyi ararken zaman harcadığı için, uygulamanızın performansını azaldığını da bulabilirsiniz.

Uygulamanız sabit uzunluktaki iletileri kullanamıyorsa, bu soruna başka bir çözüm, kuyruğun kabul edeceği ileti büyüklüğü üst sınırını bulmak için MQINQ çağrısını kullandıktan sonra MQGET çağrılarınızda bu değeri kullanın. Bir kuyruğa ilişkin ileti büyüklüğü üst sınırı, kuyruğun **MaxMsgLen** öznelitesinde saklanır. Ancak bu yöntem, bu kuyruk öznelitesinin değeri IBM MQ for IBM tarafından izin verilen üst sınır olabileceği için, 2 GB ' den büyük olabileceğinden, büyük miktarda depolama alanı kullanılabilir.

## Eşitleme noktalarının sıklığı

Eşitleme noktası içinde çok sayıda MQPUT çağrısı yapan programlar, bunları kesinleştirmeden, performans sorunlarına neden olabilir. Etkilenen kuyruklar, şu anda kullanılamaz durumda olan iletileri doldurabilir, ancak diğer görevler bu iletileri almak için bekliyor olabilir. Bu sorun, depolama açısından ve ileti almaya çalışan görevlerle bağlantılı iş parçacıkları açısından çıkarımlar içerir.

## MQPUT1 çağrısının kullanılması

Use the MQPUT1 call only if you have a single message to put on a queue. Birden çok ileti koymak istiyorsanız, MQOPEN çağrısını kullanın ve ardından bir dizi MQPUT çağrısını ve tek bir MQCLOSE çağrısını kullanın.

## Kullanıdaki iş parçacıklarının sayısı

Bir uygulama birçok iş parçacığını gerektirebilir. Her kuyruk yöneticisi işlemine izin verilen en fazla sayıda iş parçacığı ayrılır. Bazı uygulamalar sorun çıkarırsa, çok fazla iş parçacığı kullanan tasarımlarından kaynaklanıyor olabilir. Uygulamanın bu olasılığı dikkate almadığını ve bu tür oluş tipini raporlamak veya raporlamak için gerekli önlemleri aldığını göz önünde bulundurun. IBM i ' in izin verdiği iş parçacığı sayısı üst sınırı 4,095 'dir. Ancak, varsayılan değer 64 'tür. IBM MQ , süreçlerine en çok 63 iş parçacığı sağlar.

## Belirli başarımlar sorunları

Bu bölümde, depolama ve kötü performans sorunları açıklanmaktadır.

### Depolama sorunları

If you receive the system message CPF0907. Serious storage condition may exist it is possible that you are filling up the space associated with the IBM MQ for IBM i queue managers.

### Uygulamanız ya da IBM MQ for IBM i yavaş çalışıyor mu?

Uygulamanız yavaş çalışıyorsa, bunun bir döngü içinde olduğunu belirtebilir ya da kullanılabilir olmayan bir kaynak için beklenebilir. Bu yavaş çalıştırma, bir performans sorunu nedeniyle de ortaya çıkmış olabilir. Belki de sisteminiz kapasitenin limitlerine yakın bir yerde çalışıyor. Bu tip bir sorun genellikle, genellikle sabah ve öğleden sonra saatlerinde, sistem yükleme sürelerinin en yüksek yoğunlukta olduğu en yüksek sıklık sorundur. (Ağınız birden çok saat dilimine geçiyorsa, sistem yükü en yüksek değeri başka bir zamanda gerçekleşmiş olabilir.)

Performans düşüşünün sistem yüklenmesine bağlı olmadığını bulursanız, ancak bazen sistem hafifçe yüklendiğinde kötü tasarlanmış bir uygulama programı büyük olasılıkla suçlanır. Bu sorun, yalnızca belirli kuyruklara erişildiğinde ortaya çıkan bir sorun olarak ortaya çıkar.

QTOTJOB ve QADLTOTJ, araştırmaya değer sistem değerleridir.

Aşağıdaki belirtiler, IBM MQ for IBM i 'nin yavaş çalıştığını gösterebilir:

- Sisteminiz, MQSC komutlarına yanıt verebilmek için yavaşsa.
- Kuyruk derinliğinin yinelenmesi, kuyruğun çok miktarda kuyruk etkinliği olmasını beklediğiniz bir uygulama için yavaşça işlenmekte olduğunu belirtir.
- IBM MQ izleme çalışıyor mu?

## **Linux** Linux on POWER Systems - Little Endian uygulamaları için dikkat edilmesi gereken noktalar

Linux on POWER Systems - Little Endian , yalnızca 64 bit uygulamaları desteklediğinden, 32 bit uygulamalar için IBM MQ 'de herhangi bir destek sağlanmaz.

### **İlgili kavramlar**

[“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#)

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğinize karar verdiğinizde, IBM MQ tarafından sunulan özelliklerin nasıl kullanılacağına karar vermeniz gerekir.

## **z/OS** z/OS uygulamaları için tasarım ve performans konuları

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

Düşük program tasarımının performansı etkileyebileceği bir dizi yol vardır. Bu sorunlar, programın düzgün bir şekilde, diğer görevlerin başarımını olumsuz yönde etkileyebileceği için algılaması zor olabilir. MQI çağrılarını yapan programlara özgü bazı sorunlar aşağıdaki bölümlerde gösterilir.

Uygulama tasarımıyla ilgili daha fazla bilgi için bkz. [“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46.](#)

### **İleti uzunluğunun etkisi**

IBM MQ for z/OS , iletilerin 100 MB 'ye kadar veri tutmasına olanak tanısa da, bir iletteki veri miktarı, iletiyi işleyen uygulamanın performansını etkiler. Uygulamanızın en iyi performansını elde etmek için yalnızca bir iletiyle temel verileri gönderin. Örneğin, bir banka hesabını borç istemek için, istemciden sunucu uygulamasına geçilmesi gerekebilecek tek bilgi, hesap numarası ve borç tutarının tutarı.

### **İleti kalıcılığın etkisi**

Kalıcı iletiler günlüğe kaydedilir. Günlüğe kaydetme iletileri, uygulamanızın performansını azaltır, bu nedenle yalnızca temel veriler için kalıcı iletiler kullanın. Kuyruk yöneticisi durdurursa ya da başarısız olursa, bir iletteki veriler atılabilir ise, kalıcı olmayan bir ileti kullanın.

Kalıcı iletilere ilişkin veriler, günlük arabelleklerine yazılır. Bu arabellekler, aşağıdaki durumlarda günlük verileri kümelerine yazılır:

- Kesinleştirme gerçekleşir
- Bir ileti alındı ya da syncpoint dışında kaldı
- WRTHRSR arabellekleri doldurulur

Bir iş birimi içindeki birçok iletinin işlenmesi, iletilerin her iş birimi için ya da syncpoint dışında bir ileti işlenmesinden daha az giriş/çıkış oluşmasına neden olabilir.

## Belirli bir iletiyi arama

MQGET çağrısı genellikle bir kuyruktan ilk iletiyi alır. İleti ve ilinti tanıtıcılarını kullanıyorsanız ( **MsgId** ve **CorrelId** ) belirli bir iletiyi belirtmek için ileti tanımlayıcısında, kuyruk yöneticisi bu iletiyi buluncaya kadar kuyruktaki arama yapar. Belirli bir iletiyi bulmak için, IBM MQ ' un bu şekilde MQGET kullanılması, belirli bir iletiyi bulmak için, tüm kuyruğu taramak zorunda kalabileceği için uygulamanızın performansını etkiler.

Kuyruk yöneticisinin kuyruklardaki MQGET işlemlerinin hızını artırmak için kullanılabilecek bir dizini korumasını istediğinizi belirtmek için **IndexType** kuyruk özneliğini kullanabilirsiniz. Ancak, bir dizini korumak için küçük bir başarımla azaltma işlemi vardır, bu nedenle yalnızca bir dizini kullanmanız gerekiyorsa bir dizin oluşturun. İleti tanıtıcılarının ya da ilinti tanıtıcılarının dizinini oluşturmayı seçebilir ya da iletilerin sırayla alındığı kuyruklar için dizin oluşturulmamayı seçebilirsiniz. Birçok farklı anahtar değeri bulunmaya çalışın, aynı değere sahip bir çok değer değil. For example Balance1, Balance2, and Balance3, not three with Balance. Paylaşılan kuyruklar için doğru **IndexType** ' e sahip olmanız gerekir. **IndexType** kuyruk özneliğine ilişkin ayrıntılar için bkz. [IndexType](#).

Dizinlenmiş kuyrukları kullanarak kuyruk yöneticisini yeniden başlatma süresini etkilememek için, CSQ6SYSP makrosu içindeki QINDXBLD (NOWAST) parametresini kullanın. Bu, kuyruk yöneticisinin yeniden başlatılmasını, kuyruk dizin oluşturma işleminin tamamlanmasını beklemeden yeniden başlatmasını sağlar.

**IndexType** özneliğinin ve diğer nesne özneliklerinin tam açıklaması için [Nesnelerin özneliklerini kullanın](#).

## Farklı uzunluklara ilişkin iletileri içeren kuyruklar

İletinin beklenen büyüklüğünün eşleşen bir arabellek büyüklüğünü kullanarak bir ileti alın. İletinin çok uzun olduğunu belirten bir dönüş kodu alırsanız, daha büyük bir arabellek alın. Alma işlemi bu şekilde başarısız olduğunda, döndürülen veri uzunluğu dönüştürülemez ileti verilerinin büyüklüğüdür. MQGET çağrısında MQGMO\_CONVERT değerini belirlerseniz ve dönüştürme sırasında veriler genişleirse, arabellenin büyüklüğünü daha fazla artırmanız gerektiğinde, arabelleğe sığmayabilir.

Arabellek uzunluğu sıfır olan MQGET komutunu verdiyseniz, ileti büyüklüğünü döndürür ve uygulama bu büyüklükte bir arabellek alabilir ve alma işlemini yeniden yayınlayabilir. Kuyruğu işleyen birden çok uygulamanız varsa, özgün uygulama alma işlemini yeniden yayınlarken başka bir uygulama iletiyi önceden işlemiş olabilir. Sık sık büyük iletiler varsa, bu iletiler için büyük bir arabelleğe sahip olması ve ileti işlendikten sonra serbest bırakmanız gerekebilir. Bu, tüm uygulamalarda büyük arabellekler varsa, sanal saklama alanı sorunlarının azaltılmasına yardımcı olur.

If your application cannot use messages of a fixed length, another solution to this problem is to use the MQINQ call to find the maximum size of messages that the queue can accept, then use this value in your MQGET call. Bir kuyruğa ilişkin ileti büyüklüğü üst sınırı, kuyruğun **MaxMsgL** öznelisinde saklanır. This method could use large amounts of storage, however, because the value of **MaxMsgL** could be as high as 100 MB, the maximum allowed by IBM MQ for z/OS.

**Not:** Kuyruğa büyük iletiler konduktan sonra **MaxMsgL** parametresini daha da düşürebilirsiniz. Örneğin, 100 MB 'lik bir ileti yerleştirebilir ve **MaxMsgL** ' ı 50 bayt olarak ayarlayabilirsiniz. Bu, beklenen uygulamadan daha büyük iletiler elde etmek için hala mümkün olduğu anlamına gelir.

## Eşitleme noktalarının sıklığı

Eşitleme noktası içinde birçok MQPUT çağrısı yapan programlar, bunları kesinleştirmeden, performans sorunlarına neden olabilir. Etkilenen kuyruklar, şu anda kullanılamaz durumda olan iletileri doldurabilir, ancak diğer görevler bu iletileri almak için bekliyor olabilir. Bunun, depolama alanı açısından ve ileti almaya çalışan görevlerle bağlantılı iş parçacıkları açısından etkileri vardır.

Bir kural olarak birden çok uygulamanız varsa, kuyruğun işlenmesi sırasında genellikle en iyi başarıyı elde etmiş olur.

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

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

Bir görevin alabileceği ileti sayısını sınırlayabilir ya da **MAXUMSGS** kuyruk yöneticisi özniteliğinde tek bir kurtarma birimi içinde yerleştirebilirsiniz. Bu özniteliğe ilişkin bilgi için [MQSC komutları](#) içindeki **ALTER QMGR** komutuna bakın.

## MQPUT1 çağrısının avantajları

Kuyruğa koymak için tek bir iletiniz varsa MQPUT1 çağrısını kullanın. Birden çok ileti koymak istiyorsanız, MQOPEN çağrısını ve ardından bir dizi MQPUT çağrısını ve tek bir MQCLOSE çağrısını 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 (sayfa kümesi 0 ve sayfa kümesi 1, sistem ile ilgili nesnelere ve kuyruklar için önerilir). Genişletilmiş biçimiyle bir sayfa kümesi kullanabilir ve bir sayfa kümesinin kapasitesini artırabilirsiniz.

### Paylaşılan Kuyruklar

Paylaşılan kuyruklar kapasitesi, bağlantım tesisinin (CF) büyüklüğüne bağlıdır. IBM MQ , temel depolama birimlerinin giriş ve öge olduğu CF liste yapılarını kullanır. Her ileti 1 giriş olarak saklanır ve ilişkili MQMD ve diğer ileti verilerini içeren birden çok öge saklanır. Tek bir ileti tarafından tüketilen öğelerin sayısı, iletinin büyüklüğüne ve CFLEVEL (5) için, MQPUT zamanındaki etkide bulunan kural dışı yükleme kurallarıyla birlikte değişir. İleti verileri Db2 ya da SMDS ' ye doldurulduğunda daha az öge gerekir. İleti doldurulduğunda ileti verileri erişimi daha yavaş olur. İleti boşaltma ile ilişkili başarımlar ve CPU ek yüklerinin daha fazla karşılaştırması için Performans Desteği MP1H ' ye bakın.

## Performansı etkileyen şey

Performans, iletilerin ne kadar hızlı işlenebileceğini ifade edebilir ve ileti başına ne kadar CPU gerektiği anlamına da gelebilir.

### Hızlı iletilerin nasıl işlenebileceğini etkileyen

Kalıcı iletiler için en büyük etki, günlük veri kümelerinin hızıdır. Günlük veri kümelerinin hızı, üzerinde oldukları DASD ' ye bağlıdır. Bu nedenle, çekilmeyi azaltmak için, kullanılmış birimlerde günlük verileri kümesi koymak için özen gösterilebilir. G/Ç başına yazılan birden çok sayfa varsa, MQ günlüklerinin paylaşım işlemi, günlük başarımlarını artırır. Z High Performance Fibre connection (zHPF), G/Ç altsistemi meşgul olduğunda G/Ç yanıt süresi için önemli bir başarıma sahiptir.

Bir iletiyi almak ve yerleştirmek için bir istek olduğunda, kuyruğun bütünlüğünü korumak için istek sırasında kuyruğa erişim kilitlenir. Planlama amacıyla, kuyruğun tüm istek için kilitlenmesini dikkate alın. Yani, eğer bir put 100 mikrosaniye sürerse ve saniyede 10.000 'den fazla talep varsa, gecikmeler yaşayabilirsiniz. Pratikte bundan daha iyisini başarabilirsiniz, ama bu iyi bir geneldir. Başarımlarını artırmak için farklı kuyruklar kullanabilirsiniz.

Bunun olası nedenleri şunlar olabilir:

- Her CICS işleminin kullandığı ortak bir yanıt kuyruğu kullanır.

- Her CICS işlemine, kuyruğa benzersiz bir yanıt verilir
- a reply to a queue for CICS region and all transactions in the CICS region use this queue.

Yanıt, ikinci bir istek sayısına ve isteklerin yanıt sürmesine bağlıdır.

İletilerin bir sayfa kümesinden okunması gerekiyorsa, bu iletiler arabellek havuzundaki iletilerin karşılaştırılması sırasında daha yavaş olur. Arabellek havuzuna sığmaktan daha fazla iletiniz varsa, bunlar diske dökülecektir. Bu nedenle, arabellek havuzunun kısa ömürlü iletiler için yeterince büyük olduğundan emin olmanız gerekir. Uzun saatler sonra işlediğiniz iletiler varsa, bunlar büyük olasılıkla diske dökülebilirler; bu nedenle, bu iletilerin arabellek havuzundaki iletilerden daha yavaş olmasını beklemeniz gerekir.

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

MQ istekleriniz çok sayıda CPU kullanırsa, bu, iletilerin ne kadar hızlı işleneceğini etkileyebilir. Mantıksal Bölüm (LPAR) CPU için sınırlandırıldıysa, CPU ' lar beklemeyi geciktirir.

### İleti başına CPU sayısı

Genel olarak daha büyük iletiler için daha fazla CPU kullanın, bu nedenle mümkünse büyük (x MB) iletilerdir.

Kuyruklardan belirli iletiler alınırken, kuyruk yöneticisinin doğrudan iletiye gidebilmesi için kuyruk dizinlenmelidir (ve kuyruğun tüm taramasını engelleyebilir). Kuyruk dizinlenmediyse, kuyruk, iletiyi ararken kuyruktan taranır. Kuyruğun üzerinde 1000 ileti varsa, 1000 iletinin tümünü taramak zorunda kalabilir. Sonuç, çok fazla gereksiz CPU kullanımına sahip.

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

MQ V7 'de, **CORRELID** ya da **MSGID** ' e ek olarak bir seçici dizgisine göre iletileri seçebilirsiniz. Her mesajın bakılması gerekir. Bu nedenle, kuyrukte bu kadar çok sayıda ileti varsa, bu iletiler pahalı olur.

Bir uygulamanın OPEN PUT CLOSE TO PUT1 PUT1 ' den daha verimli bir uygulama için daha verimli bir uygulama.

### CICS içinde tetikleme

Tetiklenen bir kuyruğa ilişkin ileti geliş hızı düşük olduğunda, önce tetikleyiciyi kullanmak etkili olur. İleti varış hızı saniyede 10 'dan fazla ileti olduğunda, ilk hareketi tetiklemek daha verimli olur, daha sonra işlem bir ileti işleme işlemi alır ve sonraki iletiyi alır ve bu şekilde devam eder. Bir ileti kısa bir süreye ( 0.1 ile 1 saniye arasında) gelmediyse, işlem sona erer. Yüksek iş hacminde, iletileri işlemek ve iletilerin bir şekilde toplanmasını önlemek için birden çok işlemin çalışması gerekebilir. Üretilen her tetikleyici iletisinde, bu ileti bir kont ve bir tetikleyici iletisi alma gerektirir; bu ileti, iletinin maliyetini ikiye katlayan bir ileti alır.

### Kaç bağlantı ya da eşzamanlı kullanıcı destekleniyor

Her bağlantı, kuyruk yöneticisi içindeki sanal saklama alanını kullanır, böylece daha fazla eşzamanlı kullanıcı daha fazla depolama alanı kullanır. 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 süre kuyruk yöneticisinden bilgileri önbelleğe alır. Kuyruk yöneticisi içinde kullanılan sanal saklama alanı miktarı etkilenir.

**CHINIT** , yaklaşık 10.000 bağlantıya kadar destek verebilir. Bu, sanal saklama alanı ile sınırlanmıştır. Bir bağlantı daha fazla depolama kullanıyorsa (örneğin, TLS ' nin kullanımı), bağlantı başına saklama alanı artar, bu nedenle **CHINIT** daha az bağlantı destekleyebilir anlamına gelir. If you are processing large messages, these will require more storage for buffers in the **CHINIT**, so the **CHINIT** can support less messages.

Uzak kuyruk yöneticisine yönelik bağlantılar, istemci bağlantılarından daha verimlidir. Örneğin, her MQ istemcisi isteği iki ağ akışı gerektirir (biri istek için, diğeri yanıt için). Uzak kuyruk yöneticisine



bir kanalla, bir yanıt geri gelmeden önce ağ üzerinden 50 gönderilebilecek. Büyük bir istemci ağı düşünürseniz, dağıtılmış bir kutuda yoğunlaştırıcı kuyruk yöneticisi kullanılması daha verimli olabilir ve yoğunlaştırıcının içine ve dışına bir kanal girilebilir.

## Başarımı etkileyen diğer konular

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

Bir kanal başlatılırken oluşan önemli bir ek yük vardır. Kanala sık sık başlama ve durmaktan ziyade, bir kanal başlatmak ve onu bağlı bırakmak genellikle 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 köprüsü uygulamaları

This information helps you to write IMS applications using IBM MQ.

- IMS uygulamalarındaki eşitleme noktalarını ve MQI çağrılarını kullanmak için bkz. [“IBM MQ kullanarak IMS uygulamaları yazılıyor” sayfa 66.](#)
- IBM MQ - IMS köprüsünü kullanan uygulamaları yazmak için bkz. [“IMS köprü uygulamaları yazılıyor” sayfa 70.](#)

IBM MQ for z/OS üzerinde IMS ve IMS köprüsü uygulamaları hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- [“IBM MQ kullanarak IMS uygulamaları yazılıyor” sayfa 66](#)
- [“IMS köprü uygulamaları yazılıyor” sayfa 70](#)

### İlgili kavramlar

[“Message Queue Interface-Genel Bakış” sayfa 760](#)

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabılır alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OS üzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

## **IBM MQ kullanarak IMS uygulamaları yazılıyor**

There are further considerations when using IBM MQ in IMS applications These include which MQ API calls can be used and the mechanism used for syncpoint.

IBM MQ for z/OS üzerinde IMS uygulamaları yazılmasına ilişkin ek bilgi için aşağıdaki bağlantıları kullanın:

- [“IMS uygulamalarında eşitleme noktaları” sayfa 66](#)
- [“IMS uygulamalarında MQI çağruları” sayfa 67](#)

## **Kısıtlamalar**

IMS bağdaştırıcısı kullanılarak bir uygulama tarafından IBM MQ API çağrılarının kullanılabilmesi kısıtlamalar vardır.

Aşağıdaki IBM MQ API çağruları, IMS bağdaştırıcısı kullanılarak bir uygulama içinde desteklenmez:

- MQCB
- MQCB\_FUNC
- MQCTL

## **İlgili kavramlar**

[“IMS köprü uygulamaları yazılıyor” sayfa 70](#)

Bu konuda, IBM MQ - IMS köprüsünü kullanmak için uygulamalar yazılmasıyla ilgili bilgiler bulunur.

## **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) çağrısı kullanabilirsiniz. Daha fazla bilgi için IMS belgelerinde [ROLB call](#) başlıklı konuya bakın.

Kuyruk yöneticisi, iki aşamalı bir kesinleştirme protokolünün katılımcısıdır; eşgüdümçü 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 çağruları yapıldığında değil, MQCONN, MQCONNX ve MQOPEN çağruları 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 belirttik 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.

## **IMS uygulamalarında MQI çağruları**

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 ana hatları:

```
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/370 içinde gösterir. Program önce IMS ile, sonra da IBM MQ ile 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 aşılabılır:

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

Bu şunları içerir:

- İleti okuma
- Gerekli güncellemeler işleniyor
- Yanıt konuyor

Tüm iletler 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 yaklaşılabılır. MQI yanıt iletleri, işlenmekte olan MQI iletisinin yaratıcısının yetkisiyle birlikte konabilir de, IMS kaynak güncellemelerinin güvenlik etkilerinin dikkatli bir şekilde ele alınması gerekir.

- **MPP çağrısı başına bir iletinin işlenmesi ve kullanılabilir tüm iletleri 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 CMQTMCL.
*
* 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 karmaşık 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

```

## IMS köprü uygulamaları yazılıyor

Bu konuda, IBM MQ - IMS köprüsünü kullanmak için uygulamalar yazılmasıyla ilgili bilgiler bulunur.

IBM MQ - IMS köprüsü hakkında bilgi için bkz. [IMS köprüsü](#).

IBM MQ for z/OS üzerinde IMS köprüsü uygulamaları hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- “[How the IMS bridge deals with messages](#)” sayfa 70
- “[Writing IMS transaction programs through IBM MQ](#)” sayfa 944

### İlgili kavramlar

“[IBM MQ kullanarak IMS uygulamaları yazılıyor](#)” sayfa 66

There are further considerations when using IBM MQ in IMS applications These include which MQ API calls can be used and the mechanism used for syncpoint.

### *How the IMS bridge deals with messages*

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

You must also put your messages on IBM MQ queues that have been defined with a storage class that specifies the XCF group and member name of the target IMS system. Bunlar MQ-IMS köprü kuyrukları ya da basit **köprü** kuyrukları olarak bilinir.

IBM MQ-IMS köprüsü, QSGDISP (QMGR) ile tanımlandıysa ya da NOSHEARE seçeneğiyle birlikte QSGDISP (SHARED) ile tanımlandıysa, köprü kuyruğuna özel giriş erişimi (MQOO\_INPUT\_EXCLUSIVE) gerektirir.

Bir kullanıcının bir IMS uygulamasına ileti göndermeden önce IMS ' ta oturum açması gerekmez. Güvenlik denetimi için, MQMD yapısının *UserIdentifier* alanındaki kullanıcı kimliği kullanılır. Denetleme düzeyi, IBM MQ IMS'a bağlandığında belirlenir ve [IMS köprüsü için uygulama erişim denetimi](#) ' ta açıklanmıştır. Bu, sözde oturum açmanın 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 ayraçlar, [] isteğe bağlı birden çok kesimi temsil eder.
  2. MQMD yapısının *Format* alanını MQIIH yapısını kullanmak için MQFMT\_IMS olarak ayarlayın.
- IMS hareket verilerini içeren iletiler, ancak MQIIH yapısı yok:

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

IBM MQ , LL byte toplamını artı MQIIH (varsa) uzunluğunun ileti uzunluğuna eşit olmasını sağlamak için ileti verilerini doğrular.

IBM MQ - IMS köprüsü, köprü kuyruklarından ileti aldığı anda, bunları aşağıdaki gibi işler.

- İleti bir MQIIH yapısı içeriyorsa, köprü MQIIH (bkz. [MQIIH](#) ), OTMA üstbilgilerini oluşturur ve iletiyi IMS' e gönderir. İşlem kodu, giriş iletilerinde belirtilir. Bu bir LTERM ise, IMS bir DFS1288E iletilisiyle yanıtlanır. Hareket kodu bir komutu gösteriyorsa, IMS komutu yürütür; tersi durumda ileti, hareket için IMS içinde kuyruğa alınır.
- İleti IMS hareket verisi içeriyorsa, ancak MQIIH yapısı yoksa, IMS köprüsü aşağıdaki varsayımları yapar:
  - İşlem kodu, kullanıcı verilerinin 5-12 arasındaki baytlardır.
  - Hareket, etkileşimli olmayan kipte

- Hareket kesinleştirme kipinde 0 (kesinleştirme-sonra-gönder)
- MQMD ' deki *Format* , *MFSMapName* (giriş sırasında) olarak kullanılır
- Güvenlik kipi MQISS\_CHECK ' dir

The reply message is also built without an MQIHL structure, taking the *Format* for the MQMD from the *MFSMapName* of the IMS output.

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

- Commit kipi 0 (COMMIT\_THEN\_SEND) kullanan tüm iletiler için synchronized Tpipe kullanılır (bu program, IMS /DIS TMEMBER istemci TPIPE xxxx komutunun durum alanında SYN ile gösterilir)
- Kesinleştirme kipi 1 (SEND\_THEN\_COMMIT) kullanan tüm iletiler için uyumlulaştırılmamış bir Tpipe kullanılır

Tpipes, ilk kullanılanlarda IBM MQ tarafından oluşturulur. IMS yeniden başlatılıncaya kadar, uyumlulaştırılmamış bir Tpipe vardır. IMS soğuk çalışmaya başlayıncaya kadar, eşitlenmiş Tpipes var. Bu Tpipes 'i kendiniz silemezsiniz.

IBM MQ - IMS köprüsünün iletilerle nasıl ilgilenileceğini konu ile ilgili daha fazla bilgi için aşağıdaki konulara bakın:

- [“IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi” sayfa 71](#)
- [“İleti, IMS kuyruğuna konulamazsa” sayfa 72](#)
- [“IMS köprü geribildirim kodları” sayfa 72](#)
- [“IMS köprüsünden gelen iletilerde MQMD alanları” sayfa 72](#)
- [“IMS köprüsünden gelen iletilerde MQIHL alanları” sayfa 73](#)
- [“IMS' dan gelen yanıt iletileri” sayfa 74](#)
- [“IMS hareketlerinde diğer yanıt PCB ' lerinin kullanılması” sayfa 75](#)
- [“İstenmeyen iletiler IMS' den gönderiliyor” sayfa 75](#)
- [“İleti bölümlenmesi” sayfa 75](#)
- [“Data conversion for messages to and from the IMS bridge” sayfa 75](#)

### İlgili kavramlar

“Writing IMS transaction programs through IBM MQ” sayfa 944

The coding required to handle IMS transactions through IBM MQ depends on the message format required by the IMS transaction and the range of responses it can return. Ancak, uygulamanızın IMS ekran biçimlendirme bilgilerini işleyeceği zaman dikkate alınması gereken birkaç nokta vardır.

*IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi*

IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesini açıklayan bir tablo.

<i>Çizelge 4. IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi</i>		
<b>IBM MQ ileti tipi</b>	<b>Commit-then-send (kip 0)- eşitlenmiş IMS Tpipes kullanır</b>	<b>Send-then-commit (kip 1)- eşitlenmemiş IMS Tpipes kullanır</b>
Kalıcı IBM MQ iletileri	<ul style="list-style-type: none"> <li>• Kurtarılabılır tam işlevli hareketler</li> <li>• Kurtarılamaz hareketler IMS tarafından reddedilir</li> </ul>	<ul style="list-style-type: none"> <li>• Fastpath işlemleri</li> <li>• Etkileşimli hareketler</li> <li>• Tam işlevli hareketler</li> </ul>
Kalıcı olmayan IBM MQ iletileri	<ul style="list-style-type: none"> <li>• Kurtarılamaz tam işlevli hareketler</li> <li>• IMS V8 ve APAR PQ61404 ve IMS ürününün sonraki tüm sürümleriyle kurtarılabılır hareketlere izin verilir.</li> </ul>	<ul style="list-style-type: none"> <li>• Fastpath işlemleri</li> <li>• Etkileşimli hareketler</li> <li>• Tam işlevli hareketler</li> </ul>

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

*İleti, IMS kuyruğuna konulamazsa*

İleti, IMS kuyruğuna konamazsa yapılacak işlemler hakkında bilgi edinin.

İleti, IMS kuyruğuna konulamazsa, IBM MQ tarafından aşağıdaki işlem alınır:

- If a message cannot be put to IMS because the message is invalid, the message is put to the dead-letter queue, and a message is sent to the system console.
- İleti geçerliyse, ancak IMStarafından reddedilirse, IBM MQ sistem konsoluna bir hata iletisi gönderir, ileti IMS durum kodunu içerir ve IBM MQ iletisi ölü-mektup kuyruğuna konmaktadır. IMS algılama kodu 001Aise, IMS yanıt kuyruğunda başarısızlığın nedenini içeren bir IBM MQ iletisi gönderir.

**Not:** Daha önce listelenen durumlarda, IBM MQ herhangi bir nedenle iletiyi ölü-mektup kuyruğuna koyamıyorsa, ileti kaynak IBM MQ kuyruğuna geri döndürülür. Sistem konsoluna bir hata iletisi gönderilir ve kuyruktan başka ileti gönderilmez.

İletileri yeniden göndermek için, aşağıdakileri yapın: **bir**

- Kuyruğa karşılık gelen IMS içindeki Tpipes 'ı durdurun ve yeniden başlatın.
- GET ' i (DEVRE dışı) ve yeniden GET (ETKİN) olarak değiştirin.
- IMS ya da OTMA ' yı durdurun ve yeniden başlatın.
- IBM MQ altsisteminizi durdurun ve yeniden başlatın.
- İleti, bir ileti hatasından başka bir şey için IMS tarafından reddedilirse, IBM MQ iletisi kaynak kuyruğa geri döndürülür, IBM MQ kuyruğu işlemeyi durdurur ve sistem konsoluna bir hata iletisi gönderilir.

Bir kural dışı durum raporu iletisi gerekliyse, köprü, iletiyi orijinalinin yetkisiyle yanıtlama kuyruğuna koyar. İleti kuyruğa konulamazsa, rapor iletisi, köprünün yetkisiyle birlikte, ölü-mektup kuyruğuna konmaya başlanır. DLQ ' ya (DLQ) yerleştirilemiyorsa, atılır.

*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 0x001A için 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. a içindeki 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](#).

*IMS köprüsünden gelen iletilerde MQMD alanları*

Learn about the MQMD fields in messages from the IMS bridge.

Kaynak iletinin MQMD ' si, OTMA üstbilgilerinin Kullanıcı Verileri kısmında IMS tarafından taşınır. İleti IMS içinde oluştuysa, bu, IMS Hedef Çözüm Çıkışı tarafından oluşturulur. IMS ' tan alınan bir iletinin MQMD ' si şu şekilde oluşturulmuştur:

```
StrucID
  "MD"
S\u00f0fcr\u00f0fcm
  MQMD_VERSION_1
```



**Rapor**

MQRO\_NONE

**MsgType**

MQMT\_REPLY

**Son kullanma tarihi**

MQIIH ' in İşaretler alanında MQIIH\_PASS\_EXPTH değeri ayarlandıysa, bu alan kalan süre bitimi değerini içerir, aksi takdirde MQE\_UNESNC olarak ayarlanır.

**Geribildirim**

MQFB\_YOK

**Kodlama**

MQENC.Native ( z/OS sisteminin kodlaması)

**CodedCharSetId**

MQCCSI\_Q\_MGR ( z/OS sisteminin CodedCharSetID değeri)

**Biçim**

MQMD.Format

**Öncelik**

MQMD.Priority

**Kalıcılık**

Depends on commit mode: MQMD.Persistence of the input message if CM-1; persistence matches recoverability of the IMS message if CM-0

**MsgId**

MQMD.MsgId , MQRO\_PASS\_MSG\_ID, tersi durumda Yeni MsgId (varsayılan)

**CorrelId**

MQMD.CorrelId from the input message if MQRO\_PASS\_CORREL\_ID, otherwise MQMD.MsgId from the input message (the default)

**BackoutCount**

0

**ReplyToQ**

Boşluklar

**ReplyToQMgr**

Boşluklar (MQPUT sırasında kuyruk yöneticisi tarafından yerel qmgr adı olarak ayarlanır)

**UserIdentifier**

MQMD.UserIdentifier

**AccountingToken**

MQMD.AccountingToken

**ApplIdentityVerileri**

MQMD.ApplIdentityData giriş iletisi

**PutApplTipi**

Hata yoksa MQAT\_XCF, tersi durumda MQAT\_BRIDGE

**PutApplAdı**

<XCFgroupName> <XCFmemberName> hata yoksa, QMGR adı

**PutDate**

İletinin konulduğu tarih

**PutTime**

İletinin konulduğu saat

**ApplOriginVerileri**

Boşluklar

*IMS köprüsünden gelen iletelerde MQIIH alanları*

Learn about the MQIIH fields in messages from the IMS bridge.

IMS ' tan alınan bir iletinin MQIIH değeri şu şekilde oluşturulmuştur:

**StrucId**

"IH"

**S\u00fcr\u00fcm**

1

**StrucLength**

84

**Kodlama**

MQENC\_NATIVE

**CodedCharSetId**

MQCCSI\_Q\_MGR

**Biçim**

MQIIH.ReplyToFormat boş değilse, giriş iletisininMQIIH.ReplyToFormat ' u, tersi durumda 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 konursa giriş iletisininMQIIH.Authenticator değeri, başka bir şekilde boşluk olur.

**TranInstanceTanıtıcısı**

Sohbetlerde, OTMA üstbilgisinden Sohbet Tanıtıcısı/Sunucu Belirteci. V14öncesindeki IMS sürümlerinde, bu alan her zaman, sohbette değilse boş değerlerde olur. IMS V14 ' ten itibaren bu alan, sohbet sırasında olmasa da IMS tarafından ayarlanabilirler.

**TranState**

"C" ise, iletişim halinde, aksi takdirde boşluk

**CommitMode**

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

**SecurityScope**

Boş

**Ayrıldı**

Boş

*IMS' dan gelen yanıt iletileri*

Bir IMS hareketi ISRTS 'yi IOPCB' ye gönderdiğinde, ileti kaynak LTERM ya da TPIPE ' ye geri yönlendirilir.

Bunlar, IBM MQ içinde yanıt iletileri olarak görülür. IMS ' dan gelen yanıt iletileri, özgün iletide belirtilen yanıtlama kuyruğuna konilir. İleti, yanıt kuyruğuna konamazsa, köprünün yetkisini kullanarak, ölü-mektup kuyruğuna konabilecektir. İleti, ölü-mektup kuyruğuna yerleştirilemiyorsa, iletinin alınmadığını belirten bir eksi alındı bildirimini IMS ' a gönderilir. İletinin sorumluluğu daha sonra IMS' a geri gönderilir.

Kesinleştirme kipi 0 kullanıyorsanız, bu Tpipe 'den gelen iletiler köprüye gönderilmez ve IMS kuyruğunda kalır; başka bir ileti yoksa, yeniden başlatma işlemi başlatılıncaya kadar başka ileti gönderilmez.

Kesinleştirme kipi 1 kullanıyorsanız, diğer işler devam edebilir.

Yanıta bir MQIIH yapısı varsa, biçimi MQFMT\_IMS; değilse, biçim tipi, ileti eklenirken kullanılan IMS Değişiklik Yönetimi adı ile belirtilir.

### *IMS hareketlerinde diğer yanıt PCB ' lerinin kullanılması*

Bir IMS hareketi diğer yanıt PCB 'lerini (ALTPCB' ye ISRTS ya da değiştirilebilir bir PCB ' ye bir CHNG çağrısı) kullandığında, iletinin yeniden yönlendirilmesi gerekip gerekmediğini belirlemek için ön yönlendirme çıkışı (DFSYPRX0) çağrılır.

İleti yeniden yönlendirilecekse, hedef çözüm çıkışı (DFSYDRU0) hedefi doğrulamak ve üstbilgi bilgilerini hazırlamak için çağrılır ve bu çıkış programlarıyla ilgili bilgi için [Using OTMA exits in IMS](#) ve [Ön yönlendirme çıkışı DFSYPRX0](#) başlıklı konuya bakın.

Çıkışlarda işlem yoksa, bir IBM MQ kuyruk yöneticisinden başlatılan IMS hareketlerinden, IOPCB 'den ya da ALTPCB' den başlatılan tüm çıkış, aynı kuyruk yöneticisine döndürülür.

### *İstenmeyen iletiler IMS' den gönderiliyor*

IMS 'dan bir IBM MQ kuyruğuna ileti göndermek için, bir ALTPCB' ye ISRTS ' den bir IMS işlemi başlatmanız gerekir.

You need to write pre-routing and destination resolution exits to route unsolicited messages from IMS and build the OTMA user data, so that the MQMD of the message can be built correctly. Bu çıkış programlarıyla ilgili bilgi için [Ön yönlendirme çıkışı DFSYPRX0](#) ve [Hedef çözme kullanıcı çıkışı](#) başlıklı konuya bakın.

**Not:** IBM MQ - IMS köprüsü, aldığı iletinin yanıt olup olmadığını ya da istenmemiş bir ileti olup olmadığını bilmiyor. İletiyi aynı şekilde, iletinin MQMD ve MQIIH 'lerini oluştururken, iletiyle birlikte gönderilen OTMA UserData ' ne dayalı olarak bu iletiyi işler.

İstenmeyen iletiler yeni TPipes yaratabilir. Örneğin, var olan bir IMS işlemi yeni bir LTERM ' ye (örneğin PRINT01gibi) geçtiyse, ancak uygulama çıkışın OTMA aracılığıyla sağlanmasını gerektiriyorsa, bu örnekte yeni bir Tpipe (bu örnekte PRINT01 adı verilir) yaratılır. Varsayılan olarak bu, uyumlaştırılmamış bir Tpipe 'dir. Uygulama iletinin kurtarılabilir olmasını gerektiriyorsa, hedef çözme çıkış çıkış işaretini ayarlayın. Ek bilgi için [IMS Customization Guide](#) belgesine bakın.

### *İleti bölümlenmesi*

IMS işlemlerini tek ya da çok bölümlü giriş olarak tanımlayabilirsiniz.

Kaynak IBM MQ uygulaması, bir ya da daha fazla LLZZ-veri bölümü olarak MQIIH yapısının ardından kullanıcı girişini oluşturmalıdır. Bir IMS iletisinin tüm kesimleri, tek bir MQPUT ile gönderilen tek bir IBM MQ iletisinde yer almalıdır.

LLZZ veri kesiminin uzunluk üst sınırı, IMS/OTMA (32767 bayt) tarafından tanımlanır. Toplam IBM MQ ileti uzunluğu, LL baytlarının toplamını artı MQIIH yapısının uzunluğunu içerir.

Yanıtın tüm bölümleri tek bir IBM MQ iletisinde yer alır.

MQFMT\_IMS\_VAR\_STRING biçimine sahip iletilerde 32 KB sınırlamasına ilişkin daha fazla kısıtlama var. Bir ASCII karma CCSID iletisinde bulunan veriler, EBCDIC karma CCSID iletisine dönüştürüldüğünde, SBCS ile DBCS karakterleri arasında geçiş her olduğunda bir çift bayt dizilimi başlangıç baytı ya da bir çift bayt dizilimi başlangıç baytı eklenir. 32 KB sınırlaması, iletinin büyüklük üst sınırı için geçerlidir. Bunun nedeni, iletteki LL alanı 32 KB 'yi geçemediğinden, iletinin tüm üst karakter ve başlangıç karakterleri de içinde olmak üzere 32 KB' yi aşmaması gerekir. İletiyi oluşturma işlemi bunun için izin vermelidir.

### *Data conversion for messages to and from the IMS bridge*

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

Gerekli tüm çıkışların, CSQXLIB DD bildiriminde gönderme yapılan veri kümesindeki dağıtılmış kuyruğa alma olanağı için kullanılabilir olması gerekir. Bu, herhangi bir IBM MQ platformundan IBM MQ - IMS köprüsünü kullanarak bir IMS uygulamasına ileti gönderebileceğiniz anlamına gelir.

Dönüştürme hataları varsa, ileti dönüştürülmeden kuyruğa alınır; bu sonuç, köprü'nün üstbilgi biçimini tanıyamadığı için IBM MQ - IMS köprüsü tarafından bir hata olarak ele alınır. Bir dönüştürme hatası ortaya çıkarsa, z/OS konsoluna bir hata iletisi gönderilir.

Genel olarak veri dönüştürme hakkında ayrıntılı bilgi için bkz. [“Veri dönüştürme çıkışları yazılıyor” sayfa 1014](#) .

## Sending messages to the IBM MQ - IMS bridge

Dönüştürmenin doğru bir şekilde gerçekleştirildiğinden emin olmak için, kuyruk yöneticisine iletinin biçiminin ne olduğunu söylemelisiniz.

İletin bir MQIIH yapısı varsa, MQMD ' deki *Format* değeri yerleşik MQFMT\_IMS biçimine ayarlanmalıdır ve MQIIH içindeki *Format* , ileti verilerinizi tanımlayan biçimdeki ada ayarlanmalıdır. MQIIH yoksa, MQMD 'de *Format* ' i biçim adınıza ayarlayın.

Verileriniz (LLZZ'lerden başka), tüm karakter verileri (MQCHAR) ise, biçim adı olarak (MQIIH ya da MQMD ' de uygun olarak), MQFMT\_IMS\_VAR\_STRING yerleşik biçimi olarak kullanın. Ters durumda, kendi biçim adınızı kullanın; bu durumda, biçiminiz için bir veri dönüştürme çıkışı da sağlamanız gerekir. Çıkışta, verilerin kendisinin yanı sıra, iletinizdeki LLZZs 'lerin dönüştürülmesini de işlemesi gerekir (ancak iletinin başlangıcındaki herhangi bir MQIH' yi işlemek zorunda değildir).

Uygulamanız *MFSMapName* kullanıyorsa, bunun yerine MQFMT\_IMS ile iletileri kullanabilir ve MQIIH ' nin *MFSMapName* alanında IMS işlemine geçirilen eşlem adını tanımlayabilirsiniz.

## Receiving messages from the IBM MQ - IMS bridge

If an MQIIH structure is present on the original message that you are sending to IMS, one is also present on the reply message.

Yanıtınızın doğru şekilde dönüştürülmesini sağlamak için:

- Özgün iletinizde bir MQIIH yapınız 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 > biçiminde olduğunda yararlı olur.
- Özgün iletinizde MQIIH yapısına sahip değilseniz, yanıt iletisi için IMS uygulamasının ISRT ' deki MFS MOD adı olarak IOPCB olarak yanıt iletisi olmasını istediğiniz biçimi belirtin.

## JMS ve Java uygulamalarının geliştirilmesi

IBM MQ , iki Java dil arabirimi sağlar: IBM MQ classes for Java Message Service ve IBM MQ classes for Java.

### Bu görev hakkında

IBM MQ içinde, Java uygulamalarında kullanılmak üzere iki alternatif API vardır. Bir Java uygulaması, IBM MQ kaynaklarına erişmek için IBM MQ classes for JMS ya da IBM MQ classes for Java ' yi kullanabilir.

#### IBM MQ classes for JMS

IBM MQ classes for Java Message Service (JMS), IBM MQ ile birlikte verilen JMS sağlayıcısıdır. Java Platform, Enterprise Edition Connector Architecture (JCA), Java EE ortamında çalışan uygulamaların IBM MQ ya da Db2 gibi bir Enterprise Information System (EIS) ortamına bağlanması için standart bir yol sağlar.

If you are not familiar with IBM MQ, or already have JMS experience, you might find it easier to use the familiar JMS API to access IBM MQ resources, by using IBM MQ classes for JMS. JMS aynı zamanda Java Platform, Enterprise Edition ( Java EE) platformunun ayrılmaz bir parçasıdır. Java EE uygulamaları iletileri zamanuysuz olarak işlemek için iletileyle yönlendirilen Bean 'leri (MDBs) kullanabilir. JMS is also the standard mechanism for Java EE to interact with asynchronous messaging systems such as IBM MQ. Java EE ile uyumlu olan her uygulama sunucusu bir JMS sağlayıcısı içermelidir; bu nedenle, farklı uygulama sunucuları arasında iletişim kurmak için JMS ' u kullanabilirsiniz ya da bir uygulamayı, uygulamada herhangi bir değişiklik yapmadan bir JMS sağlayıcısından başka bir uygulamaya geçirebilirsiniz.

## IBM MQ classes for Java

IBM MQ classes for Java enable you to use IBM MQ in a Java environment. IBM MQ classes for Java allow a Java application to connect to IBM MQ as an IBM MQ client, or connect directly to an IBM MQ queue manager.

IBM MQ classes for Java encapsulates the Message Queue Interface (MQI), the native IBM MQ API, and uses the same object model as other object-oriented interfaces, whereas IBM MQ classes for Java Message Service implements Oracle's Java Message Service (JMS) interfaces.

If you are familiar with IBM MQ in environments other than Java, using either procedural or object-oriented languages, you can transfer your existing knowledge to the Java environment by using IBM MQ classes for Java. You can also exploit the full range of features of IBM MQ, not all of which are available in IBM MQ classes for JMS.

### Not:

IBM , IBM MQ classes for Java 'de başka bir geliştirme yapmayacaktır ve bunlar, IBM MQ 8.0' ta teslim edilen düzeyde işlevsel olarak sabitlenirler. IBM MQ classes for Java ' u kullanan var olan uygulamalar tam olarak desteklenmeye devam eder, ancak yeni özellikler eklenmez ve geliştirmeler için istek reddedilir. Tam olarak desteklenen bir şekilde, hataların IBM MQ Sistem Gereksinimleri üzerindeki değişikliklerle birlikte belirlendiği değişiklikler ile birlikte düzeltileceği anlamına gelir.

IBM MQ classes for Java , IMSiçinde desteklenmiyor.

IBM MQ classes for Java , WebSphere Libertyiçinde desteklenmiyor. 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](#).

## kullanmaIBM MQ classes for JMS

IBM MQ classes for Java Message Service (IBM MQ classes for JMS), IBM MQile birlikte verilen JMS sağlayıcısıdır. javax.jms paketinde tanımlanan arabirimlerin yanı sıra, IBM MQ classes for JMS , JMS API ' ye iki uzantı kümesi sağlar.

JMS belirtimi, uygulamaların ileti alışverişi işlemlerini gerçekleştirmek için kullanabilecekleri bir arabirim kümesini tanımlar. Belirtimin en son sürümü JMS 2.0' dir. javax.jms paketi, JMS arabirimlerini tanımlar ve bir JMS sağlayıcısı belirli bir ileti alışverişi ürünü için bu arabirimleri uygular. IBM MQ classes for JMS is a JMS provider that implements the JMS interfaces for IBM MQ.

JMS belirtimi, ConnectionFactory ve Hedef nesnelerin denetlenmesini bekler. Bir yönetici, merkezi bir havuzda denetlenen nesnelere yaratır ve bakımını yapar ve JMS uygulaması bu nesnelere Java Naming Directory Interface (JNDI) kullanarak alır. IBM MQ classes for JMS , yönetilen nesnelerin kullanımını destekler ve bir yönetici, denetlenen nesnelere oluşturmak ve bakımını yapmak için IBM MQ JMS yönetim aracını ya da IBM MQ Explorer ' ı kullanabilir.

IBM MQ classes for JMS , JMS API ' ye iki uzantı kümesi de sağlar. Bu uzantıların ana odağı, yürütme sırasında bağlantı üreticilerinin ve hedeflerin dinamik olarak oluşturulması ve yapılandırılması ile ilgilidir; ancak, uzantılar sorunun saptanması için işlev gibi ileti sistemiyle doğrudan ilgili olmayan bir işlev de sağlar.

### IBM MQ JMS uzantıları

IBM MQ classes for JMS ' ın önceki yayın düzeyleri, MQConnectionFactory, MQQueue ve MQTopic nesnelere gibi nesnelere uygulanan uzantıları içerir. Bu nesnelere, IBM MQ' e özgü özellikleri ve yöntemleri içerir. Nesnelere yönetilebilir ya da bir uygulama, çalıştırma zamanında nesnelere dinamik olarak yaratabilir. This release of IBM MQ classes for JMS maintains these extensions, which are now known as the IBM MQ JMS extensions. Bu uzantıları kullanan uygulamalar, değişiklik yapılmaksızın, kullanmaya devam edebilirsiniz.

### IBM JMS uzantıları

Bu IBM MQ classes for JMS yayın düzeyi, ileti sistemi sistemi olarak IBM MQ 'a özgü olmayan, JMS API' ye daha soysal bir uzantı kümesi sağlar. Bu uzantılar, IBM JMS uzantıları olarak bilinir ve aşağıdaki geniş hedeflere sahiptir:

- IBM JMS sağlayıcıları arasında daha yüksek bir tutarlılık düzeyi sağlamak için

- İki IBM ileti sistemi arasında bir köprü uygulaması yazmanın daha kolay olmasını sağlamak
- Bir uygulamayı bir IBM JMS sağlayıcısından başka biranaothersağlayıcıya bir uygulamayı daha kolay bir şekilde kaplamak için

Uzantılar, IBM Message Service Client for C/C++ ve IBM Message Service Client for .NET' da sağlanan işlevle benzer bir işlev sağlar.

IBM MQ 8.0' dan IBM MQ classes for JMS , Java 7 ile oluşturulmuştur.

Java 7 yürütme ortamı, daha önceki sınıf dosyası sürümlerinin çalıştırılmasını destekler.

### **İlgili kavramlar**

IBM MQ Java dil arabirimleri

“JMS modeli” sayfa 125

The JMS model defines a set of interfaces that Java applications can use to perform messaging operations. IBM MQ classes for JMS, JMS sağlayıcısı olarak, JMS nesnelere IBM MQ kavramlarına nasıl ilişkin olduğunu tanımlar. JMS belirtimi, belirli JMS nesnelere yönetilecek nesnelere olmasını bekler. JMS 2.0 , JMS 1.1'tan klasik API' yi de korurken, basitleştirilmiş bir API ' yi sunar.

“JMS 2.0 işlevselliğini kullanma” sayfa 299

JMS 2.0 , IBM MQ classes for JMS' a birçok yeni işlev alanı sunar.

### **Neden IBM MQ classes for JMS kullanmalıyım?**

Using IBM MQ classes for JMS has a number of advantages including being able to reuse any existing JMS skills in your organization, and applications being more independent from the JMS provider and the underlying IBM MQ configuration.

IBM MQ classes for JMS , Java uygulamalarının IBM MQ kaynaklarına erişmek için kullanabileceği iki alternatif API ' den biridir. Diğer API: IBM MQ classes for Java. IBM MQ classes for Java ' ı kullanan var olan uygulamalar tam olarak desteklenmeye devam etse de, yeni uygulamalar IBM MQ classes for JMS (bkz. “API ' nin seçimi” sayfa 79) olanağını kullanmalıdır.

### **Summary of advantages of using IBM MQ classes for JMS**

IBM MQ classes for JMS ' un kullanılması, var olan JMS becerilerini yeniden kullanmanıza ve uygulama bağımsızlığı sağlamanıza olanak sağlar.

- JMS becerilerini yeniden kullanabilirsiniz.

IBM MQ classes for JMS , ileti sistemi sistemi olarak IBM MQ için JMS arabirimlerini uygulayan bir JMS sağlayıcısıdır. If your organization is new to IBM MQ, but already has JMS application development skills, you might find it easier to use the familiar JMS API to access IBM MQ resources rather than one of the other APIs provided with IBM MQ.

- JMS , Java Platform, Enterprise Edition ' un ayrılmaz bir parçasıdır (Java EE).

JMS , Java EE platformunda ileti alışverişi için kullanılacak doğal API ' dir. Java EE ile uyumlu olan 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). Note in particular that Java EE applications use MDBs to process messages asynchronously, and all messages are delivered to MDBs as JMS messages.

- Bağlantı üreticileri ve hedefleri, bir uygulamaya sabit olarak kodlanmaktan ziyade, merkezi bir havuzda JMS tarafından yönetilen nesnelere olarak depolanabilir.

Bir yönetici, merkezi bir havuzda JMS denetimli nesnelere oluşturabilir ve bakımını yapabilir ve IBM MQ classes for JMS uygulamaları Java Naming Directory Interface (JNDI) olanağını kullanarak bu nesnelere alabilir. JMS bağlantı fabrikaları ve hedefleri, kuyruk yöneticisi adları, kanal adları, bağlantı seçenekleri, kuyruk adları ve konu adları gibi IBM MQ' e özgü bilgileri içerir. Bağlantı fabrikaları ve hedefler, yönetilen nesnelere olarak depolandıysa, bu bilgiler bir uygulamaya değişmez olarak kodlanmaz. Bu nedenle bu düzenleme, temeldeki IBM MQ yapılandırmasından bir derece bağımsızlığa sahip bir uygulama sağlar.

- JMS , uygulama taşınabilirliği sağlayabilen bir sektör standardı API ' sidir.

Bir JMS uygulaması, denetlenen nesnelere olarak saklanan bağlantı fabrikalarını ve hedeflerini almak için JNDI kullanılabilir ve ileti alışverişi işlemlerini gerçekleştirmek için yalnızca javax.jms paketinde tanımlı arabirimleri kullanır. Uygulama daha sonra IBM MQ classes for JMS gibi herhangi bir JMS sağlayıcısından tamamen bağımsızdır ve uygulamada herhangi bir değişiklik yapılmaksızın bir JMS sağlayıcısından başka bir providersağlayıcısından dışarı aktarılabilir. JNDI belirli bir uygulama ortamında kullanılabilir değilse, bir IBM MQ classes for JMS uygulaması, yürütme sırasında dinamik olarak bağlantı üreticileri ve hedefleri oluşturmak ve yapılandırmak için JMS API uzantılarını kullanabilir. The application is then completely self-contained, but is tied to IBM MQ classes for JMS as the JMS provider.

- Bridge applications might be easier to write by using JMS.

Bir 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 kullanarak karmaşık olabilir. Bunun yerine, her ileti sistemi için bir tane olmak üzere iki JMS sağlayıcıyı kullanarak bir köprü uygulaması yazabilirsiniz. Uygulama daha sonra yalnızca bir API, JMS API ' yı kullanır ve yalnızca JMS iletilerini işler.

## Yerleştirilebilir ortamlar

Bir Java EE uygulama sunucusu ile bütünleştirme sağlamak için Java EE standartları, ileti alışverişi sağlayıcılarının bir kaynak bağdaştırıcısı sağlamalarını gerektirir. Java EE Connector Architecture (JCA) belirtiminin ardından IBM MQ , herhangi bir sertifikalı Java EE ortamı içinde ileti sistemi işlevlerini sağlamak için JMS ' i kullanan bir kaynak bağdaştırıcısı sağlar.

While it has been possible to use the IBM MQ classes for Java inside Java EE, this API is not engineered or optimized for this purpose. See the IBM technote [J2EE/JEE Ortamlarındaki WebSphere MQ Java Arabirimlerini Kullanma](#) for details of IBM MQ classes for Java considerations within Java EE.

Java EE ortamının dışında OSGi ve JAR dosyaları sağlar ve bu dosyalar yalnızca IBM MQ classes for JMS' u edinmenizi sağlar. Bu JAR dosyaları artık bağımsız olarak ya da Maven gibi yazılım yönetimi çerçevelerinin içinde kullanıma hazır olarak daha kolay devreye alınabilir. Daha fazla bilgi için bkz. IBM teknik notu [JMS için WebSphere MQ sınıflarının alınması](#).

## API ' nin seçimi

Yeni uygulamalar, IBM MQ classes for JMS yerine IBM MQ classes for JMS ' yi kullanmalıdır.

IBM MQ classes for JMS , IBM MQ' un noktadan noktaya iletişim ve yayınlama/abone olma ileti sistemi özelliklerine erişim sağlar. Ayrıca, JMS standart ileti sistemi modeli için destek sağlayan JMS iletileri gönderirken, uygulamalar ek üstbilgiler olmadan da ileti gönderebilir ve alabilir; örneğin, C MQI uygulamaları gibi diğer IBM MQ uygulamalarıyla da çalışabilir. MQMD ve MQ ileti yüklerinin tam denetimi kullanılabilir. İleti akışı, zaman uyumsuz koyma ve rapor iletileri gibi ek IBM MQ özellikleri de kullanılabilir. Sağlanan PCF yardımcı sınıflarını kullanarak, IBM MQ PCF denetim iletileri JMS API aracılığıyla gönderilebilir ve alınabilir ve kuyruk yöneticilerini denetlemek için kullanılabilir.

Features that have recently been added to IBM MQ, such as asynchronous consume and automatic reconnection, are not available in the IBM MQ classes for Java, but are available in the IBM MQ classes for JMS. IBM MQ classes for Java ' i kullanan var olan uygulamalar tam olarak desteklenmeye devam eder.

IBM MQ classes for JMS aracılığıyla olmayan IBM MQ özelliklerine erişmeniz gerekirse, Enhancement (RFE) için bir Request (Request for Enhancement; Geliştirme İsteği) artırabilirsiniz. IBM daha sonra uygulamanın IBM MQ classes for JMS uygulamasında mümkün olup olmadığını ya da izlenebilecek en iyi uygulama olup olmadığını bildiren bir bilgi verebilir. Ek ileti sistemi özellikleri için, IBM açık standarda katkıda bulunan olarak, bu özellikler JCP işleminin bir parçası olarak yükseltilebilir.

## İlgili görevler

[IBM MQ classes for JMS uygulamalarının izlenmesi](#)

[Java ve JMS sorun giderme](#)

## İlgili bilgiler

[IBM RFE Alt Görev Süreci](#)

[JMS Java Belirtimi Gözden Geçirme Süreci](#)



## IBM MQ classes for JMS için önkoşullar

Bu konu, IBM MQ classes for JMS komutunu kullanmadan önce bilmeniz gereken bilgileri içerir. IBM MQ classes for JMS uygulamalarını geliştirmek ve çalıştırmak için, bazı yazılım bileşenlerine önkoşul olarak gereksinim duyarsınız.

IBM MQ classes for JMS ile ilgili önkoşullarla ilgili bilgi için bkz. [IBM MQ](#).

IBM MQ classes for JMS uygulamalarını geliştirmek için bir Java SE Yazılım Geliştirme Takımı 'na (SDK) gereksinim duyarsınız. İşletim sisteminiz tarafından desteklenen JDK'lerle ilgili ayrıntılar için bkz. [IBM MQ](#).

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 için, Qshell, işletim sisteminin 30. seçeneği.
-  z/OS, UNIX and Linux System Services (USS) için.

IBM JSSE sağlayıcısı, FIPS sertifikalı bir şifreleme sağlayıcısı içerir; bu nedenle, FIPS 140-2 uyumluluğu için hemen kullanıma hazır olarak programlı olarak yapılandırılabilir. Bu nedenle, FIPS 140-2 uyumluluğu doğrudan doğruya IBM MQ classes for Java ve IBM MQ classes for JMS' den desteklenebilir.

Oracle' ın JSSE sağlayıcısıyla ilgili olarak yapılandırılmış FIPS sertifikalı bir şifreleme sağlayıcısı olabilir, ancak bu, anında kullanım için 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 uyumluluğu el ile etkinleştirebilir, ancak IBM şu anda bu tür uyumluluğu sağlayamıyor olabilir.

IPv6 adresleri Java sanal makinesiz (JVM) ve işletim sisteminizdeki TCP/IP somutlaması tarafından destekleniyorsa, IBM MQ classes for JMS uygulamalarınızda Internet Protocol Sürüm 6 (IPv6) adreslerini kullanabilirsiniz. IBM MQ JMS yönetim aracı (bkz. [Administration Tool kullanarak JMS nesnelerini yapılandırma](#) ) IPv6 adreslerini de kabul eder.

The IBM MQ JMS administration tool and IBM MQ Explorer use the Java Naming Directory Interface (JNDI) to access a directory service, which stores administered objects. IBM MQ classes for JMS uygulamaları, yönetilen nesnelere bir dizin hizmetinden almak için JNDI ' i de kullanabilir. Hizmet sağlayıcı, dizin hizmetine JNDI çağrılarını eşleyerek bir dizin hizmetine erişim sağlayan bir koddur. A file system service provider in the files `fscontext.jar` and `providerutil.jar` is supplied with IBM MQ classes for JMS. Dosya sistemi hizmet sağlayıcısı, yerel dosya sistemine dayalı olarak bir dizin hizmetine erişim sağlar.

LDAP sunucusuna dayalı bir dizin hizmeti kullanmak istiyorsanız, bir LDAP sunucusu kurmalı ve yapılandırmalı ya da var olan bir LDAP sunucusuna erişiminiz olmalıdır. In particular, you must configure the LDAP server to store Java objects. LDAP sunucunuzu kurmaya ve yapılandırmaya ilişkin bilgi için sunucunuzla birlikte sağlanan belgelere bakın.

## Kurma ve Yapılandırma IBM MQ classes for JMS

Bu bölümde, IBM MQ classes for JMS 'u kurduğunuzda oluşturulan dizinler ve dosyalar açıklanır ve kuruluşun sonradan IBM MQ classes for JMS ' in nasıl yapılandırılacağı anlatılır.

### İlgili kavramlar

[“IBM MQ kaynak bağdaştırıcısının kullanılması” sayfa 415](#)

Kaynak bağdaştırıcısı, bir uygulama sunucusunda çalışan uygulamaların IBM MQ kaynaklarına erişmelerini sağlar. Gelen ve giden iletişimi destekler.



## IBM MQ classes for JMS için kurulu olan

IBM MQ classes for JMS' u kurduğunuzda, bir dizi dosya ve dizin oluşturulur. Windows' ta, bazı yapılandırma otomatik olarak ortam değişkenleri ayarlanarak kuruluş sırasında gerçekleştirilir. On other platforms, and in certain Windows environments, you must set environment variables before you can run IBM MQ classes for JMS applications.

Çoğu işletim sistemi için, IBM MQ ürününü kurduğunuzda IBM MQ classes for JMS isteğe bağlı bir bileşen olarak kurulur.

IBM MQ kuruluşuna ilişkin ek bilgi için aşağıdaki başlara bakın:

► **Multi** IBM MQ ürününü kurma

► **z/OS** IBM MQ for z/OS ürününü kurma

**Önemli:** “IBM MQ classes for JMS yeniden yerelleşme JAR dosyaları” sayfa 82’inde açıklanan yeniden yerelleşme JAR dosyaları dışında, IBM MQ classes for JMS JAR dosyalarının ya da yerel kitaplıkların diğer makinelere ya da IBM MQ classes for JMS ' in kurulu olduğu bir makinede farklı bir konuma kopyalanması desteklenmez.

## Kuruluş dizinleri

Çizelge 5 sayfa 81 , her altyapıda IBM MQ classes for JMS kütüklerinin kurulu olduğunu gösterir.

Çizelge 5. IBM MQ classes for JMS kuruluş dizinleri	
Platform	Dizin
► <b>Linux</b> ► <b>UNIX</b> UNIX and Linux	MQ_INSTALLATION_PATH/java
► <b>Windows</b> Windows	MQ_INSTALLATION_PATH\java
► <b>IBM i</b> IBM i	/QIBM/ProdData/mqm/java
► <b>z/OS</b> z/OS	MQ_INSTALLATION_PATH/mqm/V9R1M0/java
	MQ_INSTALLATION_PATH/opt/mqm/java

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Kuruluş dizini aşağıdaki içeriği içerir:

- The IBM MQ classes for JMS JAR files, including the relocatable JAR files, which are in the MQ\_INSTALLATION\_PATH\java\lib directory.
- The IBM MQ native libraries, which are used by applications that use the Java Native Interface.

32 bit yerel kitaplıklar, MQ\_INSTALLATION\_PATH\java\lib dizinine kurulur ve 64 bit yerel kitaplıklar MQ\_INSTALLATION\_PATH\java\lib64 dizininde bulunur.

IBM MQ yerel kitaplıklarıyla ilgili daha fazla bilgi için bkz. “Java Native Interface (JNI) kitaplıklarının yapılandırılması” sayfa 86.

- “IBM MQ classes for JMS ile sağlanan komut dosyaları” sayfa 111’inde anlatılan ek komut dosyaları. Bu komut dosyaları, MQ\_INSTALLATION\_PATH\java\bin dizininde bulunur.
- IBM MQ classes for JMS API ' nin belirtileri. The Javadoc tool has been used to generate the HTML pages that contain the specifications of the API.

HTML sayfaları, MQ\_INSTALLATION\_PATH\java\doc\WMQJMSClasses dizininde bulunur:

- ► **ULW** UNIX, Linux, and Windows' ta, bu alt dizin tek tek HTML sayfalarını içerir.
- ► **IBM i** IBM i' ta, HTML sayfaları wmqjms\_javadoc . jar adlı bir dosyada yer alıyor.

- **z/OS** z/OS' ta, HTML sayfaları wmqjms\_javadoc . jaradlı bir dosyada yer alıyor.
- OSGi desteđi. OSGi paketleri java\lib\OSGi dizinine kurulur ve “OSGi desteđi” sayfa 112içinde açıklanmıştır.
- Herhangi bir Java Platform, Enterprise Edition 7 ( Java EE 7) uyumlu uygulama sunucusunda devreye alınabilen IBM MQ kaynak bađdařtırıcısı.

IBM MQ kaynak bađdařtırıcısı MQ\_INSTALLATION\_PATH\java\lib\jca dizininde yer alıyor; daha fazla bilgi için bkz. “IBM MQ kaynak bađdařtırıcısının kullanılması” sayfa 415

- **Windows** Windowsüzerinde, hata ayıklama için kullanılabilir simgeler, MQ\_INSTALLATION\_PATH\java\lib\symbols dizinine kurulur.

Kuruluř dizininde, diđer IBM MQ bileřenlerine ait bazı dosyalar da bulunur.

## Örnek Uygulamalar

Bazı örnek uygulamalar IBM MQ classes for JMSile birlikte sađlanır. Çizelge 6 sayfa 82 , her altyapıda örnek uygulamaların kurulu olduđu yeri gösterir.

Platform		Dizin
<b>AIX</b> UNIX and Linux		MQ_INSTALLATION_PATH/samp/jms
<b>Windows</b> Windows		MQ_INSTALLATION_PATH\tools\jms
<b>IBM i</b> IBM i		/QIBM/ProdData/mqm/java/samples/jms
<b>z/OS</b> z/OS		MQ_INSTALLATION_PATH/mqm/V9R1M0/java/samples/jms
		MQ_INSTALLATION_PATH/opt/mqm/samp/jms

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduđu üst düzey dizini temsil eder.

Kuruluřtan sonra, uygulamaları derlemek ve çalıřtırmak için bazı yapılanıř görevlerini gerçekleřtirmeniz gerekebilir.

“IBM MQ classes for JMS için ortam deđiřkenlerini ayarlama” sayfa 84 , basit IBM MQ classes for JMS uygulamalarını çalıřtırmak için gerekli olan sınıf yolunu açıklar. Bu konuda, özel durumlarda başvurulması gereken ek JAR dosyaları ve IBM MQ classes for JMSile birlikte verilen komut dosyalarını çalıřtırmak için ayarlamamız gereken ortam deđiřkenleri de açıklanır.

Bir uygulamanın izlenmesi ve günlüđe kaydedilmesi gibi özellikleri denetlemek için bir yapılandırma özellikleri dosyası sađlamamız gerekir. IBM MQ classes for JMS yapılandırma özellikleri dosyası “IBM MQ classes for JMS yapılandırma dosyası” sayfa 88içinde açıklanmıştır.

### İlgili kavramlar

Kaynak bađdařtırıcısı konuřlandırılırken sorunlar oluřtu

### İlgili görevler

“IBM MQ classes for JMS örnek uygulamalarının kullanılması” sayfa 108

IBM MQ classes for JMS örnek uygulamaları, JMS API ' nin ortak özelliklerine genel bir bakıř sađlar. Bunları, kuruluřunuzu ve ileti sistemi sunucunuzu ayarlamak ve kendi uygulamalarınızı oluřturmanıza yardımcı olmak için kullanabilirsiniz.

IBM MQ classes for JMS yeniden yerelleřme JAR dosyaları

Yeniden yerelleřme JAR dosyaları, IBM MQ classes for JMS' u çalıřtırmak için gereken sistemlere taşınabilir.

### Önemli:

- Yeniden yereldeki JAR dosyaları içinde açıklanan yeniden yerelleşme JAR dosyaları dışında, IBM MQ classes for JMS JAR dosyalarının ya da yerel kitaplıkların diğer makinelere ya da IBM MQ classes for JMS ' in kurulu olduğu bir makineden farklı bir konuma kopyalanması desteklenmez.
- Do not include the relocatable JAR files within applications deployed into Java EE application servers, such as WebSphere Application Server or WebSphere Liberty. Bu ortamlarda, IBM MQ kaynak bağdaştırıcısı yerine konuşlandırılmalı ve kullanılmalı. Note that WebSphere Application Server embeds the IBM MQ resource adapter, so there is no need to deploy it manually into this environment.
- Sınıf yükleyici çakışmalarını önlemek için, yeniden yerelleşme JAR dosyalarının aynı Java yürütme ortamı içindeki birden çok uygulama içinde paketlemek için önerilmez. Bu senaryoda, IBM MQ yeniden yerelleştirilebilir JAR dosyalarının Java yürütme ortamının sınıf yolunda (classpath) kullanılabilmesini sağlar.
- Uygulamalarınız içindeki yeniden yerelleme JAR dosyalarını paketliyorsanız, tüm önkoşul JAR dosyalarını Yeniden yerelleme JAR dosyalarıyla başlıklı konuda açıklandığı gibi içerdiğinizden emin olun. You should also ensure that you have appropriate procedures to update the bundled JAR files as part of application maintenance, to ensure that the IBM MQ classes for JMS remain current and known issues are re-mediated.

## Yereldeki JAR dosyaları yeniden yerlesin

Bir teşebbüs içinde, aşağıdaki dosyalar IBM MQ classes for JMS çalıştırmak zorunda olan sistemlere taşınabilir:

- `com.ibm.mq.allclient.jar`
- `com.ibm.mq.traceControl.jar`
- `jms.jar`
- `fscontext.jar`
- `providerutil.jar`
- `bcpkix-jdk15on.jar`
- `bcprov-jdk15on.jar`
- `V 9.1.0.9` `bcutil-jdk15on.jar`
- `V 9.1.2` `org.json.jar`

`jms.jar` dosyası, `javax.jms.jar` için bir arabirim dosyasıdır ve IBM dışı JMS sınıflarını içerir.

Uygulamanız bir dosya sistemi bağlamı kullanarak JNDI aramaları gerçekleştiriyorsa, `fscontext.jar` ve `providerutil.jar` dosyaları gereklidir.

Bouncy Castle güvenlik sağlayıcısı ve CMS desteği JAR dosyaları gereklidir. Daha fazla bilgi için bkz. [AMS ile IBM dışı JRE ' ler için destek](#). Aşağıdaki JAR dosyaları gereklidir:

- `bcpkix-jdk15on.jar`
- `bcprov-jdk15on.jar`
- `V 9.1.0.9` `bcutil-jdk15on.jar`

`V 9.1.2` IBM MQ classes for JMS uygulamanız JSON biçiminde bir CCDT kullanıyorsa, `org.json.jar` dosyası gereklidir.

`com.ibm.mq.allclient.jar` dosyası, IBM MQ classes for JMS, IBM MQ classes for Javave PCF ve Üstbilgiler Sınıflarını içerir. Bu dosyayı yeni bir konuma taşırsanız, bu yeni konumu yeni IBM MQ Fix Packs ile alıkoymak için gerekli adımları gerçekleştirdiğinizden emin olun. Ayrıca, geçici bir düzeltme alıyorsanız, bu dosyanın kullanılmasının IBM Desteği tarafından bilinen bir şekilde sağlandığından emin olun.

`com.ibm.mq.allclient.jar` dosyasının sürümünü belirlemek için şu komutu kullanın:

```
java -jar com.ibm.mq.allclient.jar
```

Aşağıdaki örnek, bu komuttan bazı örnek çıktıları göstermektedir:

```
C:\Program Files\IBM\MQ_1\java\lib>java -jar com.ibm.mq.allclient.jar
Name:      Java Message Service Client
Version:   9.1.0.0
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:   9.1.0.0
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:   9.1.0.0
Level:    p000-L140428.1 mqjbd=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:   9.1.0.0
Level:    p000-L140428.1
Build Type: Production
Location:  file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```

`com.ibm.mq.traceControl.jar` dosyası, IBM MQ classes for JMS uygulamalarına ilişkin izlemeyi devingen olarak denetlemek için kullanılır. Daha fazla bilgi için bkz. [Java için IBM MQ sınıflarını ve JMS için IBM MQ sınıfları kullanılarak çalışan bir işlemdeki izlemeyi denetleme.](#)





#### *IBM MQ classes for JMS için ortam değişkenlerini ayarlama*

IBM MQ classes for JMS uygulamalarını derleyebilmeniz ve çalıştırabilmeniz için, CLASSPATH ortam değişkeninize ilişkin ayarın IBM MQ classes for JMS Java arşiv (JAR) dosyasını içermesi gerekir. Gereksinimlerinize bağlı olarak, diğer JAR dosyalarını sınıf yolunuza eklemeniz gerekebilir. IBM MQ classes for JMS ile sağlanan komut dosyalarını çalıştırmak için diğer ortam değişkenleri ayarlanmalıdır.





## Bu görev hakkında

**Önemli:** Java `-Xbootclasspath` seçeneğinin IBM MQ classes for JMS ögesini içerecek şekilde ayarlanması desteklenmez.

IBM MQ classes for JMS uygulamalarını derlemek ve çalıştırmak için, [Çizelge 7 sayfa 84](#) içinde gösterildiği gibi altyapınıza ilişkin CLASSPATH ayarını kullanın. Bu ayar, IBM MQ classes for JMS örnek uygulamalarını derleyebilmeniz ve çalıştırabilmeniz için örnekler dizinini içerir. Alternatif olarak, ortam değişkenini kullanmak yerine **java** komutunda sınıf yolunu belirtebilirsiniz.

Hizmet olarak sunulan	CLASSPATH ayarı
 AIX	CLASSPATH= MQ_INSTALLATION_PATH/java/lib/com.ibm.mqjms.jar: MQ_INSTALLATION_PATH/samp/jms/samples:
 Linux  Solaris	CLASSPATH= MQ_INSTALLATION_PATH/java/lib/com.ibm.mqjms.jar: MQ_INSTALLATION_PATH/samp/jms/samples:
 IBM i	CLASSPATH=/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar: /QIBM/ProdData/mqm/java/samples/jms/samples:

Çizelge 7. Örnek uygulamalar da içinde olmak üzere IBM MQ classes for JMS uygulamalarını derlemek ve çalıştırmak için CLASSPATH ayarı (devamı var)

Hizmet olarak sunulan	CLASSPATH ayarı
 Windows Windows	CLASSPATH= MQ_INSTALLATION_PATH\java\lib\com.ibm.mqjms.jar; MQ_INSTALLATION_PATH\tools\jms\samples;
 z/OS	 CLASSPATH= MQ_INSTALLATION_PATH/mqm/V9R0M0/java/lib/ com.ibm.mqjms.jar: MQ_INSTALLATION_PATH/mqm/V9R0M0/java/samples/jms/samples:   CLASSPATH= MQ_INSTALLATION_PATH/mqm/V9R1M5/java/lib/ com.ibm.mqjms.jar: MQ_INSTALLATION_PATH/mqm/V9R1M5/java/samples/jms/samples:

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

com.ibm.mqjms.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 nesnelere 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 nesnelere 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 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, Çizelge 8 sayfa 87'içinde gösterildiği gibi IBM MQ classes for JMS kitaplıklarının saklandığı dizini belirtir.

## Yordam

### Windows

Windows işletim sistemlerinde, IBM MQ kurulduktan sonra **setmqenv** komutunu çalıştırın.

Önce bu komutu çalıştırmazsanız, bir **dspmqver** komutu verdiğinizde aşağıdaki hata iletisi görüntülenebilir:

```
> V9.1.0 AMQ8351: IBM MQ Java ortamı yapılandırılmadı  
doğru ya da IBM MQ JRE özelliği kurulmadı.
```

**Not:** **V9.1.0** IBM MQ Java Runtime Environment (JRE) programını kurmadıysanız bu ileti beklenir ([Ek Windows özellikleri önkoşul denetimikonusuna](#) bakın).

- Başka bir platformda, ortam değişkenlerini kendiniz ayarlayın:
  - **Linux** **UNIX** UNIX, ya da Linux sistemlerinde 32 bit JVM kullanıyorsanız ortam değişkenlerini ayarlamak için setjmsenv komut dosyasını kullanabilirsiniz.
  - **Linux** **UNIX** Bir UNIX ya da Linux sisteminde 64 bit JVM kullanıyorsanız ortam değişkenlerini ayarlamak için setjmsenv64 komut dosyasını kullanabilirsiniz. Bu komut dosyası `MQ_INSTALLATION_PATH/java/bin` dizininde bulunur; burada `MQ_INSTALLATION_PATH`, IBM MQ 'in kurulu olduğu üst düzey dizini gösterir.

setjmsenv ya da setjmsenv64 komut dosyasını çeşitli şekillerde kullanabilirsiniz: Bu komut dosyasını, tabloda gösterildiği gibi gerekli ortam değişkenlerini ayarlamak için temel olarak kullanabilir ya da bir metin düzenleyicisini 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 başlatmış olduğunuz her kabuk penceresinde, JMS doğrulama işlemi sırasında `./setjmsenv` ya da `./setjmsenv64` yazarak çalıştırmanız gerekir.

**IBM i** IBM üzerinde, `QIBM_MULTI_THREADED` ortam değişkenini Yolarak ayarlamanız gerekir. Birden çok iş parçacıklı uygulamaları, tek iş parçacıklı uygulamaları çalıştırdığınız şekilde çalıştırabilirsiniz. Ek bilgi için [IBM MQ 'nün Java ve JMS ile ayarlanması](#) başlıklı konuya bakın.

#### *Java Native Interface (JNI) kitaplıklarının yapılandırılması*

Bağ tanımları iletimi kullanılarak bir kuyruk yöneticisine bağlanan ya da istemci aktarımı kullanılarak kuyruk yöneticisine bağlanan ve Javadişındaki dillerde yazılmış kanal çıkış programlarını kullanan IBM MQ classes for JMS uygulamaları, Java Native Interface (JNI) kitaplıklarına erişime izin veren bir ortamda çalıştırılmalıdır.

## Başlamadan önce

WebSphere Application Server ortamının kullanılmasına ilişkin ek bilgi için [Yerel kitaplık bilgileri ile IBM MQ ileti alışverişi sağlayıcısının yapılandırılması](#) başlıklı konuya bakın.

## Bu görev hakkında

To set up this environment, you must configure the environment's library path so that the Java virtual machine (JVM) can load the mqjbn library before you start the IBM MQ classes for JMS application.

IBM MQ , iki Java Native Interface (JNI) kitaplığı sağlar:

### **mqjbn**

Bu kitaplık, bağ tanımları taşıma özelliğini kullanarak kuyruk yöneticisine bağlanan uygulamalar tarafından kullanılır. Bu arabirim, IBM MQ classes for JMS ile kuyruk yöneticisi arasındaki arabirimi sağlar. IBM MQ 9.0 ile kurulan mqjbn kitaplığı, herhangi bir IBM MQ 9.0 (ya da önceki) kuyruk yöneticisine bağlanmak için kullanılabilir.

### **mqjexitstub02**

Bir uygulama istemci aktarımı kullanılarak bir kuyruk yöneticisine bağlandığında ve Javadişında bir dilde yazılmış bir kanal çıkış programı kullanıyorsa, mqjexitstub02 kitaplığı IBM MQ classes for JMS tarafından yüklenir.

Bazı altyapılarda, IBM MQ bu JNI kitaplıklarının 32 bit ve 64 bit sürümlerini kurar. Her bir platforma ilişkin kitaplıkların yeri [Tablo 1](#) 'de gösterilir.

Çizelge 8. Her altyapıya ilişkin IBM MQ classes for JMS kitaplıklarının yeri

Platform	IBM MQ classes for JMS kitaplıklarını içeren dizin
<b>AIX</b> AIX	<i>MQ_INSTALLATION_PATH</i> /java/lib (32 bit kitaplık) <i>MQ_INSTALLATION_PATH</i> /java/lib64 (64 bit kitaplık)
<b>Linux</b> Linux (POWER, x86-64 ve zSeries s390x altyapıları)	
<b>Solaris</b> Solaris (x86-64 ve SPARC platformları)	
<b>Windows</b> Windows	<i>MQ_INSTALLATION_PATH</i> \java\lib (32 bit kitaplık) <i>MQ_INSTALLATION_PATH</i> \java\lib64 (64 bitlik kitaplıklar)
<b>z/OS</b> z/OS	<i>MQ_INSTALLATION_PATH</i> /mqm/V9R1M0/java/lib (31 bit ve 64 bitlik kitaplıklar)

*MQ\_INSTALLATION\_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

**Not:** **z/OS** z/OS' ta, 31 bit ya da 64 bit Java virtual machine (JVM) kullanabilirsiniz. Hangi JNI kitaplıklarının kullanılacağını belirtmenize gerek yoktur; IBM MQ classes for JMS , hangi JNI kitaplıklarının yükleneceğini kendisinin belirleyebileceğini belirtir.

## Yordam

1. JVM ' nin **java.library.path** özelliğini yapılandırın; bu, şu iki şekilde yapılabilir:

- Aşağıdaki örnekte gösterildiği gibi, JVM bağımsız değişkenini belirterek:

```
-Djava.library.path=path_to_library_directory
```

**Linux** Örneğin, varsayılan konum kurulumu için Linux üzerindeki bir 64 bit JVM için şunu belirtin:

```
-Djava.library.path=/opt/mqm/java/lib64
```

- Shell 'in ortamını yapılandırarak, JVM kendi `java.library.path`' ı ayarlayacak. Bu yol, platforma ve IBM MQ' un kurulu olduğu konumlara göre değişir. Örneğin, 64 bit JVM ve varsayılan bir IBM MQ kurulumu için aşağıdaki ayarları kullanabilirsiniz:

```
AIX export LIBPATH=/usr/mqm/java/lib64:$LIBPATH
```

```
Solaris 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 şekilde yapılandırılmadığında gördüğünüz kural dışı durum yığınının bir örneği aşağıdaki gibidir:

```
Nedeni: com.ibm.mq.jmqi.local.LocalMQ$4: CC=2;RC=2495;  
AMQ8598: WebSphere MQ yerel JNI kitaplığının yüklenmesi başarısız oldu: 'mqjbn'd'.  
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)
```

```

at com.ibm.mq.jmqi.local.LocalServer.<init>(LocalServer.java:230)
sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Yönteminde)

sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:86)

sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java: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)
at
com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection(WMQConnectionFactory.java:8437)
... 7 daha fazla
Nedeni: java.lang.UnsatisfiedLinkError: mqjbn ( java.library.path içinde 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 *uygulama-adi* , çalıştırılacak IBM MQ classes for JMS uygulamasının adıdır.

IBM MQ neden kodu 2495 (MQRC\_MODULE\_NOT\_FOUND) içeren bir kural dışı durum, aşağıdaki durumlarda IBM MQ classes for JMS tarafından verilir:

- The IBM MQ classes for JMS application is run in a 32-bit Java runtime environment, and a 64-bit environment has been set up for the IBM MQ classes for JMS, as the 32-bit Java runtime environment is unable to load the 64-bit Java Native Library.
- IBM MQ classes for JMS uygulaması 64 bit Java runtime environment' te çalıştırılır ve 64 bit Java runtime environment 32 bit Java Yerel Kitaplığı yükleyemediğinden, IBM MQ classes for JMS için 32 bit ortam ayarlanmıştır.

#### IBM MQ classes for JMS yapılandırma dosyası

Bir IBM MQ classes for JMS yapılandırma dosyası, IBM MQ classes for JMS' u yapılandırmak için kullanılan özellikleri belirtir.

**Not:** Yapılanış kütüğünde tanımlı olan özellikler, JVM sistem özellikleri olarak da ayarlanabilir. Bir özellik hem yapılandırma dosyasında, hem de bir sistem özelliği olarak ayarlandıysa, sistem özelliği öncelikli olur. Bu nedenle, gerekiyorsa, yapılanış kütüğündeki herhangi bir özelliği, **java** komutunda bir sistem özelliği olarak belirterek geçersiz kılabilirsiniz.

Bir IBM MQ classes for JMS yapılandırma dosyasının biçimi, standart bir Java özellikler dosyasının biçimidir. IBM MQ classes for JMS kuruluş dizininin bin alt dizininde `mqjbn.config` adlı örnek bir yapılanış kütüğü belirtildi. Bu dosya, desteklenen tüm özellikleri ve bunların varsayılan değerlerini belirler.

Bir IBM MQ classes for JMS yapılandırma dosyasının adını ve konumunu seçebilirsiniz. Uygulamanıza başladığınızda, aşağıdaki biçimi kullanarak 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 yapılandırma dosyasının adını ve yerini belirten bir URL ' dir. Şu tiplerin URL ' leri desteklenir: http, file, ftp ve jar.

Aşağıda bir **java** komutu örneği yer alıyor:

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config MyAppClass
```

This command identifies the IBM MQ classes for JMS configuration file as the file `D:\mydir\myjms.config` on the local Windows system.



Bir uygulama başlatıldığında, IBM MQ classes for JMS , yapılandırma dosyasının içeriğini okur ve belirtilen özellikleri bir iç özellik deposunda saklar. **java** komutu bir yapılandırma dosyasını tanıtmıyorsa ya da yapılandırma dosyası bulunamazsa, IBM MQ classes for JMS tüm özellikler için varsayılan değerleri kullanır.

Bir uygulama ile kuyruk yöneticisi ya da aracı arasındaki desteklenen aktarımlardan herhangi biriyle bir IBM MQ classes for JMS yapılandırma dosyası kullanılabilir.

## Bir IBM MQ MQI client yapılandırma dosyasında belirtilen özelliklerin geçersiz kılınması

Bir IBM MQ MQI client yapılandırma dosyası, IBM MQ classes for JMS' u yapılandırmak için kullanılan özellikleri de belirtebilir. Ancak, bir IBM MQ MQI client yapılandırma dosyasında belirtilen özellikler yalnızca, bir uygulama istemci kipinde bir kuyruk yöneticisine bağlandığında geçerlidir.

If required, you can override any attribute in a IBM MQ MQI client configuration file by specifying it as a property in a IBM MQ classes for JMS configuration file. To override an attribute in a IBM MQ MQI client configuration file, use an entry with the following format in the IBM MQ classes for JMS configuration file:

```
com.ibm.mq.cfg. stanza. propName = propValue
```

Girdideki değişkenler aşağıdaki anlamlara sahiptir:

### **stanza**

Özniteliği içeren IBM MQ MQI client yapılandırma dosyasındaki stanza adının adı

### **propName**

The name of the attribute as specified in the IBM MQ MQI client configuration file

### **propValue**

IBM MQ MQI client yapılandırma dosyasında belirtilen özneniteliğin değerini geçersiz kılan özelliğin değeri

Alternatively, you can override an attribute in an IBM MQ MQI client configuration file by specifying the property as a system property on the **java** command. Özneniteliği bir sistem özneniteliği olarak belirtmek için önceki biçimi kullanın.

Bir IBM MQ MQI client yapılandırma dosyasında yalnızca aşağıdaki öznenitelikler IBM MQ classes for JMS ile ilgilidir. Diğer öznenitelikleri belirtmiş ya da geçersiz kıyorsanız, bu herhangi bir etkisi olmaz. Özellikle, İstemci yapılandırma dosyasının STANA kısmı içindeki ChannelDefinitionFile ve ChannelDefinitionDirectory ' nin kullanılmadığını lütfen unutmayın. CCDT ' yi IBM MQ classes for JMS ile nasıl kullanabilmeye ilişkin ayrıntılar için “Using a client channel definition table with IBM MQ classes for JMS” sayfa 262 konusuna bakın.

Çizelge 9. İstemci yapılandırma dosyasında hangi özneniteliğin bulunduğunu içeren	
Stanza	Öznenitelik
<a href="#">İstemci yapılandırma dosyasının STANA kısmı</a>	Put1DefaultAlwaysSync
<a href="#">İstemci yapılandırma dosyasının STANA kısmı</a>	DefRecon
<a href="#">İstemci yapılandırma dosyasının STANA kısmı</a>	ReconDelay
<a href="#">İstemci yapılandırma dosyasının STANA kısmı</a>	PasswordProtection
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	ExitsDefaultYolu
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	ExitsDefaultPath64
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	JavaExitsClasspath
<a href="#">İstemci yapılandırma dosyasının JMQUI kısmı</a>	useMQCSPauthentication

Çizelge 9. İstemci yapılandırma dosyasında hangi öz niteliğin bulunduğunu içeren (devamı var)

Stanza	Öznitelik
İstemci yapılandırma dosyasının <u>MessageBuffer</u> kısmı	MaximumSize
İstemci yapılandırma dosyasının <u>MessageBuffer</u> kısmı	PurgeTime
İstemci yapılandırma dosyasının <u>MessageBuffer</u> kısmı	UpdatePercentage
İstemci yapılandırma dosyasının TCP stanzası	ClntRcvBufSize
İstemci yapılandırma dosyasının TCP stanzası	ClntSndBufSize
İstemci yapılandırma dosyasının TCP stanzası	Bağlantı_Zamanaşımı
İstemci yapılandırma dosyasının TCP stanzası	KeepAlive

IBM MQ MQI client yapılandırmasındaki ek ayrıntılar için [Yapılandırma kütüğü](#) kullanarak istemci yapılandırılması başlıklı konuya bakın.

#### Java Standard Environment Trace stanza

IBM MQ classes for JMS izleme olanağını yapılandırmak için Java Standard Environment Trace Settings stanza olanağını kullanın.

#### **com.ibm.msg.client.commonservices.trace.outputName = traceOutputAd**

*traceOutputName* , izleme çıkışının gönderileceği dizin ve dosya adıdır.

Varsayılan olarak izleme bilgileri, uygulamanın yürürlükteki çalışma dizinindeki bir izleme dosyasına yazılır. İzleme dosyasının adı, uygulamanın çalışmakta olduğu ortama bağlıdır:

- 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`.
- From IBM MQ 9.0.0 Fix Pack 2, if the application has loaded the IBM MQ classes for JMS from the JAR file `com.ibm.mqjms.jar`, trace is written to a file called `mqjava_%PID%.trc`.
- From IBM MQ 9.0.0 Fix Pack 2, if the application has loaded the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.trc`.
- **V 9.1.0.5** **V 9.1.5** From IBM MQ 9.1.5 ve IBM MQ 9.1.0 Fix Pack 5, if the application has loaded the IBM MQ classes for JMS from the JAR file `com.ibm.mqjms.jar`, trace is written to a file called `mqjava_%PID%.cl%u.trc`.
- **V 9.1.0.5** **V 9.1.5** From IBM MQ 9.1.5 ve IBM MQ 9.1.0 Fix Pack 5, if the application has loaded the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.cl%u.trc`.

Burada `%PID%` , izlenmekte olan uygulamanın işlem tanıtıcısıdır ve `%u` , farklı Java sınıf yükleyicileri altında izleme çalıştıran iş parçacıkları arasında dosyaları ayırt etmek için kullanılan benzersiz bir sayıdır.

If a process ID is unavailable, a random number is generated and prefixed with the letter `f`. Süreç tanıtıcısını belirttiğiniz bir dosya adına eklemek için `%PID%dizgisini` kullanın.

Başka bir dizin belirtirseniz, bu dizin var olmalıdır ve bu dizin için yazma izniniz olmalıdır. Yazma izniniz yoksa, izleme çıkışı `System.err` olarak yazılır.

#### **com.ibm.msg.client.commonservices.trace.include = includeList**

*includeList* , izlenecek paketlerin ve sınıfların bir listesidir ya da özel değerler ALL ya da NONE.

Paketi ya da sınıf adlarını noktalı virgöl ( ; ) ile ayırın. *includeList* varsayılan olarak ALLdeğerine ayarlanır ve IBM MQ classes for JMSiçindeki tüm paketleri ve sınıfları izler.

**Not:** Bir paket dahil edebilir, ancak o paketin alt paketlerini dışlayabilirsiniz. For example, if you include package a . b and exclude package a . b . x, the trace includes everything in a . b . y and a . b . z, but not a . b . x or a . b . x . 1.

**com.ibm.msg.client.commonservices.trace.exclude = *excludeList***

*excludeList* , izlenmeyen paketler ve sınıfların bir listesidir ya da özel değerler ALL ya da NONE.

Paketi ya da sınıf adlarını noktalı virgöl ( ; ) ile ayırın. *excludeList* varsayılan olarak NONEdeğerine ayarlanır ve bu nedenle, IBM MQ classes for JMS içindeki hiçbir paketi ve sınıfı izlemekten dışlanmaz.

**Not:** Bir paketi dışlayabilir, ancak o paketin alt paketlerini de içerebilirsiniz. For example, if you exclude package a . b and include package a . b . x, the trace includes everything in a . b . x and a . b . x . 1, but not a . b . y or a . b . z.

Hem içerilen hem de dışlanan paket ya da sınıf, aynı düzeyde belirtilmiş ve dışlanmış olarak dahil edilir.

**com.ibm.msg.client.commonservices.trace.maxBytes = *maxArrayByte***

*maxArrayBytes* , herhangi bir bayt dizisinden izlenen bayt sayısı üst sınıştır.

*maxArrayBytes* pozitif bir tamsayıya ayarlandıysa, izleme dosyasına yazılacak bayt dizideki bayt sayısını sınırlar. *maxArrayBytes* yazıldıktan sonra bayt dizisini keser. *maxArrayBytes* ayarı sonucunda elde edilen izleme dosyası büyüklüğü azaltılıyor ve izleme işlemi, uygulamanın performansında etkisini azaltır.

Bu özellik için 0 değeri, izleme dosyasına herhangi bir byte dizisinin içeriğinin hiçbirinin gönderilmemesinin anlamına gelir.

Varsayılan değer -1değeridir. Bu değer, izleme dosyasına gönderilen bayt dizisindeki bayt sayısındaki herhangi bir sınırı kaldırır.

**com.ibm.msg.client.commonservices.trace.limit = *maxTraceByte***

*maxTraceBytes* , bir izleme çıkış dosyasına yazılan bayt sayısı üst sınıştır.

*maxTraceBytes* , *traceCycles* ile çalışır. Yazılan izleme baytlarının sayısı sınıra yakınsa, dosya kapatılır ve yeni bir izleme çıkış dosyası başlatılır.

0 değeri, bir izleme çıkış dosyasının sıfır uzunluğuna sahip olduğu anlamına gelir. Varsayılan değer -1değeridir. Bu, bir izleme çıkış dosyasına yazılacak veri miktarının sınırsız olduğu anlamına gelir.

**com.ibm.msg.client.commonservices.trace.count = *traceCycles***

*traceCycles* , çevrim yoluyla çevrilecek izleme çıkış dosyalarının sayısıdır.

Yürürlükteki izleme çıkış dosyası *maxTraceBytes* ile belirtilen sınıra ulaşırsa, dosya kapatılır. Ek izleme çıkışı, sonraki izleme çıkış dosyasına sırayla yazılır. Her izleme çıkış dosyası, dosya adının sonuna eklenen sayısal bir sonek tarafından ayırt edilir. Yürürlükteki ya da en son izleme çıkış dosyası mqjms . trc . 0, sonraki en son izleme çıkış dosyası mqjms . trc . 1. Daha eski izleme dosyaları, sınırlamaya kadar aynı numaralandırma örüntülerini izler.

*traceCycles* varsayılan değeri 1 'dir. *traceCycles* 1 ise, yürürlükteki izleme çıkış dosyası büyüklük üst sınırına ulaştığında, dosya kapatılır ve silinir. Aynı adı taşıyan yeni bir izleme çıkış dosyası başlatılmış. Bu nedenle, bir kerede tek bir izleme çıkış dosyası vardır.

**com.ibm.msg.client.commonservices.trace.parameter = *traceParameters***

*traceParameters* yöntem değiştirgelerinin ve dönüş değerlerinin izlemenin içerilip içerilmeyeceğini denetler.

*traceParameters* varsayılan olarak TRUEdeğerine ayarlanır. *traceParameters* FALSEolarak ayarlandıysa, yalnızca yöntem imzaları izlenir.

**com.ibm.msg.client.commonservices.trace.startup = *başlangıç***

Kaynakların ayrıldığı, IBM MQ classes for JMS ' un kullanıma hazırlama aşaması vardır. Ana izleme olanağı, kaynak ayırma aşaması sırasında kullanıma hazırlanır.

*startup* TRUE olarak ayarlandıysa, başlatma izlemesi kullanılır. İzleme bilgileri hemen üretilir ve izleme olanağı da dahil olmak üzere tüm bileşenlerin kuruluşunu içerir. Başlatma izleme bilgileri, yapılandırma sorunlarını tanılamak için kullanılabilir. Başlatma izleme bilgileri her zaman `System.err`'a yazılır.

*startup* varsayılan olarak FALSE değerine ayarlanır.

*startup*, kullanıma hazırlama işlemi tamamlanmadan önce denetlenir. Bu nedenle, yalnızca komut satırındaki özelliği bir Java sistem özelliği olarak belirtin. Bunu IBM MQ classes for JMS yapılandırma dosyasında belirtmeyin.

#### **com.ibm.msg.client.commonservices.trace.compress = compressedTrace**

İzleme çıkışı sıkıştırmak için *compressedTrace*, TRUE olarak ayarlayın.

*compressedTrace* varsayılan değeri FALSE değeridir.

*compressedTrace* TRUE olarak ayarlandıysa, izleme çıkışı sıkıştırılır. Varsayılan izleme çıkışı dosyası adı, `.truzantısına` sahiptir. If compression is set to FALSE, the default value, the file has the extension `.trc` to indicate it is uncompressed. Ancak, izleme çıkışının dosya adı *traceOutputName* içinde belirtilmiş olsa da, bu ad kullanılırsa, dosyaya sonek uygulanmaz.

Sıkıştırılmış izleme çıkışı, sıkıştırılmamış boyuttan küçük. Daha az G/Ç olduğundan, sıkıştırılmamış izlemenin daha hızlı bir şekilde yazılabiliyor. Sıkıştırılmış izleme, IBM MQ classes for JMS 'un performansı, sıkıştırılmamış izlemenin dışında daha az etkiye sahiptir.

*maxTraceBytes* ve *traceCycles* ayarlandıysa, birden çok düz dosya yerine birden çok sıkıştırılmış izleme dosyası oluşturulur.

IBM MQ classes for JMS denetimli olarak sona erdirilirse, sıkıştırılmış izleme dosyası geçerli olmayabilir. Bu nedenle, izleme sıkıştırması yalnızca IBM MQ classes for JMS denetimli bir şekilde kapatıldığında kullanılmalıdır. Yalnızca araştırılmakta olan sorunlar JVM 'nin beklenmedik bir şekilde durmasına neden değilse, izleme sıkıştırmayı kullanın. `System.Halt()` kapatma ya da olağandışı olmayan, denetimsiz JVM sonlandırıcılarıyla sonuçlanabilen sorunları tanımlarken 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 alt düzeylerin tümünü içerir. Örneğin, izleme düzeyi TRACE\_INFO olarak ayarlandıysa, izleme düzeyi TRACE\_EXCEPTION, TRACE\_WARNING ya da TRACE\_INFO tanımlı düzeyli herhangi bir izleme noktası yazılır. Diğer tüm izleme noktaları dışlanır.

#### **com.ibm.msg.client.commonservices.trace.standalone = standaloneTrace**

*standaloneTrace*, bir WebSphere Application Server ortamında IBM MQ JMS istemci izleme hizmetinin kullanılıp kullanılmayacağını denetler.

*standaloneTrace* değeri TRUE olarak ayarlanırsa, izleme yapılandırmasını saptamak için IBM MQ JMS istemci izleme özellikleri kullanılır.

If *standaloneTrace* is set to FALSE, and the IBM MQ JMS client is running in an WebSphere Application Server container, the WebSphere Application Server trace service is used. Oluşturulan izleme bilgileri, uygulama sunucusunun izleme ayarlarına bağlı olarak değişir.

*standaloneTrace* varsayılan değeri FALSE değeridir.

### *Günlüğe kaydetme kısmı*

IBM MQ classes for JMS günlük olanağını yapılandırmak için Logging stanza programını kullanın.

Günlüğe kaydetme kısmında aşağıdaki özellikler yer alır:

#### **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 ' un çalışmakta olduğu Java Runtime Environment için yürürlükteki çalışma dizinine yazılan mqjms . Logdeğeridir.

Özellik aşağıdaki değerlerden birini alabilir:

- tek bir yol adı
- Yol adlarının virgülle ayrılmış listesi (tüm kütüklerde tüm veriler günlüğe kaydedilir)

Her yol adı, mutlak ya da görel 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 verilerinin günlüğe kaydedileceği herhangi bir çağrıdan günlüğe kaydedilen bayt sayısı üst sınırı.

#### **Pozitif tamsayı**

Günlük çağrısı başına bu bayt değerine kadar veri yazılır.

**0**

Veri yazılmadı.

**-1**

Sınırsız veri yazılıdır (varsayılan).

#### **com.ibm.msg.client.commonservices.log.limit**

Herhangi bir 1 günlük dosyasına yazılan bayt sayısı üst sınırı (varsayılan değer 262144).

#### **Pozitif tamsayı**

Veri, günlük dosyası başına bu değere kadar yazılır.

**0**

Veri yazılmadı.

**-1**

Sınırsız veri yazılıdır.

#### **com.ibm.msg.client.commonservices.log.count**

Çevrilecek günlük kütüklerinin sayısı. Her dosya

com.ibm.msg.client.commonservices.trace.limit izleme sayısına ulaştığında sonraki dosyada başlar, varsayılan değer 3 'tür.

#### **Pozitif tamsayı**

Çevrilecek dosya sayısı.

**0**

Tek bir dosya.

### *Java SE Specifics stanza*

Use the Java SE Specifics stanza to configure properties that are used when the IBM MQ classes for JMS are being used in a Java Standard Edition environment.

#### **com.ibm.msg.client.commonservices.j2se.produceJavaCore = TRUE|FALSE**

IBM MQ classes for JMS , bir FDC dosyası oluşturduktan hemen sonra JavaCore dosyasının hemen yazılıp yazılmayacağını belirler. Bu değer TRUE olarak ayarlanırsa, IBM MQ classes for JMS ' in çalışmakta olduğu Java Runtime Environment ' in çalışma dizininde JavaCore dosyası üretilir.

#### **TRUE**

Generate JavaCore, subject to the Java Runtime Environment's ability to do so.

## YANLIŞ

JavaCore oluşturmayın; bu varsayılan değerdir.

### *IBM MQ Özellikleri-Stanza*

IBM MQ classes for JMS ' in IBM MQ ile nasıl etkileşimde bulunacağını etkileyen özellikleri ayarlamak için IBM MQ Properties stanza olanağını kullanın.

### **com.ibm.msg.client.wmq.compat.base.internal.MQQueue.smallMsgsBufferReductionThreshold**

When an application that uses the IBM MQ classes for JMS is connecting to an IBM MQ queue manager using IBM MQ messaging provider migration mode, the IBM MQ classes for JMS uses a default buffer size of 4 KB when it receives messages. Uygulamanın almaya çalıştığı ileti 4 KB ' den büyükse, IBM MQ classes for JMS arabelleği yeniden boyutlandırır ve iletiyi sığdıracak kadar büyük olur. Daha sonra, daha büyük arabellek büyüklüğü daha sonra gelen iletiler alındığında kullanılır.

Bu özellik, arabellek büyüklüğünün 4 KB ' ye geri düşmesini denetler. Varsayılan olarak, daha büyük arabellek büyüklüğünden daha az sayıda art arda ileti alındığında arabellek büyüklüğü 4 KB ' ye geri düşer. Bir ileti alındığında arabellek büyüklüğünü 4 KB ' ye geri döndürmek için, özelliği 0 değerine ayarlayın.

**0**

Arabellek her zaman varsayılan boyuta sıfırlanır.

**10**

Bu varsayılan değerdir. Arabellek, onuncu iletinden sonra yeniden boyutlandırılır.

### **com.ibm.msg.client.wmq.receiveConversionCCSID**

IBM MQ classes for JMS kullanan bir uygulama, IBM MQ ileti alışverişi sağlayıcısı olağan kipini kullanarak bir IBM MQ kuyruk yöneticisine bağlanıldığında, kuyruk yöneticisinden ileti almak için kullanılan MQMD yapısındaki varsayılan CCSID değerini geçersiz kılmak için `receiveConversionCCSID` özelliği ayarlanabilir. Varsayılan olarak, MQMD, 1208 olarak ayarlanmış bir CCSID alanı içerir; ancak, örneğin, kuyruk yöneticisi iletileri bu kod sayfasına dönüştüremiyorsa, bu değer değiştirilebilir.

Geçerli değerler, geçerli bir CCSID numarası ya da aşağıdaki değerlerden biri olabilir:

**-1**

Altyapıyı varsayılan olarak kullanın.

**1208**

Bu varsayılan değerdir.

### *İstemci kipi specifics stanza*

Use the Client-mode specifics stanza to specify properties that are used when the IBM MQ classes for JMS connect to a queue manager that is using the CLIENT transport.

### **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. Değer alt sınırı 3000 'dir ve alt değerler, bu alt sınır değeriyle aynı şekilde ele alınır.

### *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 biçimiyle ilgili değişiklikleri tanıtır. IBM MQ 8.0 , değiştirilen bu davranışları önceki somutlamalara geri döndürmek için `TRUE` olarak ayarlanabilen `com.ibm.mq.jms.SupportMQExtensions` özelliğini içerir. Değiştirilen davranışların tersine çevrilmesi, bazı JMS 2.0 uygulamaları için ve ayrıca JMS 1.1 API kullanan ancak IBM MQ 8.0 IBM MQ classes for JMS' a karşı çalıştırılan bazı uygulamalar için gerekli olabilir.

## TRUE

SupportMQExtensions ayarı TRUEolarak ayarlandığında, aşağıdaki üç işlev alanı geri çevrilir:

### İleti önceliği

İletilere öncelik atanabilir: 0 - 9. JMS 2.0öncesinde iletiler, bir kuyruğun varsayılan önceliğinin kullanıldığını gösteren -1değerini de kullanabilirdi. JMS 2.0 does not allow a message priority of -1 to be set. SupportMQExtensions değerinin açılması, -1 değerinin kullanılmasını sağlar.

### İSTEMCİ TANITICISI

JMS 2.0 belirtimi, boş değerli olmayan istemci tanıtıcılarının bir bağlantı gerçekleştirdiklerinde benzersizlik olup olmadığını denetimlerini gerektirir. SupportMQExtensions' in açılması, bu gereksinimin göz ardı edildiğine ve bir istemci tanıtıcısının yeniden kullanılabileceği anlamına gelir.

### NoLocal

JMS 2.0 belirtimi, bu sabit açık olduğunda, bir tüketici aynı istemci tanıtıcısı tarafından yayınlanan iletileri alamayabilmenizi gerektirir. JMS 2.0öncesinde, bu öznelik, kendi bağlantısı tarafından yayınlanan iletilerin alınmasını önlemek için bir abonede ayarlanmıştır. SupportMQExtensions ' in açılması bu davranışı önceki somutlamaya geri çevirir.

## YANLIŞ

Davranış değişiklikleri korunur.

### com.ibm.msg.client.jms.ByteStreamReadOnlyAfterSend= TRUE|FALSE

From IBM MQ 8.0.0 Fix Pack 2, after an application has sent a Bytes or Stream message, IBM MQ classes for JMS can set the state of the message that has just been sent to either read only, or write only.

## TRUE

Nesneler, gönderildikten sonra salt okunur olarak ayarlanır. Bu değerin ayarlanması, JMS 2.0 belirtimiyle uyumluluğu sağlar.

## YANLIŞ

Nesneler, yalnızca gönderildikten sonra yazılır olarak ayarlanır. Bu varsayılan değerdir.

### İlgili kavramlar

["SupportMQExtensions özelliği" sayfa 304](#)

JMS 2.0 belirtimi, belirli davranışların çalışma biçimiyle ilgili değişiklikleri tanıtır. IBM MQ 8.0 ve daha sonra, bu değiştirilen davranışları önceki somutlamaya geri döndürmek için TRUEolarak ayarlanabilen com.ibm.mq.jms.SupportMQExtensionsözelliğini içerir.

*z/OSüzerinde IBM MQ classes for JMS için STEPLIB yapılışı*

z/OS'da, yürütme sırasında kullanılan STEPLIB' in IBM MQ SCSQAUTH ve SCSQANLE kitaplıklarını içermesi gerekir. Başlatma JCL ' de ya da .profile dosyasını kullanarak bu kitaplıkları belirtin.

From UNIX and Linux System Services, you can add these using a line in your .profile as shown in the following code snippet, replacing th1qua1 with the high-level data set qualifier that you chose when installing IBM MQ:

```
export STEPLIB=th1qua1.SCSQAUTH:th1qua1.SCSQANLE:$STEPLIB
```

Diğer ortamlarda, genellikle başlatma JCL ' yi, STEPLIB bitişirmesine SCSQAUTH ve SCSQANLE içerecek şekilde düzenlemeniz gerekir.

```
STEPLIB DD DSN=th1qua1.SCSQAUTH,DISP=SHR  
DD DSN=th1qua1.SCSQANLE,DISP=SHR
```

*IBM MQ classes for JMS ve yazılım yönetimi araçları*

Apache Maven gibi yazılım yönetimi araçları IBM MQ classes for JMSile kullanılabilir.

Birçok büyük geliştirme kuruluşu, üçüncü kişi kitaplıklarının havuzlarını merkezi olarak yönetmek için bu araçları kullanır.

IBM MQ classes for JMS , 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 kurulumu 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 dağıtımına 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:

- `com.ibm.mq.allclient.jar`.
- IBM MQ classes for JMS kullanıyorsanız `jms.jar` gereklidir.
- Bir dosya sistemi JNDI bağlamında saklanan IBM MQ classes for JMS ve JMS denetimli nesnelere erişiyorsanız, `fscontext.jar` gereklidir.
- Bir dosya sistemi JNDI bağlamında saklanan IBM MQ classes for JMS ve JMS denetimli nesnelere erişiyorsanız `providerutil.jar`.

IBM MQ 9.0' den Bouncy Castle güvenlik sağlayıcısı ve CMS destek JAR dosyaları gereklidir. Daha fazla bilgi için bkz. [“IBM MQ classes for JMS için kurulu olan” sayfa 81](#) ve [IBM dışı JRE ' ler için destek](#).

### **Running IBM MQ classes for JMS applications under the Java security manager**

IBM MQ classes for JMS , Java güvenlik yöneticisi etkinleştirilmiş olarak çalışabilir. Uygulamaları Java security manager etkin bir şekilde başarıyla çalıştırmak için, Java virtual machine (JVM) olanağını uygun bir ilke yapılandırma dosyasıyla yapılandırmalısınız.

Uygun bir ilke tanımlama dosyası oluşturmanın en kolay yolu, Java runtime environment (JRE) ile sağlanan ilke yapılandırma dosyasını değiştirmemektir. Çoğu sistemde bu dosya, JRE dizininize göre `lib/security/java.policy` dizininde yer almaktadır. İlke yapılandırma dosyasını, tercih ettiğiniz düzenleyiciyi kullanarak ya da JRE ' nize birlikte verilen ilke aracı programını kullanarak düzenleyebilirsiniz.

#### **Önemli:**

Mümkün olduğunda, *izin listesi* terimi *beyaz listeler*iminin yerini almıştır. IBM MQ 9.0 ve sonraki yayınlarda, bu konuda (**`com.ibm.mq.jms.*`**) belirtilen Java sistem özelliği adlarını içerir. Var olan bir yapılandırmayı değiştirmeniz gerekmez. Önceki sistem özellik adları da çalışmaya devam eder.

Uygulamanıza sahip Java security manager mekanizmasını kullanıyorsanız, aşağıdaki izinleri vermeniz gerekir:

- FilePermission on any allowlist file that you use, with read permission for ENFORCEMENT mode, write permission for DISCOVER mode.
- **`com.ibm.mq.jms.allowlist`**, **`com.ibm.mq.jms.allowlist.discover`** ve **`com.ibm.mq.jms.allowlist.mode`** özelliklerinde PropertyPermission (okuma).

Daha fazla bilgi için bkz [“Allowlisting kavramları” sayfa 117](#).

### **Örnek ilke yapılandırma dosyası**

Aşağıda, IBM MQ classes for JMS ' un varsayılan güvenlik yöneticisi altında başarılı bir şekilde çalışmasını sağlayan bir ilke yapılandırma dosyası örneği yer alıyor. Bu dosyanın, belirli dosyaların ve dizinlerin konumlarını belirtmek için özelleştirilmesi gerekir: `MQ_INSTALLATION_PATH` , IBM MQ ' un kurulu olduğu üst düzey dizini temsil eder, `MQ_DATA_DIRECTORY` MQ veri dizininin yerini gösterir ve `QM_NAME` , erişimin yapılandırılmakta olduğu 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";
};

```

Örnekler, `grant` deyimini IBM MQ sınıfları için gereken izinleri içerir. Bu izin ifadelerini politik konfigürasyon dosyasına eklerken, dosya yollarını kurulumun yapıldığı yere ve uygulamaların saklandığı yere göre değiştirmeniz gerekebilir.

IBM MQ sınıfları için verilen örnek uygulamalar ve bunları çalıştırmak için komut dosyaları, güvenlik yöneticisini etkinleştirmez.

## **IBM MQ classes for JMS uygulamaları için kuruluş sonrası ayarları**

This topic tells you what authorities IBM MQ classes for JMS applications need in order to access the resources of a queue manager. Ayrıca, bağlantı kipleri tanıtlır ve uygulamaların istemci kipinde bağlanabilmesi için kuyruk yöneticisinin nasıl yapılandırılacağı anlatılır.

**IBM MQ benioku dosyasını kontrol etmeyi unutmayın. Bu konuyla ilgili bilgilerin yerini alan bilgiler içerebilir.**

*Objects used by JMS that require authorization for non-privileged users*

Ayrıcalıklı olmayan kullanıcılara, JMStarafından kullanılan kuyruklara erişmek için yetki verilmesine gerek vardır. Her JMS uygulamasının, çalıştığı kuyruk yöneticisi için yetkilendirmesi gerekir.

IBM MQ' ta erişim denetimi hakkında ayrıntılı bilgi için bkz. [Güvenliği ayarlama](#).

IBM MQ classes for JMS uygulamaları, kuyruk yöneticisi için connect ve inq yetkisine gereksinim duyarlar. **setmqaut** denetim komutunu kullanarak uygun yetkiler ayarlayabilirsiniz. Örneğin:

```
setmqaut -m QM1 -t qmgr -g jmsappsgrout +connect +inq
```

Noktadan noktaya iletişim etki alanı için aşağıdaki yetkiler gereklidir:

- MessageProducer nesnelere tarafından kullanılan kuyruklar put yetkisine gereksinim duyarlar.
- MessageConsumer ve QueueBrowser nesnelere göre kullanılan kuyruklar get, inq ve browse yetkilerine gereksinim duyarlar.
- QueueSession.createTemporaryQueue () yönteminin, QueueConnectionFactory nesnesinin TEMPMODEL özelliği tarafından belirlenen model kuyruğuna erişmesi gerekir. Varsayılan olarak bu model kuyruğu SYSTEM.TEMP.MODEL.QUEUE.

Bu kuyruklardan herhangi birinin diğer ad kuyrukları varsa, bunların hedef kuyrukları sorgu yetkisi gerektirir. Hedef kuyruk bir küme kuyruğıysa, aynı zamanda göz atma yetkisi de gerektirir.

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 kipine ilişkin ek bilgi için [JMS PROVIDERVERSION özelliğinin yapılandırılması](#) başlıklı konuya bakın.

Buna ek olarak, IBM MQ classes for JMS bu kipteki bir kuyruk yöneticisine bağlanıyorsa, iletilerin yayınlandığı uygulamaların TopicConnectionFactory ya da konu nesnesi tarafından belirtilen akış kuyruğuna erişmesi gerekir. Varsayılan değer olarak, bu kuyruk SYSTEM.BROKER.DEFAULT.STREAM.

ConnectionConsumer, IBM MQ Resource Adapter ya da WebSphere Application Server IBM MQ Messaging Provider kullanıyorsanız, ek yetkilendirmeye gerek olabilir.

ConnectionConsumer tarafından okunacak kuyrukların get, inq ve browse yetkilerine sahip olması gerekir. Sistemin öldüğü mektup kuyruğı ve ConnectionConsumer tarafından kullanılan geri yedekleme kuyruğunda ya da rapor kuyruğunun tak ve düzgeçiş yetkilerine sahip olması gerekir.

Bir uygulama yayınlama/abone olma ileti alışverişi gerçekleştirmek için IBM MQ ileti alışverişi sağlayıcısı olağan kipini kullandığında, uygulama kuyruk yöneticisi tarafından sağlanan tümleşik yayınlama/abone

olma işlevinden yararlanır. Kullanılan konuların ve kuyrukların güvenliğini sağlamaya ilişkin bilgi için bkz. [Güvenliği yayınla/abone ol](#) .

### *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 JMS sürümüyle 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 86](#) .

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 JMS tarafından kullanılan bağlantı seçeneklerini değiştirmek için [CONNNOPT](#) Connection Factory özelliğini değiştirin.

Bağlantı seçenekleriyle ilgili daha fazla bilgi için bkz. [“MQCONNX çağrısını kullanarak kuyruk yöneticisine bağlanma” sayfa 775](#)

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](#).kismına bakın.

Configuring your queue manager so that IBM MQ classes for JMS applications can connect in client mode  
To configure your queue manager so that IBM MQ classes for JMS applications can connect in client mode, you must create a server connection channel definition and start a listener.

## Sunucu bağlantısı kanal tanımlaması yaratılması

Tüm altyapılarda, bir sunucu bağlantı kanalı tanımlaması yaratmak için MQSC komutu DEFINE CHANNEL 'ı kullanabilirsiniz. Aşağıdaki örneğe bakın:

```
DEFINE CHANNEL(JAVA.CHANNEL) CHLTYPE(SVRCONN) TRPTYPE(TCP)
```

**IBM i** IBM i' ta, aşağıdaki örnekte olduğu gibi, CRTMQMCHL CL komutunu kullanabilirsiniz:

```
CRTMQMCHL CHLNAME(JAVA.CHANNEL) CHLTYPE(*SVRCN)  
TRPTYPE(*TCP)  
MQMNAME(QMGRNAME)
```

Bu komutta QMGRADı , kuyruk yöneticinizin adıdır.

**Linux** **Windows** Linux ve Windows üzerinde, IBM MQ Explorer kullanarak bir sunucu bağlantı kanalı tanımlaması da yaratabilirsiniz.

**z/OS** z/OS üzerinde, bir sunucu bağlantı kanalı tanımlaması yaratmak için işlemleri ve denetim panolarını kullanabilirsiniz.

Kanalın adı (JAVA.CHANNEL , uygulamanızın kuyruk yöneticisine bağlanmak için kullandığı bağlantı üreticisinin CHANNEL özelliği tarafından belirlenen kanal adıyla aynı olmalıdır. CHANNEL özelliğinin varsayılan değeri SYSTEM.DEF.SVRCONN.

## Dinleyici başlatma

Önceden başlatılmamışsa, kuyruk yöneticiniz için bir dinleyici başlamanız gerekir.

**Multi** Multiplatforms'ta, aşağıdaki örnekte gösterildiği gibi, MQSC komutu DEFINE LISTENER' ı kullanarak bir dinleyici nesnesi yarattıktan sonra bir dinleyiciyi başlatmak için MQSC komutu START DINLEYICISINI kullanabilirsiniz:

```
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) PORT(1414)  
START LISTENER(LISTENER.TCP)
```

**z/OS** z/OS' ta, aşağıdaki örnekte olduğu gibi yalnızca START LISTENER komutunu kullanıyorsunuz, ancak bir dinleyici başlatabilmeniz için kanal başlatıcı adres alanının başlatılması gerektiğini unutmayın:

```
START LISTENER TRPTYPE(TCP) PORT(1414)
```

**IBM i** IBM i' ta, aşağıdaki örnekte olduğu gibi, bir dinleyici başlatmak için STRMQMLSR CL komutunu da kullanabilirsiniz:

```
STRMQMLSR PORT(1414) MQMNAME(QMGRNAME)
```

Bu komutta QMGRADı , kuyruk yöneticinizin adıdır.

**ULW** On UNIX, Linux, and Windows, you can also use the control command **Runmq1sr** to start a listener, as in the following example:

```
runmq1sr -t tcp -p 1414 -m QMgrName
```

Bu komutta, *QMgrName* kuyruk yöneticinizin adıdır.

**Linux** **Windows** Linux ve Windows üzerinde, IBM MQ Explorer komutunu kullanarak bir dinleyici de başlatabilirsiniz.

**z/OS** z/OS işletim sistemi üzerinde, bir dinleyici başlatmak için işlemleri ve denetim panolarını da kullanabilirsiniz.

İletişiminizin dinlemede olduğu kapının numarası, uygulamanızın kuyruk yöneticisine bağlanmak için kullandığı bağlantı üreticisinin PORT (Kapı) özelliği tarafından belirlenen kapı numarasıyla aynı olmalıdır. PORT (Kapı) özelliğinin varsayılan değeri 1414 'tür.

### **IBM MQ classes for JMS için noktadan noktaya IVT**

IBM MQ classes for JMS ile 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.QUEUE ve daha sonra, iletiyi kuyruktan alır. Program, yürütme sırasında dinamik olarak gerektirdiği tüm nesnelere yaratılabilir ve yapılandırılabilir ya da bir dizin hizmetinden yönetilen nesnelere almak için JNDI 'yi kullanabilir.

Sinama bağımsız olduğundan ve bir dizin hizmetinin kullanılmasını gerektirmediğinden, önce JNDI kullanmadan kuruluş doğrulama sınamasını çalıştırın. Denetlenen nesnelere ilişkin açıklamalar için [Yönetim aracını kullanarak JMS nesnelere yapılandırma başlıklı konuya](#) bakın.

### **JNDI kullanmadan noktadan noktaya kuruluş doğrulama sınaması**

Bu testte, IVT programı yürütme sırasında dinamik olarak gerektirdiği tüm nesnelere yaratır ve yapılandırır ve JNDI kullanmaz.

IVT programını çalıştırmak için bir komut dosyası sağlanır. Komut dosyası, UNIX and Linux sistemlerinde IVTRun ve Windows sisteminde IVTRun.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin bin alt dizininde bulunur.

Sinamayı bağ tanımlama kipinde çalıştırmak için şu komutu girin:

```
IVTRun -nojndi [-m qmgr ] [-v providerVersion ] [-t]
```

Testi istemci kipinde çalıştırmak için önce kuyruk yöneticisini “Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102’de 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 csid ] [-t]
```

z/OS sistemlerinde eşdeğer bir komut dosyası sağlanmaz, ancak aşağıdaki komutu kullanarak Java sınıfını doğrudan çağırarak IVT 'yi bağ tanımlama kipinde çalıştırabilirsiniz:

```
java com.ibm.mq.jms.MQJMSIVT -nojndi [-m qmgr ] [-v providerVersion ] [-t]
```

Sınıf yolu com.ibm.mq.jms.jar içermelidir.

Komutlara ilişkin değiştirgeler aşağıdaki anlamlara sahiptir:

#### **-m qmgr**

IVT programının bağlandığı kuyruk yöneticisinin adı. Sinamayı bağ tanımlama kipinde çalıştırır ve bu değiştirgeyi atlarsanız, IVT programı varsayılan kuyruk yöneticisine bağlanır.

#### **-host anasistemadi**

Kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi.

#### **-port kapı**

Kuyruk yöneticisinin dinleyicisinin dinlediği kapının numarası. Varsayılan değer 1414'dir.

### **-channel kanal**

IVT programının kuyruk yöneticisine bağlanmak için kullandığı MQI kanalının adı. Varsayılan değer SYSTEM.DEF.SVRCONN'dir.

### **-v providerVersion**

IVT programının bağlanmayı beklediği kuyruk yöneticisinin yayın düzeyi.

Bu değiştirge, bir MQQueueConnectionFactory nesnesinin PROVIDERVERSION özelliğini ayarlamak için kullanılır ve geçerli değerler PROVIDERVERSION özelliğinkiyle 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 objects](#) içindeki PROVIDERVERSION özelliğinin açıklaması.

Varsayılan değer unspecified'dir.

### **-ccsid ccsid**

Bağlantı tarafından kullanılacak kodlanmış karakter takımının ya da kod sayfasının tanıtıcısı (CCSID). Varsayılan değer 819'dir.

### **-t**

İzleme etkinleştirildi. Varsayılan olarak izleme devre dışıdır.

Başarılı bir sınıma, 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:414d5120514d5f6d6277202020202001edb14620005e03
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 nesnelere 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 nesnelere 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 80. IBM MQ JMS yönetim aracının nasıl yapılandırılacağı hakkında bilgi için [JMS yönetim aracının yapılandırılması](#) başlıklı konuya bakın.

IVT programı, dizin hizmetinden bir MQQueueConnectionFactory nesnesi ve bir MQQueue nesnesini almak için JNDI 'yı kullanabilmelidir. Bu yönetilen nesnelere sizin için yaratmak üzere bir komut dosyası sağlanır. Komut dosyası, UNIX and Linux sistemlerinde IVTSetup ve Windows üzerinde IVTSetup.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin bin alt dizininde bulunur. Komut dosyasını çalıştırmak için şu komutu girin:

```
IVTSetup
```

Komut dosyası, yönetilen nesnelere 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 kullanmalısınız. IBM MQ Explorer JMS yönetim aracının nasıl kullanılacağına ilişkin bilgi için [Yönetim aracını kullanarak JMS nesnelere yapılandırma](#) başlıklı konuya bakın. IBM MQ Explorer' in nasıl kullanılacağına ilişkin bilgi için bkz. [IBM MQ Explorer 'a Giriş](#) ya da IBM MQ Explorer ile sağlanan yardım.

MQQueue nesnesi için ivtQ adıyla bağ tanımlandı ve SYSTEM.DEFAULT.LOCAL.QUEUE.

Denetlenen nesnelere 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 URL 'si. URL adresi 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 nesnelere silerek testten sonra toparlamak iyi bir uygulamadır. Bu amaçla bir komut dosyası sağlanır. Komut dosyası, Windows üzerinde UNIX and Linux sistemlerinde IVTTidy ve IVTTidy.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin bin alt dizininde bulunur.

## Noktadan noktaya kuruluş doğrulama sınaması 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 için ortam değişkenlerini ayarlama” sayfa 84](#) içinde 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 Clientolan bağlantı kipi.

### **hostname**

Kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi.

Bu ileti, IVT programının bağlanmaya çalıştığı kuyruk yöneticisinin kullanılmadığı 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 nesnelere saklamak için yapılandırılmamış ya da nesnelere ya da soneke ilişkin izinler doğru değil. Bu durumda daha fazla yardım için LDAP sunucusuna 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 Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102](#).



## **IBM MQ classes for JMS için IVT yayınlama/abone olma**

IBM MQ classes for JMS ile 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 nesnelere yaratabilir ve yapılandırabilir ya da bir dizin hizmetinden yönetilen nesnelere almak için JNDI 'yı kullanabilir.

Sinama bağımsız olduğundan ve bir dizin hizmetinin kullanılmasını gerektirmediğinden, önce JNDI kullanmadan kuruluş doğrulama sınavını çalıştırın. Denetlenen nesnelere ilişkin açıklamalar için [Yönetim aracını kullanarak JMS nesnelere yapılandırma](#) başlıklı konuya bakın.

## **JNDI kullanmadan yayınlama/abone olma kuruluş doğrulama sınavı**

Bu testte, IVT programı yürütme sırasında dinamik olarak gerektirdiği tüm nesnelere yaratır ve yapılandırır ve JNDI kullanmaz.

IVT programını çalıştırmak için bir komut dosyası sağlanır. Komut dosyası, UNIX and Linux sistemlerinde PSIVTRun ve Windows sisteminde PSIVTRun.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin bin alt dizininde bulunur.

Sinamayı bağ tanımlama kipinde çalıştırmak için şu komutu girin:

```
PSIVTRun -nojndi [-m qmgr ] [-bqm brokerQmgr ] [-v providerVersion ] [-t]
```

Sinamayı istemci kipinde çalıştırmak için, önce kuyruk yöneticisini "[Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması](#)" sayfa 1102 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 ccid ] [-t]
```

Komutlara ilişkin deęiřtirgeler ařaęıdaki anlamlara sahiptir:

### **-m qmgr**

IVT programının baęlandığı kuyruk yöneticisinin adı. Sinamayı bağ tanımlama kipinde çalıştırır ve bu deęiřtirgeyi atarsanız, IVT programı varsayılan kuyruk yöneticisine bağlanır.

### **-host anasistemadi**

Kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi.

### **-port kapı**

Kuyruk yöneticisinin dinleyicisinin dinlediği kapının numarası. Varsayılan deęer 1414 deęeridir.

### **-channel kanal**

IVT programının kuyruk yöneticisine bağlanmak için kullandığı MQI kanalının adı. Varsayılan deęer SYSTEM.DEF.SVRCONN deęeridir.

### **-bqm brokerQmgr**

Aracının çalıştığı kuyruk yöneticisinin adı. Varsayılan deęer, IVT programının baęlandığı kuyruk yöneticisinin adıdır.

Bu deęiřtirge, kuyruk yöneticisi sürüm numarası v /7 ya da üstü ile ilgili deęil.

### **-v providerVersion**

IVT programının bağlanmayı beklediği kuyruk yöneticisinin yayın düzeyi.

Bu deęiřtirge, bir MQTopicConnectionFactory nesnesinin PROVIDERVERSION özelliğini ayarlamak için kullanılır ve geçerli deęerler PROVIDERVERSION özelliğinkiyle aynıdır. Bu nedenle, geçerli deęerleri de içinde olmak üzere, bu parametreyle ilgili daha fazla bilgi için, [IBM MQ classes for JMS nesnelere yapılandırma](#) başlıklı konuya bakın.

Varsayılan deęer unspecified deęeridir.

### -ccsid ccsid

Bağlantı tarafından kullanılacak kodlanmış karakter takımının ya da kod sayfasının tanıtıcısı (CCSID). Varsayılan değer 819'dedir.

### -t

İzleme etkinleştirildi. Varsayılan olarak izleme devre dışıdır.

Başarılı bir sınıma, 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(tm) Message Service 7.0  
Publish/Subscribe Installation Verification Test
```

```
Creating a TopicConnectionFactory  
Creating a Connection  
Creating a Session  
Creating a Topic  
Creating a TopicPublisher  
Creating a TopicSubscriber  
Creating a TextMessage  
Adding text  
Publishing the message to topic://MQJMS/PSIVT/Information  
Waiting for a message to arrive [5 secs max]...
```

```
Got message:  
JMSMessage class: jms_text  
JMSType: null  
JMSDeliveryMode: 2  
JMSExpiration: 0  
JMSPriority: 4  
JMSMessageID: ID:414d5120514d5f6d6277202020202001edb14620006706  
JMSTimestamp: 1187182520203  
JMSCorrelationID: ID:414d5120514d5f6d6277202020202001edb14620006704  
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: 414D5143514D5F6D6277202020202001EDB14620006601  
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 nesnelere almak için JNDI 'y'ı 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 nesnelere 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 80](#). IBM MQ JMS yönetim aracının nasıl yapılandırılacağı hakkında bilgi için [JMS yönetim aracının yapılandırılması](#) başlıklı konuya bakın.

IVT programının, dizin hizmetinden bir MQTopicConnectionFactory nesnesi ve bir MQTopic nesnesini almak için JNDI 'y'ı kullanabilmesi gerekir. Bu yönetilen nesnelere sizin için yaratmak üzere bir

komut dosyası sağlanır. Komut dosyası, UNIX and Linux sistemlerinde IVTSetup ve Windows üzerinde IVTSetup.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin bin alt dizininde bulunur. Komut dosyasını çalıştırmak için şu komutu girin:

```
IVTSetup
```

Komut dosyası, yönetilen nesnelere 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 yönetim aracını ya da IBM MQ Gezgini 'ni kullanmanız gerekir. IBM MQ JMS yönetim aracının nasıl kullanılacağına ilişkin bilgi için [Yönetim aracını kullanarak JMS nesnelere yapılandırma](#) başlıklı 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 nesnelere 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 URL 'si. URL adresi 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 initCtxOlgü**

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 nesnelere silerek testten sonra toparlamak iyi bir uygulamadır. Bu amaçla bir komut dosyası sağlanır. Komut dosyası, Windows üzerinde UNIX and Linux sistemlerinde IVTTidy ve IVTTidy.bat olarak adlandırılır ve IBM MQ classes for JMS kuruluş dizininin bin alt dizininde 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 için ortam değişkenlerini ayarlama"](#) sayfa 84 içinde 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 Clientolan bağlantı kipi.

#### **hostname**

Kuyruk yöneticisinin çalıştığı sistemin anasistem adı ya da IP adresi.

Bu ileti, IVT programının bağlanmaya çalıştığı kuyruk yöneticisinin kullanılmadığı 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 nesnelere 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 Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102.](#)

### **IBM MQ classes for JMS örnek uygulamalarının kullanılması**






IBM MQ classes for JMS örnek uygulamaları, JMS API ' nın ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluşunuzu ve ileti sistemi sunucunuzu ayarlamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

### **Bu görev hakkında**

Kendi uygulamalarınızı yaratmak için yardıma gerek 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 (ConnectionFactory, Connection, Session, Destination, Destination, bir üretici ya da bir tüketicisi ya da her ikisi) yaratmak ve uygulamanızın nasıl çalışmasını istediğinizi belirtmek için gerekli olan belirli özellikleri yaratmak için gereken temel adımları tanımlayın. Daha fazla bilgi için bkz [“IBM MQ classes for JMS uygulamaları yazılıyor” sayfa 125.](#) Örnekler, IBM MQ' un gelecekteki yayınlarında değişikliğe tabi olabilir.

[Çizelge 10 sayfa 109](#) , her altyapıda IBM MQ classes for JMS örnek uygulamalarının kurulu olduğu yeri gösterir:

Çizelge 10. IBM MQ classes for JMS örnek uygulamaları için kuruluş dizinleri

Platform	Dizin
 UNIX  Linux	MQ_INSTALLATION_PATH/samp/jms/samples
 Windows	MQ_INSTALLATION_PATH\tools\jms\samples
 IBM i	/qibm/proddata/mqm/java/samples/jms/samples
 z/OS	MQ_INSTALLATION_PATH/java/samples/jms

Bu dizin içinde, Çizelge 11 sayfa 109’ünde gösterildiği gibi bir ya da daha fazla örnek uygulama içeren alt dizinler vardır.

Çizelge 11. IBM MQ classes for JMS örnek uygulamalar






Örneklerin adı	Tanım
JmsBrowser.java	Adı belirtilen kuyrukta bulunan tüm iletileri, kaldırmadan, bir tüketici uygulaması tarafından alınacakları sırayla içeren bir JMS kuyruk tarayıcısı uygulaması.
JmsConsumer.java	Adı belirtilen kuyrukta bulunan, bunları kaldırmadan, bir tüketici uygulaması tarafından alınacakları sırayla, bağlantı üreticisi eşgörünümünün ve hedef yönetim ortamının ilk bağlamındaki hedef somut örneğe bakılarak alınacağı bir JMS kuyruk tarayıcısı uygulaması (yalnızca bu örnek, yalnızca dosya sistemi bağlamını destekler).
JmsJndiConsumer.java	Bağlantı üreticisi eşgörünümünün ve hedef yönetim ortamının bir başlangıç bağlamındaki hedef somut örneğine bakarak bir iletiyi (kuyruk ya da konu) alan bir JMS tüketici (alıcı ya da abone) uygulaması (yalnızca bu örnek dosya sistemi bağlamını destekler).
JmsJndiProducer.java	Bağlantı üreticisi eşgörünümünün ve hedef yönetim ortamının ilk bağlamdaki hedef somut örneğine bakarak, adlandırılan hedefe (kuyruğa ya da konuya) yalın bir ileti gönderen bir JMS üreticisi (gönderen ya da yayınlayıcı) uygulaması (yalnızca bu örnek dosya sistemi bağlamını destekler).
JmsProducer.java	Adı belirtilen hedefe (kuyruk ya da konu) yalın bir ileti gönderen bir JMS üreticisi (gönderen ya da yayınlayıcı) uygulaması.
<b>/interactive/</b>	
SampleConsumerJava.java	Bir topici/kuyruktan ileti alın.
SampleProducerJava.java	İleti (ler) i bir topici/kuyruğa gönder.
<b>/interactive/helper/</b>	
BaseOptions.java	Kullanıcı seçenek (ler) işlevini sağlamak için genişletilebilen bir soyut sınıf.
IsValidType.java	Geçerlilik denetleyici sınıfları için soyut sınıf.
JmsApp.java	Tüketici/üretici işlevlerini sağlamak için genişletilebilir bir soyut sınıf.

Çizelge 11. IBM MQ classes for JMS örnek uygulamalar (devamı var)

Örneklerin adı	Tanım
Keys.java	Örnek uygulamalara ilişkin seçenekleri tanımlayan bir anahtar kümesi.
Literals.java	Sabit hazır bilgiler kümesi.
MyContext.java	Seçeneklerin sunulmakta olduğu bağlam.
Options.java	Kullanıcı seçenekleri için işlevsellik sağlar.
OptionsPresenter.java	Geçerli seçeneklerin sunulmakta olduğu bağlam.
<b>/simple/</b>	
SimpleAsyncPutPTP.java	Noktadan noktaya ileti alışverişi için basit bir uygulama; ileti zamanuyumsuz olarak gönderilir ( <i>yangın-ve-unut</i> ileti sistemi olarak da bilinir). Hiçbir ileti alınmıyor.
SimpleDurableSub.java	Dayanıklı abonelik olanağını sergileyen basit bir uygulama.
SimpleJNDILookup.java	İlk bağlamı kullanarak JMS nesnelerinin aranmasını gösteren, minimal ve basit bir uygulama. Kuyruk yöneticisiyle bağlantı yapılmadı ve hiçbir ileti gönderilmedi ya da gönderilmedi.
SimpleMQMDDRead.java	Bir JMS uygulamasının MQ Message Descriptor (MQMD) alanlarını JMS ileti özellikleri olarak nasıl kullanılabilir kıldığını gösteren yalın bir uygulama. İleti gönderilmez; kullanılan kuyruğun bazı iletilerle doldurulduğunu varsaymaktadır.
SimpleMQMDWrite.java	Bir JMS uygulamasının MQ Message Descriptor (MQMD) alanlarını nasıl yazabileceğini gösteren basit bir uygulama. Hiçbir ileti alınmıyor.
SimplePTP.java	Noktadan noktaya ileti sistemi için en düşük düzeyde ve basit bir uygulama.
SimplePubSub.java	Yayınlama-abone olma ileti alışverişi için basit ve basit bir uygulama.
SimpleReadAheadPTP.java	Noktadan noktaya ileti alışverişi için basit bir uygulama; iletiler kuyruk yöneticisinden (ileriye doğru okuma olanağı olarak da bilinir) akıtılır. İleti gönderilmez; kullanılan kuyruğun bazı iletilerle doldurulduğunu varsaymaktadır.
SimpleRequestor.java	İstek iletisi göndermek için istekte bulunan ve sonra bekleyen ve alacak olan basit bir uygulama yanıt. Not: Başka bir uygulamanın istek iletisini işleyecek ve yanıt iletisini göndereceği varsayılır.
SimpleResponder.java	Bir ileti için hedef alan ve daha sonra iletinin replyTo hedefine bir yanıt gönderen basit bir uygulama. Uygulama, SimpleRequestor örneğiyle birlikte çalışmak üzere yazılır.
SimpleRetainedPub.java	Alıkonan bir yayını gösteren basit bir uygulama. Hiçbir ileti alınmıyor.
SimpleWMQJMSPTP.java	Noktadan noktaya ileti sistemi için en düşük düzeyde ve basit bir uygulama.
SimpleWMQJMSPubSub.java	Yayınlama/abone olma ileti alışverişi için basit ve basit bir uygulama.

IBM MQ classes for JMS , örnek uygulamaları çalıştırmak için kullanılacak runjms adlı bir komut kütüğü sağlar. This script sets up the IBM MQ environment to allow you to run the IBM MQ classes for JMS sample applications.

Çizelge 12 sayfa 111 , her altyapıda komut dosyasının yerini gösterir:

Çizelge 12. runjms komut dosyasının konumu	
Platform	Dizin
 UNIX  Linux	MQ_INSTALLATION_PATH/java/bin/runjms
 Windows	MQ_INSTALLATION_PATH\java\bin\runjms.bat
 IBM i	/qibm/proddata/mqm/java/bin/runjms ya da /qibm/proddata/mqm/java/bin/runjms64
 z/OS	MQ_INSTALLATION_PATHjava/bin/runjms

Örnek bir uygulamayı çağırmak için runjms komut dosyasını kullanmak için aşağıdaki adımları tamamlayın:

## 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 listesini görüntüler.

3. Örneği bu deęiřtirgelerle çalıştırmak için aşağıdaki komutu girin:

```
Path to the runjms script/runjms sample_application_name parameters
```

## Örnek

 Örneğin, Linux üzerinde 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 81

IBM MQ classes for JMS' u kurduğunuzda, bir dizi dosya ve izin oluşturulur. Windows' ta, bazı yapılandırma otomatik olarak ortam deęiřkenleri ayarlanarak kuruluř sırasında gerekleřtirilir. On other platforms, and in certain Windows environments, you must set environment variables before you can run IBM MQ classes for JMS applications.

## IBM MQ classes for JMS ile saęlanan komut dosyaları

IBM MQ classes for JMS kullanıldığında gerekleřtirilmesi gereken genel görevlere yardımcı olması için bir dizi komut dosyası saęlanmıřtır.

Çizelge 13 sayfa 112 , tüm komut dosyalarını ve bunların kullanımlarını listeler. Komut dosyaları, IBM MQ classes for JMS kuruluř dizininin bin alt dizininde bulunur.

Çizelge 13. IBM MQ classes for JMS ile sağlanan komut dosyaları	
Utility	Kullan
Temizleme <sup>1</sup>	Bu komut dosyası önceki yayınlarla uyumluluk için korunur, ancak hiçbir işlev gerçekleştirmez. Abonelik bilgilerini el ile temizleme işlemi artık gerekli değil
DefaultConfiguration	Varsayılan yapılandırma uygulamasını Windows dışındaki altyapılarda çalıştırır.
formatLog <sup>1</sup>	Bu komut dosyası önceki yayınlarla uyumluluk için korunur, ancak hiçbir işlev gerçekleştirmez. Günlük çıkışı artık okunabilir metinde üretilir.
IVTRun <sup>1</sup> IVTSetup <sup>1</sup> VTTiry <sup>1</sup>	Noktadan noktaya kuruluş doğrulama sınavında ( "IBM MQ classes for JMS için noktadan noktaya IVT" sayfa 101) açıklandığı gibi kullanılır.
JMSAdmin <sup>1</sup>	Runs the IBM MQ JMS administration tool, as described in <a href="#">Denetim aracı başlatılıyor</a> .
JMSAdmin.config	The configuration file for the IBM MQ JMS administration tool, as described in <a href="#">JMS yönetim aracının yapılandırılması</a> .
PSIVTRun <sup>1</sup>	Runs the publish/subscribe installation verification test program, as described in <a href="#">"IBM MQ classes for JMS için IVT yayınlama/abone olma" sayfa 105</a> .
PSReportDump.class	Bu sınıf önceki yayın düzeyleriyle uyumluluk için korunur, ancak hiçbir işlev gerçekleştirmez.
setjmsenv	Sets the environment variables for running an IBM MQ classes for JMS application in a 32-bit Java virtual machine (JVM) on UNIX and Linux systems, as described in <a href="#">"IBM MQ classes for JMS için ortam değişkenlerini ayarlama" sayfa 84</a> .
setjmsenv64	Sets the environment variables for running an IBM MQ classes for JMS application in a 64-bit JVM on UNIX and Linux systems, as described in <a href="#">"IBM MQ classes for JMS için ortam değişkenlerini ayarlama" sayfa 84</a> .

**Not:**

1. Windows üzerinde, dosya adı .bat uzantısına sahiptir.

**OSGi desteği**

OSGi, uygulamaların kod paketleri olarak konuşlandırılmasını destekleyen bir çerçeve sağlar. Dokuz OSGi kod paketi IBM MQ classes for JMS' nin bir parçası olarak sağlanır.

OSGi, kod paketleri biçiminde gelen uygulamaların konuşlandırılmasını destekleyen genel amaçlı, güvenli ve yönetilen Java çerçevesini sağlar. OSGi uyumlu aygıtlar, paketleri yükleyebilir ve kurabilir ve artık gerekli olmadığında bunları kaldırabilir. Çerçeve, kod paketlerinin dinamik ve ölçeklenebilir bir şekilde kurulup güncellenmesini yönetir.

IBM MQ classes for JMS , aşağıdaki OSGi paketlerini içerir.

**com.ibm.msg.client.osgi.jmsversion\_number.jar**

IBM MQ classes for JMS içindeki ortak kod katmanı. JMS için IBM MQ sınıflarının katmanlı mimarisine ilişkin bilgi için bkz. [JMS mimarisi için IBM MQ sınıfları](#).

**com.ibm.msg.client.osgi.jms.prereq\_version\_number.jar**

Ortak katmana ilişkin önkoşul 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 katmana ilişkin iletiler.

**com.ibm.msg.client.osgi.wmq\_version\_number.jar**

IBM MQ classes for JMS içindeki IBM MQ ileti alışverişi sağlayıcısı. For information about the layered architecture of IBM MQ classes for JMS, see [JMS mimarisi için IBM MQ sınıfları](#).

**com.ibm.msg.client.osgi.wmq.prereq\_version\_number.jar**

IBM MQ ileti alışverişi sağlayıcısına ilişkin önkoşul JAR dosyaları.

**com.ibm.msg.client.osgi.wmq.nls\_version\_number.jar**

IBM MQ ileti alışverişi sağlayıcısına ilişkin iletiler.

**com.ibm.mq.osgi.allclient\_version\_number.jar**

Bu JAR dosyası, uygulamaların hem IBM MQ classes for JMS hem de IBM MQ classes for Java' yi kullanmasını ve PCF iletilerini işleme kodunu da kapsamalarını sağlar.

**com.ibm.mq.osgi.allclientprereqs\_version\_number.jar**

This JAR file provides the prerequisites for `com.ibm.mq.osgi.allclient_version_number.jar` where `version_number` is the version number of IBM MQ that is installed.

Bu paketler, IBM MQ kurulumunuzun `java/lib/OSGi` alt dizinine ya da Windows üzerindeki `java\lib\OSGi` klasörüne kurulur.

From IBM MQ 8.0, use the bundles `com.ibm.mq.osgi.allclient_8.0.0.0.jar`, and `com.ibm.mq.osgi.allclientprereqs_8.0.0.0.jar` for any new applications. Bu kod paketlerinin kullanılması, hem IBM MQ classes for JMS hem de IBM MQ classes for Java ' yi aynı OSGi çerçevesi içinde çalıştırılmama kısıtlamasını kaldırır, ancak diğer tüm kısıtlamalar yine de geçerli olur. Ürünün IBM MQ 8.0 tarihinden önceki sürümleri için, IBM MQ classes for JMS ya da IBM MQ classes for Java kullanılmasının bu kısıtlaması uygulanır.

IBM MQ kurulumunuzun `java/lib/OSGi` alt dizinine ya da Windows üzerindeki `java\lib\OSGi` klasörüne de kurulu olan `com.ibm.mq.osgi.javaversion_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. Bir OSGi Yayın Düzeyi 3 ortamında çalışmıyorlar.

OSGi yürütme ortamı gereken DLL kütüklerini ya da paylaşılan kitaplıkları bulabilmesi için, sistem yolunu ya da kitaplık yolunu doğru olarak ayarlamalısınız.

If you use the OSGi bundles for the IBM MQ classes for JMS, temporary topics do not work. Ayrıca, OSGi gibi birden çok sınıf yükleyici ortamındaki sınıfların yüklenmesi sırasında sorunlu bir sorun nedeniyle, Java ' ta yazılan kanal çıkış sınıfları desteklenmez. Bir kullanıcı paketi IBM MQ classes for JMS kod paketlerinden haberdar olabilir, ancak IBM MQ classes for JMS kod paketleri herhangi bir kullanıcı kod paketinin farkında değildir. Sonuç olarak, bir IBM MQ classes for JMS kod paketinde kullanılan sınıf yükleyici, bir kullanıcı kod paketindeki bir kanal çıkış sınıfını yükleyemez.

OSGi hakkında daha fazla bilgi için [OSGi Alliance web sitesine](#) bakın.

## **z/OS V 9.1.0 MQ Adv. VUE JMS üzerinde çalışan toplu iş uygulamalarına istemci bağlantılığı z/OS**

Bir istemci bağlantısı kullanarak z/OS üzerindeki IBM MQ classes for JMS uygulaması, **ADVCAP(ENABLED)** özneliği olan z/OS üzerindeki bir kuyruk yöneticisine bağlanabilir. İstemci bağlantısının kullanılması IBM MQ topolojilerini basitleştirebilir.

**ADVCAP(ENABLED)** değeri yalnızca, IBM MQ Advanced for z/OS Value Unit Edition (bkz. [IBM MQ ürün tanıtıcıları ve dışa aktarma bilgileri](#)) olarak lisanslanan ve **QMGRPROD** ayarı ZAVANTAJLI olan z/OS kuyruk yöneticisi için geçerlidir.

**QMGRPROD** ile ilgili daha fazla bilgi için bkz. [DISPLAY QMGR ADVCAP](#) ve [START QMGR](#) .

Toplu işin desteklenen tek ortam olduğunu unutmayın; JMS for CICS ya da JMS for IMS desteği yoktur.

z/OS üzerindeki bir IBM MQ classes for JMS uygulaması, z/OS üzerinde çalışmayan bir kuyruk yöneticisine ya da **ADVCAP** (ENABLED) seçeneği ayarlanmamış bir kuyruk yöneticisine bağlanmak için istemci kipi bağlantısını kullanamaz.

z/OS üzerinde bir IBM MQ classes for JMS uygulaması istemci kipini kullanarak bağlanmayı denerse ve buna izin verilmezse, JMSFMQ0005 kural dışı durum iletisi yayınlanır.

## Advanced Message Security (AMS) desteği

IBM MQ 9.1' den IBM MQ classes for JMS istemci uygulamaları, uzak z/OS sistemlerinde IBM MQ Advanced for z/OS Value Unit Edition kuyruk yöneticilerine bağlanırken AMS komutunu kullanabilir.

jcceracfsyeni anahtar deposu tipi yalnızca z/OS üzerinde keystore.conf içinde desteklenir; burada:

- Özellik adı önceki jcceracfs ve bu ad önceki büyük ve küçük harfe duyarlı değildir.
- Anahtar deposu bir RACF anahtarlığı.
- Parolalar gerekli değildir ve yoksayılr. Bunun nedeni, RACF anahtarlarının parola kullanmaması olabilir.
- Sağlayıcıyı belirtirseniz, sağlayıcı IBMJCEolmalıdır.

jcceracfs komutunu AMS ile 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:

```
jcceracfs.keystore=safkeyring://JOHNDOE/drq.ams.keyring
```

## IBM MQ classes for JMS ' in ayrı ayrı alınması

The IBM MQ classes for JMS are available within a self-extracting JAR file that you can download from Fix Central if you want to obtain just the IBM MQ classes for JMS JAR files, for deployment into a software management tool, or to use with stand-alone client applications.

## Başlamadan önce

Bu görevi başlatmadan önce, makineniz üzerinde kurulu bir Java runtime environment (JRE) kurulu olduğundan ve JRE ' nin sistem yoluna eklendiğinden emin olun.

Bu kuruluş işleminde kullanılan Java kuruluş programı, kök (root) ya da belirli bir kullanıcı olarak çalışmamaya gerek yoktur. Tek gereksinim, çalıştırıldığı kullanıcının, dosyaların içeri girmesini istediğiniz dizine yazma erişimi olduğu gibi çalıştırılır.

## Bu görev hakkında

IBM MQ 8.0 öncesinde, IBM WebSphere MQ classes for Java ya da IBM WebSphere MQ classes for JMS ayrı bir yükleme olarak kullanılamaz. For IBM WebSphere MQ 7.5 or earlier, if you are developing and running Java language applications that use either the IBM WebSphere MQ classes for Java or IBM WebSphere MQ classes for JMS, you need to install them either by performing a full server installation or by installing one of the client SupportPacs onto the system where the application is being developed and the system where the application will run. Bu kurulum, IBM WebSphere MQ classes for Java ve IBM WebSphere MQ classes for JMS dosyalarından daha fazla dosya kurar.

Ancak, IBM MQ 8.0' den aşağıdaki dosyalar, otomatik olarak çıkarılacak bir JAR dosyası içinde bulunur; bu dosya, aşağı yükleme ve kuruluş boyutunu ve kuruluşu gerçekleştirmek için gereken süreyi en aza indirir:

- The IBM MQ classes for JMS
- The IBM MQ classes for Java
- IBM MQ kaynak bağıdaştırıcısı
- IBM MQ OSGi kod paketleri

**V 9.1.0.8** From IBM MQ 9.1.0 Fix Pack 8, the JMSAdmin tool is installed as part of the self-extracting JAR file, which contains the following extra files that are related to the JMSAdmin tool:

- Windows üzerinde JMSAdmin aracını başlatmak için kullanılan JMSAdmin.bat dosyası.
- Aracı Linux ve UNIX platformlarında başlatmak için kullanılan JMSAdmin komut dosyası.
- JMSAdmin aracına ilişkin örnek yapılandırma dosyası (JMSAdmin.config).

**V 9.1.0.8** Kendi kendine ayıklayan JAR dosyası kullanılarak kurulan bir istemci, bir dosya sistemi bağlamında (.bindings dosyası) JMS denetimli nesnelere oluşturmak için JMSAdmin aracını kullanabilir. İstemci aynı zamanda bu yönetilen nesnelere de kullanabilir ve kullanabilir.

**V 9.1.0.8** Daha önce, kendi kendine çıkarma JAR dosyası com.ibm.mq.allclient.jar dosyasını ve tüm önkoşul JAR dosyalarını belirtilen kuruluş dizini içindeki wmq/JavaSE dizinine kurdu.\* IBM MQ 9.1.0 Fix Pack 8' tan bu dosyalar, wmq/JavaSE/lib dizinine kurulur ve JMSAdmin aracı ile ilgili dosyalar wmq/JavaSE/bin dizinine kurulur.

Yürütülebilir JAR dosyasını çalıştırdığınızda, bu dosya, kabul edilmesi gereken IBM MQ lisans sözleşmesini görüntüler. It asks for a directory in which to install the IBM MQ classes for Java, IBM MQ classes for JMS, the resource adapter, and OSGi bundles. Seçilen kuruluş dizini yoksa, yaratılır ve program kütükleri kurulur. Ancak, izin varsa, bir hata bildirilir ve hiçbir dosya kurulmaz.

## Yordam

1. Download the IBM MQ Java / JMS client JAR file from Fix Central.

a) Bu bağlantıyı tıklatın: [IBM MQ Java / JMS Client](#).

b) Görüntülenen kullanılabilir düzeltmeler listesinde, IBM MQ sürümünüze ilişkin istemciyi bulun.

Örneğin:

```
release level: 9.1.4.0-IBM-MQ-Install-Java-All
Continuous Delivery Release:9.1.4 IBM MQ JMS and Java 'All Client'
```

Daha sonra istemci dosyası adını tıklatın ve karşıdan yükleme işlemini izleyin.

2. Kuruluşu, dosyayı karşıdan yüklediğiniz dizinden başlatın.

Kuruluşu başlatmak için, aşağıdaki biçimi kullanarak bir komut girin:

```
java -jar V.R.M.F-IBM-MQ-Install-Java-All.jar
```

Burada V.R.M.F , ürün sürüm numarasıdır; örneğin, 9.1.4.0 ve V.R.M.F-IBM-MQ-Install-Java-All.jar , Fix Central' dan aşağı yüklenen dosyanın adıdır.

Örneğin, IBM MQ 9.1.4 yayın düzeyi için Java / JMS istemcisini kurmak için aşağıdaki komutu kullanırdınız:

```
java -jar 9.1.4.0-IBM-MQ-Install-Java-All.jar
```

**Not:** Bu kuruluşu gerçekleştirmek için, makineniz üzerinde kurulu bir JRE kurulu olmalıdır ve sistem yoluna eklenmelidir.

Komutu girdiğinizde, aşağıdaki bilgiler görüntülenir:

```
IBM MQ V9.1' i kullanmadan, çıkarmadan ya da kurmadan önce kabul etmeniz gerekir
1. terimler. IBM International License Agreement for Evaluation of
Programlar 2. IBM Uluslararası Program Lisans Sözleşmesi ve ek
lisans bilgileri. Lütfen aşağıdaki lisans sözleşmelerini dikkatli bir şekilde okuyun.
```

Lisans sözleşmesi ayrı ayrı görüntülenebilir  
--viewLicenseSözleşme seçeneği.

Şimdi lisans koşullarını görüntülemek için Enter tuşuna ya da atlamak için 'x' tuşuna basın.

### 3. Lisans koşullarını gözden geçirin ve kabul edin:

#### a) Lisansı görüntülemek için Enter tuşuna basın.

Diğer bir seçenek olarak, x tuşuna basılması lisansın görüntülenmesini sağlar.

Lisans görüntüledikten sonra ya da x seçeneğini işaretlerseniz, aşağıdaki ileti görüntülenir:

```
Ek lisans bilgileri ayrı ayrı görüntülenebilir  
--viewLicenseBilgi seçeneği.
```

Ek lisans bilgilerini şimdi görüntülemek için Enter tuşuna ya da atlamak için 'x' tuşuna basın.

#### b) Ek lisans koşullarını görüntülemek için Enter tuşuna basın.

Diğer bir seçenek olarak, x tuşuna basılması ek lisans koşullarının görüntülenmesini sağlar.

Ek lisans koşulları görüntüledikten sonra ya da x seçeneğini işaretlerseniz hemen aşağıdaki ileti görüntülenir:

Aşağıdaki "I Agree" (Kabul ediyorum) seçeneğini belirleyerek, lisans sözleşmesi ve IBM dışı koşullar (varsa). Eğer yapmazsanız Katılıyorum, "Kabul etmiyorum" seçeneğini belirleyin.

[ 1] Kabul ediyorum seçeneğini belirleyin ya da [ 2] Kabul etmiyorum:

#### c) Lisans sözleşmesini kabul etmek ve kuruluş dizinini seçmeye devam etmek için 1 'i seçin.

Diğer bir seçenek olarak, 2 seçeneğinin belirlenmesi hemen sona erdirilsin.

1 değerini seçerseniz, aşağıdaki ileti görüntülenir:

Ürün dosyaları için izin girin ya da varsayılan değeri kabul etmek için boş bırakın.  
Varsayılan hedef izin H: \WMQ

Ürün dosyaları için hedef izin?

### 4. Java / JMS istemcisine ilişkin kuruluş dizinini belirtin:

- Ürün dosyalarını varsayılan konuma kurmak istiyorsanız, bir değer belirtmeden Enter tuşuna basın.
- Ürün dosyalarını varsayılan değer olarak farklı bir konuma kurmak istiyorsanız, ürün dosyalarını kurmak istediğiniz dizinin adını belirtin ve kuruluşu başlatmak için Enter tuşuna basın.

Belirttiğiniz izin adı önceden var olmamalı, tersi durumda, kuruluşu başlattığınızda bir hata bildirilir ve hiçbir dosya kurulmaz.

Önceden var olmadığı sürece, seçilen kuruluş dizini yaratılır ve program dosyaları bu dizine kurulur. Kuruluş sırasında, seçtiğiniz kuruluş dizini içinde, adı wmq olan yeni bir izin yaratılır. Üç alt izin, JavaEE, JavaSE ve OSGi, wmq dizininde aşağıdaki içerikle oluşturulur:

```
.\JavaEE:  
wmq.jmsra.ivt.ear  
wmq.jmsra.rar  
  
.\JavaSE:  
com.ibm.mq.allclient.jar  
com.ibm.mq.traceControl.jar  
fscontext.jar  
jms.jar  
providerutil.jar  
  
.\OSGi:  
com.ibm.mq.osgi.allclient_V.R.M.F.jar  
com.ibm.mq.osgi.allclientprereqs_V.R.M.F.jar
```

Burada V.R.M.F, Sürüm, Yayın, Değişiklik ve Düzeltme Paketi numarasıdır.

Kuruluş işlemi tamamlandığında, aşağıdaki örnekte gösterildiği gibi bir onay iletisi görüntülenir:

```
Dosyalar H ' ye çıkarılıyor: \WMQ \wmq  
Tüm ürün dosyaları başarıyla çıkarıldı.
```

## IBM MQ classes for JMS içinde ödenek listesi

Java object serialization and deserialization mechanism has been identified as a potential security risk. IBM MQ classes for JMS içindeki allowlistleme, bazı diziselleştirme risklerine karşı koruma sağlar.

The Java object serialization and deserialization mechanism has been identified as a potential security risk because deserialization instantiates arbitrary Java objects, where there is the potential for maliciously sent data to cause various problems. Serileştirmenin dikkate değer bir uygulaması, rasgele nesnelere sarmak ve aktarmak için serileştirme kullanan Java Message Service (JMS) ObjectMessages biçimlerinde yer alıyor.

Serileştirme allowing, serileştirme pozları oluşturan bazı risklere karşı, olası bir hafifletme biçimidir. By explicitly specifying which classes can be encapsulated in, and extracted from, ObjectMessages, allowlisting provides some protection against some serialization risks.

## IBM MQ classes for JMS içinde ödenek listesi

Bakınız:

- Allowlisteye genel bakış için [“Allowlisting kavramları” sayfa 117](#)
- Allowlist nasıl ayarlanmanıza ilişkin bilgi için [“JMS allowlist uygulamasını kurma ve kullanma” sayfa 120](#)
- WebSphere Application Server' ta bir allowlist nasıl ayarladığınız hakkında bilgi için [“WebSphere Application Server içinde ödenek listesi” sayfa 122](#) .

### İlgili kavramlar

[“Running IBM MQ classes for JMS applications under the Java security manager” sayfa 96](#)

IBM MQ classes for JMS , Java güvenlik yöneticisi etkinleştirilmiş olarak çalışabilir. Uygulamaları Java security manager etkin bir şekilde başarıyla çalıştırmak için, Java virtual machine (JVM) olanağını uygun bir ilke yapılandırma dosyasıyla yapılandırmalısınız.

### Allowlisting kavramları

IBM MQ classes for JMS' ta, JMS ObjectMessage arabiriminin uygulanmasında sınıfların ödenek listesi desteği, Java nesne serileştirme ve dizisel biçimden geri çevirme düzeneği ile ilgili olabilecek bazı güvenlik risklerine karşı bir hafifletme sağlar.

## IBM MQ classes for JMS içinde ödenek listesi

### Önemli:

Mümkün olduğunda, *izin listesi* terimi *beyaz listelerinin* yerini almıştır. IBM MQ 9.0 ve sonraki yayınlarda, bu konuda (**com.ibm.mq.jms.\***) belirtilen 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 ObjectMessage arabiriminin uygulanmasında sınıfların geri listesini destekler.

Allowlist, ObjectMessage.setObject() ile hangi Java sınıflarının diziselleştirilebileceğini ve ObjectMessage.getObject() ile dizisel biçimden geri çevrilebilir olduğunu tanımlar.

Attempts to serialize or deserialize an instance of a class not included in the allowlist with ObjectMessage cause a javax.jms.MessageFormatException to be thrown, with a java.io.InvalidClassException as its cause.

### Allowlist oluşturma

**Önemli:** IBM MQ classes for JMS , allowlist ile dağıtılamaz. ObjectMessages kullanılarak aktarılacak sınıfların seçimi bir uygulama tasarım seçimidir ve IBM MQ bunu önlemez.

Bu nedenle, allowlisting mekanizması, iki işlem kipinin yapılmasına olanak sağlar:

## Keşif

In this mode, the mechanism produces a listing of fully qualified class names, reporting all classes that have been observed to be serialized or deserialized in ObjectMessages.

## Zorlama

Bu kipte, mekanizma, allowlist 'te olmayan sınıfları diziselleştirme ya da dizisel biçimden geri çevirme girişimlerini reddederek, allowlist (allowing) listesini uygular.

Bu mekanizmayı kullanmak istiyorsanız, başlangıçta diziselleştirilmiş ve dizisel biçimden geri çevrilen sınıfların listesini toplamak, listeyi gözden geçirmek ve allowlisteniz için bunu temel olarak kullanmak için DISPLAY (DISCOVERY) kipinde çalışmalısınız. Listenin değiştirilmeden kullanılması uygun olabilir, ancak bu işlemi yapmaya karar vermeden önce listenin ilk önce gözden geçirilmesi gerekir.

## Allowlisteleme mekanizmasını denetleme

Allowlisteleme mekanizmasını denetlemek için kullanılacak üç sistem özelliği vardır:

### com.ibm.mq.jms.allowlist

Bu özellik aşağıdaki yollardan biriyle belirtilebilir:

- Allowlist (allowlist) içeren dosyanın yol adı, dosya URI biçiminde ( `file:` ile başlayarak). DISABLE kipinde bu dosya, allowlicting mekanizmasının içine yazılır. Dosya var olmamalıdır. Dosya varsa, mekanizma bu dosyanın üzerine yazmak yerine bir kural dışı durum yayınlar. UYGULAMA kipinde bu dosya, allowlisteleme mekanizması tarafından okunurdu.
- Allowlist (allowlist) oluşturan tam olarak nitelenmiş sınıf adlarından oluşan virgülle ayrılmış.

Bu özellik ayarlanmamış ise, allowlist mekanizması etkin değildir.

Bir Java security manager kullanıyorsanız, IBM MQ classes for JMS JAR dosyalarının bu dosyaya okuma ve yazma erişimine sahip olduğundan emin olmalısınız.

### com.ibm.mq.jms.allowlist.discover

- Bu özellik ayarlanmazsa ya da false değerine ayarlanırsa, allowlist düzeneği UYGULAMA kipinde çalışır.
- Bu özellik true (doğru) olarak ayarlanırsa ve allowlist bir dosya URI 'si olarak belirtilmişse, allowlist mekanizması DISABLE kipi içinde çalışır.
- Bu özellik doğru olarak ayarlanırsa ve allowlist, sınıf adları listesi olarak belirtilmişse, allowlist mekanizması uygun bir kural dışı durum oluşturur.
- Bu özellik true olarak ayarlanırsa ve allowlist `com.ibm.mq.jms.allowlist` özelliği kullanılarak belirtilmediyse, allowlist mekanizması etkin değildir.
- Bu özellik true (doğru) olarak ayarlanırsa ve allowlist dosyası önceden varsa, allowlist düzeneği bir `java.io.InvalidClassException` yayınlar ve bu girişlere dosya eklenmez.

### com.ibm.mq.jms.allowlist.mode

Bu dizgi özelliği şu üç yöntemden herhangi birinde belirtilebilir:

- Bu özellik SERIALIZE olarak ayarlandıysa, UYGULAMA kipi yalnızca `ObjectMessage.setObject()` yönteminde allowlist geçerlilik denetimini gerçekleştirir.
- Bu özellik DESERIALIZE olarak ayarlandıysa, UYGULAMA kipi yalnızca `ObjectMessage.getObject()` yönteminde allowlist geçerlilik denetimini gerçekleştirir.
- Bu özellik ayarlanmazsa ya da başka bir değere ayarlanmışsa, UYGULAMA kipi hem `ObjectMessage.getObject()` hem de `ObjectMessage.setObject()` yöntemlerinde allowlist geçerlilik denetimi gerçekleştirir.

## Allowlist dosyasının biçimi

Bunlar, allowlist dosyasının biçiminin ana özellikleridir:

- Allowlist dosyası, platforma uygun satır sonlarıyla birlikte varsayılan platform dosyası kodlamasında yer alıyor.

**Not:** Bir allowlist dosyası kullanılıyorsa, o dosya her zaman yazılır ve JVM için varsayılan dosya kodlamasını kullanarak okunur.

Allowlist dosyası aşağıdaki yöntemlerden biriyle üretilirse, bu sorun olmaz:

- **z/OS** z/OS üzerinde çalışan ve z/OS üzerinde çalışmakta olan diğer bağımsız uygulamalar tarafından kullanılan bağımsız bir uygulama tarafından üretilir.
- Herhangi bir altyapıda WebSphere Application Server içinde çalışan ve başka bir WebSphere Application Server yönetim ortamı tarafından kullanılan bir uygulama tarafından üretilir.
- **Multi** IBM MQ for Multiplatforms üzerinde çalışan ve IBM MQ for Multiplatforms üzerinde çalışan diğer bağımsız uygulamalar tarafından ya da herhangi bir altyapıda WebSphere Application Server içinde çalışan uygulamalar tarafından kullanılan bağımsız bir uygulama tarafından üretilir.

Ancak, WebSphere Application Server ASCII kullandıkça ve bağımsız bir JVM, EBCDIC ' yi kullanıyorsa, allowlist dosyası aşağıdaki yöntemlerden biriyle üretiliyse, dosya kodlama sorunları olacaktır:

- z/OS' ta oluşturulan, daha sonra bağımsız uygulamalar tarafından z/OS ya da WebSphere Application Server dışında bir altyapıda çalışan bağımsız uygulamalar tarafından kullanılır.
- Generated by either WebSphere Application Server or a standalone application running on a platform other than z/OS, then used by a stand-alone application on z/OS.
- Boş olmayan her bir satır, tam olarak nitelenmiş bir sınıf adı içerir. Boş satırlar yoksayılr.
- Açıklamalar, '#' karakterini izleyen herhangi bir şey, satırın sonuna kadar eklenebilir ve yoksayılr.
- Çok temel bir karalama mekanizması vardır:
  - '\*', bir sınıf adının **son** ögesi olabilir.
  - '\*', bir sınıf adının **tek** ögesiyle eşleşir; yani, sınıfın bir parçası değil, ancak bu ögedir.

So `com.ibm.mq.*` would match `com.ibm.mq.MQMessage` but not `com.ibm.mq.jmqi.remote.api.RemoteFAP`.

Genel, belirtik bir paket adı olmayan sınıflara ilişkin varsayılan paketteki sınıflar için çalışmazsa, "\*" sınıf adı reddedilir.

- Hatalı biçimlendirilmiş allowlist dosyaları; örneğin, `com.ibm.mq.*.Message` gibi bir girdiyi içeren, genel arama karakterinin son öge olmadığı dosyalar, bir `java.lang.IllegalArgumentException` ' nin atılmasına neden olur.
- Boş bir allowlist dosyası, `ObjectMessage` kullanımını tamamen devre dışı bırakmanın etkisine sahiptir.

## Allowlist 'in biçimi virgülle ayrılmış bir liste olarak biçimlendirir

Aynı genel arama mekanizması, virgülle ayrılmış bir liste olarak bir allowlist için kullanılabilir.

- '\*', bir komut satırında ya da bir kabuk komut dosyasında ya da toplu iş dosyasında belirtildiyse, işletim sistemi tarafından genişletilebilir, bu nedenle özel işleme gerekebilir.
- '#' açıklama karakteri yalnızca bir dosya belirtildiğinde geçerlidir. Allowlist, sınıf adlarının virgülle ayrılmış bir listesi olarak belirtilirse, işletim sisteminin ya da kabuğun bunu işlemediğini varsayarak, birçok UNIX ya da Linux kabuğunda varsayılan açıklama karakteri olduğu için, normal bir karakter olarak işlem görür.

## Ödenek listesi ne zaman olur?

Uygulama ilk olarak bir `ObjectMessage setMessage()` ya da `getMessage()` yöntemini çalıştırdığında ödenek listesi başlatılır.

Sistem özellikleri değerlendirilir, allowlist dosyası açılır ve EXTACY (uygulama) kipinde, mekanizmanın kullanıma hazırlandığında, allowlistelenen sınıfların listesi yüklenir. Bu noktada, uygulama için IBM MQ JMS günlük dosyasına bir giriş yazılır.

Mekanizma ilk kullanıma hazırlandığında, parametreleri değişmeyebilir. Kullanıma hazırlama süresi, uygulama davranışına bağlı olduğu için kolay tahmin edilmemektedir. Bu nedenle, sistem özelliği ayarları ve allowlist dosya içeriği, uygulamanın başlatıldığı zamandan itibaren değişmez olarak kabul edilmelidir. Sonuçlar garanti edilmediği için, uygulama çalışırken allowlist dosyasının özelliklerini ya da içeriğini değiştirmeyin.

## Dikkate alınacak noktalar

The best approach to mitigating the risks intrinsic to the Java serialization mechanism would be to explore alternative approaches to data transfer such as using JSON instead of ObjectMessage. Using Advanced Message Security (AMS) mechanisms can add further security by ensuring that messages come from trusted sources.

Uygulamanıza sahip Java security manager mekanizmasını kullanıyorsanız, aşağıdaki izinleri vermeniz gerekir:

- FilePermission on any allowlist file that you use, with read permission for ENFORCEMENT mode, write permission for DISCOVER mode.
- **com.ibm.mq.jms.allowlist**, **com.ibm.mq.jms.allowlist.discover** ve **com.ibm.mq.jms.allowlist.mode** özelliklerindePropertyPermission (okuma).

## Daha fazla bilgi

Allowlistes ile ilgili daha fazla bilgi için bkz. [“JMS allowlist uygulamasını kurma ve kullanma” sayfa 120](#) ve [“WebSphere Application Serverinde ödenek listesi” sayfa 122](#).

### İlgili kavramlar

[“Running IBM MQ classes for JMS applications under the Java security manager” sayfa 96](#)

IBM MQ classes for JMS , Java güvenlik yöneticisi etkinleştirilmiş olarak çalışabilir. Uygulamaları Java security manager etkin bir şekilde başarıyla çalıştırmak için, Java virtual machine (JVM) olanağını uygun bir ilke yapılandırma dosyasıyla yapılandırmalısınız.

## JMS allowlist uygulamasını kurma ve kullanma

Bu bilgiler, bir allowlist ürününün nasıl çalıştığını ve bir uygulamanın işleyebileceği ObjectMessages tiplerinin bir listesini içeren bir allowlist dosyası oluşturmak için IBM MQ classes for JMS ' ta bulunan işlevleri kullanarak nasıl bir şekilde ayarladığınızı açıklar.

## Başlamadan önce

### Önemli:

Mümkün olduğunda, *izin listesi* terimi *beyaz listeleriminin* yerini almıştır. IBM MQ 9.0 ve sonraki yayınlarda, bu konuda (**com.ibm.mq.jms.\***) belirtilen Java sistem özelliği adlarını içerir. Var olan bir yapılandırmayı değiştirmeniz gerekmez. Önceki sistem özellik adları da çalışmaya devam eder.

Before starting this task, make sure that you have read and understood [“Allowisting kavramları” sayfa 117](#)

## Bu görev hakkında

Allowlisteleme işlevselliğini etkinleştirdiğinizde, IBM MQ classes for JMS bu işlevselliği aşağıdaki şekillerde kullanır:

- Bir uygulama bir ObjectMessagegöndermek istediğinde, aşağıdaki yöntemi kullanarak bunu iki yoldan biriyle yaratabilir:
  - Session.createObjectMessage(Serializable) method, passing in the object that is to be contained within the message.
  - Session.createObjectMessage() method, to create an empty ObjectMessage, and then calling ObjectMessage.setObject(Serializable) to store the object to be sent inside the ObjectMessage.



Session.createObjectMessage(Serializable) ya da ObjectMessage.setObject(Serializable) yöntemleri çağrıldığında, JMS için sınıflar, geçirilen nesnenin allowlist 'te adı geçen bir tipte olup olmadığını denetler.

Söz konusu tip söz konusu ise, nesne diziselleştirilir ve ObjectMessage içinde saklanır. Ancak, nesne, allowlist 'te olmayan bir tipte, IBM MQ classes for JMS iletiyi içeren bir JMSEException yayınlayın:

```
JMSCC0052: Nesne diziselleştirilirken kural dışı durum oluştu:  
'java.io.InvalidClassException: < object class>; The class serileştirilmeyebilir  
ya da allowlist '< allowlist>' içine dahil edilmediği için dizisel biçimden geri çevrildi.
```

uygulamaya geri dönün.

**Önemli:** Kural dışı durum Session.createObjectMessage(Serializable) yönteminden atılırsa, ObjectMessage yaratılmaz. Similarly, if the JMSEException is thrown from the ObjectMessage.setObject(Serializable) method, the object will not be added to the ObjectMessage.

- Bir uygulama ObjectMessage(Nesne İletisi) alırsa, nesnenin içinde bulunan nesneyi almak için ObjectMessage(Nesne İletisi).getObject() yöntemini çağırır. When this method is called, the IBM MQ classes for JMS check the type of object contained within the ObjectMessage, to see if that object is of a type specified in the allowlist.

Bu durumda, nesne dizisel biçimden geri çevrilir ve uygulamaya geri döndürülür. Ancak, nesne, allowlist 'te olmayan bir tipte, IBM MQ classes for JMS iletiyi içeren bir JMSEException yayınlayın:

```
JMSCC0053: Bir ileti dizisel biçimden geri çevrilirken kural dışı durum oluştu:  
'java.io.InvalidClassException: < object class>; The class may not be  
diziselleştirilmiş ya da dizisel biçimden geri çevrilmemiş olarak dizisel biçimden  
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);
```

Allowlistleme etkinleştirilmediği için, uygulama iletiyi gerekli hedefe başarıyla yerleştirebilir.

Tek bir giriş ( java.net.URL ) içeren C:\allowlist.txt adlı bir dosya oluşturursanız ve uygulamayı Java sistem özelliği kümesiyle yeniden başlarsanız:

```
-Dcom.ibm.mq.jms.allowlist=file:/C:/allowlist.txt
```

Allowlist işlevi etkindir. Uygulama hala, allowlist 'te belirtilen tip olarak java.net.URI tipinde bir nesne içeren ObjectMessage ögesini yaratabilir ve gönderebilirler.

However, if you change the allowlist.txt file so that the file contains the single entry java.util.Calendar, as the allowlist functionality is still enabled, when the application calls:

```
ObjectMessage msg = session.createObjectMessage(testURL);
```

IBM MQ classes for JMS allowlist ögesini denetleyin ve java.net.URI için bir giriş içermediğini bulun.

Sonuç olarak, JMSCC0052 iletisinin bulunduğu bir JMSEException iletisi yayınlanır.

Similarly, suppose you have another application that receives ObjectMessages using this code:

```
ObjectMessage message = (ObjectMessage)receiver.receive(30000);  
if (message != null) {  
    Object messageBody = objectMessage.getObject();  
    if (messageBody instanceof java.net.URI) {  
        : : : : : : :  
    }  
}
```

Allowlistening etkinleştirilmediyse, uygulama herhangi bir nesne için nesne içeren ObjectMessages (Nesne İletileri) alabilir. The application then checks if the object is of type java.net.URL before performing the appropriate processing.

Uygulamayı şimdi Java sistem özelliği ile başlıyorsanız:

```
-Dcom.ibm.mq.jms.allowlist=java.net.URL
```

set, allowlocting işlevselliği açık. Uygulama çağrıldığında:

```
Object messageBody = objectMessage.getObject();
```

ObjectMessage.getObject() yöntemi yalnızca java.net.URL türündeki nesnelere döndürür.

ObjectMessage içinde yer alan nesne bu tipte değilse, ObjectMessage.getObject() yöntemi, JMSSC0053 iletisini içeren bir JMSEException yayınlıyor. Daha sonra, uygulamanın iletiye ne yapacağına karar vermesi gerekir; örneğin, ileti, o kuyruk yöneticisi için ölü harf kuyruğunda taşınabilir.

Uygulama, yalnızca ObjectMessage içindeki nesne java.net.URL tipteyse olağan şekilde döndürülür.

## Yordam

1. Aşağıdaki Java sistem özellikleri ile ObjectMessages işlemini 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 yaratır.

2. Uygulama, belirli bir süre içinde ObjectMessages temsilcisinin bir örneğini işledikten sonra, bunu durdurun.

Allowlist dosyası, uygulamanın çalışırken işlendiği ObjectMessages içinde bulunan tüm nesne tiplerinin bir listesini içerir.

Uygulamayı yeterli bir süre çalıştırdıysanız, bu liste uygulamanın işleyeceği olası ObjectMessages içinde bulunan olası tüm nesne tiplerini içerir.

3. Aşağıdaki sistem özelliği kümesiyle uygulamayı yeniden başlatın:

```
-Dcom.ibm.mq.jms.allowlist=file:/<path to your allowlist file>
```

Bu, allowlistening olanağını etkinleştirir ve IBM MQ classes for JMS , allowlist olmayan bir tipin ObjectMessage ögesini algılasa, JMSSC0052 ya da JMSSC0053 içeren bir JMSEException yayınlanır.

## WebSphere Application Serverinde ödenek listesi

How you use IBM MQ classes for JMS allowlisting in WebSphere Application Server.

### Önemli:

Mümkün olduğunda, *izin listesi* terimi *beyaz listeler*iminin yerini almıştır. IBM MQ 9.0 ve sonraki yayınlarda, bu konuda (**com.ibm.mq.jms.\***) belirtilen Java sistem özelliği adlarını içerir. Var olan bir yapılandırmayı değiştirmeniz gerekmez. Önceki sistem özellik adları da çalışmaya devam eder.

WebSphere Application Server kurulumunuzun, allowlicating özelliğini destekleyen bir IBM MQ kaynak bağdaştırıcısı sürümü içerdiğinden emin olmalısınız.

İki ürünün kullanılmasına ilişkin ek bilgi için [“IBM MQ ve WebSphere Application Server ' in birlikte kullanılması” sayfa 472 ' e bakın.](#)

IBM MQ 9.0.0 Fix Pack 1 , uygun işlevselliği içerir.

Uygulama sunucusu güncellendikten sonra, aşağıdaki Java sistem özelliklerini kullanabilirsiniz:

- -Dcom.ibm.mq.jms.allowlist
- -Dcom.ibm.mq.jms.allowlist.discover

[“JMS allowlist uygulamasını kurma ve kullanma” sayfa 120](#) içinde açıklanmaktadır.

**Not:** You need to set the Java system properties as generic JVM arguments, on the Java virtual machine used to run the application server, and the application server restarted for the changes to take effect.

Ek bilgi için [Java sanal makine ayarları](#) içindeki *Genel JVM bağımsız değişkenleriyle* ilgili bölüme bakın.

Özellikleri ayarlamak için, *Süreç tanımlamaları* 'nda Java virtual machine penceresine gidin ve uygun bağımsız değişkeni girin.

Aşağıdaki ayar:

```
-Dcom.ibm.mq.jms.allowlist=<youruserId>_MyObject
```

Uygulama sunucusunun allowlist *youruserId\_MyObject* 'ı kullanmasına neden olur. Yalnızca tip nesnelere uygulama sunucusu tarafından işlenir.

Aşağıdaki ayarlar:

```
-Dcom.ibm.mq.jms.allowlist.discover=true  
-Dcom.ibm.mq.jms.allowlist=file:C:/allowlist.txt
```

configure the application server to use *Keşfet* mode, and record details of the JMS ObjectMessages, that the application server processes, to the file C:\allowlist.txt

Aşağıdaki ayar:

```
-Dcom.ibm.mq.jms.allowlist=file:C:/allowlist.txt
```

uygulama sunucusunun C:/allowlist.txt dosyasını yüklemesine neden olur ve allowlist 'i belirlemek için o dosyadaki bilgileri kullanır.

### İlgili kavramlar

[“Running IBM MQ classes for JMS applications under the Java security manager”](#) sayfa 96

IBM MQ classes for JMS , Java güvenlik yöneticisi etkinleştirilmiş olarak çalışabilir. Uygulamaları Java security manager etkin bir şekilde başarıyla çalıştırmak için, Java virtual machine (JVM) olanağını uygun bir ilke yapılandırma dosyasıyla yapılandırmalısınız.

## IBM MQ classes for JMS' da karakter dizgisi dönüştürmeleri

IBM MQ classes for JMS , doğrudan karakter dizesi dönüştürmesi için CharsetEncoders ve CharsetDecoders 'ı doğrudan kullanır. Karakter dizesi dönüştürmesi için varsayılan davranış, iki sistem özelliği ile yapılandırılabilir. Eşlenebilir karakterler içeren iletilerin işlenmesi, UnmappableCharacterişlemi ve yerine koyma byte 'ları ayarlanmak üzere ileti özellikleri kullanılarak 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 gerçekleştirilmiştir.

Bu yöntemlerin herhangi birini kullanarak, bozuk biçimli ya da çevrilemeyen verilerin varsayılan yerine ( REPLACE) ilişkin sonuçlar elde edilebilir. This behavior can obscure errors in applications, and lead to unexpected characters, for example ?, in translated data.

IBM MQ 8.0'tan bu tür sorunları daha önce ve daha etkin bir şekilde algılamak için IBM MQ classes for JMS , CharsetEncoders ve CharsetDecoders 'ı doğrudan kullanır ve yanlış biçimli ve çevrilemeyen verilerin işlenmesini açık bir şekilde yapılandırır. Varsayılan davranış, uygun bir MQException yayınlamak REPORT 'e ait olur.

## Yapılandırılıyor

Translating from UTF-16 (the character representation used in Java) to a native character set, such as UTF-8, is termed encoding, while translating in the opposite direction is termed decoding.

Şu anda kod çözme işlemi, CharsetDecoders için varsayılan davranış, kural dışı durum yayınlamak hata bildirme davranışını ele alır.

Bir ayar, hem kodlamada hem de kod çözülürken hata işlenmesini denetlemek üzere bir `java.nio.charset.CodingErrorAction` belirlemek için kullanılır. Diğer bir ayar, kodlamayı

kodlarken, yeni baytı ya da byte 'ları denetlemek için kullanılır. Kod çözme işlemlerinde varsayılan Java yerine koyma dizgisi kullanılacak.

## JMS için IBM MQ Sınıflarındaki UnmappableCharacter işlemleri ve yerine koyma byte 'ları ayarları

IBM MQ 8.0' tan, UnmappableCharacter işlemlerini ve yerine koyma byte 'ları ayarlamak için iki özellik daha kullanılabilir. Uygun sabit tanımlamalar `com.ibm.msg.client.wmq.WMQConstants` içinde

### JMS\_IBM\_UNMAPPABLE\_ACTION

Bir kodlamada ya da kod çözme işleminde bir karakter eşlenemediğinde uygulanacak `CodingErrorAction` 'i ayarlar ya da uygular.

Bunu `CodingErrorAction.{REPLACE|REPORT|IGNORE}.toString()` olarak aşağıdaki gibi ayarlamamız gerekir:

```
public static final String JMS_IBM_UNMAPPABLE_ACTION = "JMS_IBM_Unmappable_Action";
```

### JMS\_IBM\_UNMAPPABLE\_REPLACEMENT

Bir kodlama işleminde bir karakter eşlenemediğinde uygulanacak ikame baytları belirler ya da alır.

Varsayılan Java yerine koyma dizgisi, kodu çözme işlemlerinde kullanılır.

```
public static final String JMS_IBM_UNMAPPABLE_REPLACEMENT = "JMS_IBM_Unmappable_Replacement";
```

`JMS_IBM_UNMAPPABLE_ACTION` ve `JMS_IBM_UNMAPPABLE_REPLACEMENT` özellikleri, varış noktalarında ya da iletilerde ayarlanabilir. İletide ayarlanan bir değer, iletinin gönderilmekte olduğu hedefte belirlenen değeri geçersiz kılar.

`JMS_IBM_UNMAPPABLE_REPLACEMENT` 'in tek bir byte olarak ayarlanması gerektiğini unutmayın.

## Sistem varsayılanlarını ayarlamak için sistem özellikleri

IBM MQ 8.0' tan, karakter dizgisi dönüştürmeyle ilgili varsayılan davranışı yapılandırmak için aşağıdaki iki Java sistem özelliği kullanılabilir.

### `com.ibm.mq.cfg.jmqi.UnmappableCharacterAction`

Kodlama ve kod çözme işlemleri için çevrilemeyen veriler için yapılacak işlemi belirtir. Değer `REPORT`, `REPLACE` ya da `IGNORE` olabilir.

### `com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement`

Bir kodlama işleminde bir karakter eşlenemediğinde uygulanacak ikame baytları ayarlar ya da alır.

Varsayılan Java yerine koyma dizgisi, kod çözme işlemlerinde kullanılır.

Java karakteri ile yerel bayt gösterimleri arasındaki karışıklığı önlemek için, yerel karakter kümesindeki yerine koyma byte 'ı gösteren ondalık sayı olarak `com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement` değerini belirtmeniz gerekir.

For example, the decimal value of ?, as a native byte, is 63 if the native character set is ASCII-based, such as ISO-8859-1, while it is 111 if the native character set is EBCDIC.

**Not:** Note that if an `MQMD` or `MQMessage` object has either the `unmappableAction` or `unMappableReplacement` fields set, then the values of these fields take precedence over the Java system properties. Bu, Java sistem özellikleri tarafından belirtilen değerlerin, gerekirse her ileti için geçersiz kılınabilmesini sağlar.

## İlgili kavramlar

“IBM MQ classes for Java' da karakter dizgisi dönüştürmeleri” sayfa 325

IBM MQ classes for Java , doğrudan karakter dizesi dönüştürmesi için `CharsetEncoders` ve `CharsetDecoders` 'ı doğrudan kullanır. Karakter dizesi dönüştürmesi için varsayılan davranış, iki sistem özelliği ile yapılandırılabilir. Eşlenebilir karakterler içeren iletilerin işlenmesi, `com.ibm.mq.MQMD` aracılığıyla yapılandırılabilir.

## IBM MQ classes for JMS uygulamaları yazılıyor

JMS modeline kısa bir giriş yaptıktan sonra bu konu, IBM MQ classes for JMS uygulamalarının nasıl yazılabileceğiyle ilgili ayrıntılı kılavuz sağlar.

### JMS modeli

The JMS model defines a set of interfaces that Java applications can use to perform messaging operations. IBM MQ classes for JMS, JMS sağlayıcısı olarak, JMS nesnelerinin IBM MQ kavramlarına nasıl ilişkin olduğunu tanımlar. JMS belirtimi, belirli JMS nesnelerinin yönetilecek nesnelere olmasını bekler. JMS 2.0 , JMS 1.1'tan klasik API' yi de korurken, basitleştirilmiş bir API ' yi sunar.

The JMS specification and the javax.jms package define a set of interfaces that Java applications can use to perform messaging operations.

From IBM MQ 8.0, IBM MQ supports the JMS 2.0 version of the JMS standard, which introduces a simplified API, while also retaining the classic API, from JMS 1.1.

### Basitleştirilmiş API

JMS 2.0 introduces the simplified API, while also retaining the domain specific and domain independent interfaces from JMS 1.1. Basitleştirilmiş API, ileti göndermek ve almak için gerekli olan nesnelerin sayısını azaltır ve aşağıdaki arabirimlerden oluşur:

#### ConnectionFactory

ConnectionFactory , bir JMS istemcisi tarafından bir Bağlantı oluşturmak için kullanılan, yönetilen bir nesnedir. Bu arabirim, klasik API ' da da kullanılır.

#### JMSBağlam

Bu nesne, klasik API ' nin Connection ve Session nesnelerini birleştirir. JMSBağlam nesnelere, diğer JMSBağlam nesnelere yaratılabilir ve temeldeki bağlantı yinelenir.

#### JMSÜretici

Bir JMSÜreticisi bir JMSBağlamı tarafından yaratılır ve bir kuyruğa ya da konuya ileti göndermek için kullanılır. JMSÜreticisi nesnesi, iletiyi göndermek için gereken nesnelerin yaratılmasına neden olur.

#### JMSTüketici

JMSTüketicisi bir JMSBağlamı tarafından oluşturulur ve bir konu ya da kuyruktan ileti almak için kullanılır.

Basitleştirilmiş API ' nin bir dizi etkisi vardır:

- JMSBağlam nesnesi her zaman temel bağlantıyı otomatik olarak başlatır.
- JMSProducers and JMSConsumers can now work directly with message bodies, without having to get the whole message object, by using the Message's `getBody` method.
- Bir 'body' göndermeden önce, bir 'body' göndermeden önce, JMSÜreticisi nesnesi üzerinde ileti özellikleri ayarlanabilir. JMSÜreticisi, iletiyi göndermek için gerekli olan tüm nesnelerin yaratılmasını işleyecek. JMS 2.0 kullanılarak, özellikler ayarlanabilir ve aşağıdaki gibi gönderilen bir ileti kullanılabilir:

```
context.createProducer().
setProperty("foo", "bar").
setTimeToLive(10000).
setDeliveryMode(NON_PERSISTENT).
setDisableMessageTimestamp(true).
send(dataQueue, body);
```

JMS 2.0 , iletilerin birden çok tüketici arasında paylaşılabilirdiği paylaşımlı abonelikler de sunar. Tüm JMS 1.1 abonelikleri paylaşılmayan abonelikler olarak ele alınır.

### Klasik API

Aşağıdaki liste, klasik API ' nin ana JMS arabirimlerini özetlemektedir:

## Hedef

Bir hedef, bir uygulamanın iletileri gönderdiği ya da bir uygulamanın iletileri aldığı bir kaynaktır ya da her ikisi de olabilir.

## ConnectionFactory

Bir ConnectionFactory nesnesi, bir bağlantı için bir yapılandırma özellikleri kümesini sarsalıyor. Bir uygulama, bağlantı oluşturmak için bir bağlantı üreticisi kullanır.

## Bağlantı

Bir bağlantı nesnesi, bir uygulamanın ileti sistemi ile etkin bağlantısını sarsalıyor. Uygulama, oturum yaratmak için bir bağlantı kullanır.

## Session

Oturum, ileti göndermek ve almak için tek bir iş parçacıklı bağlamdır. Bir uygulama, ileti, ileti üreticileri ve ileti tüketicileri oluşturmak için bir oturumu kullanır. Bir oturum hareket edilir ya da hareket edilmez.

## İleti

Bir ileti nesnesi, bir uygulamanın gönderdiği ya da gönderdiği bir iletiyi sarsalıyor.

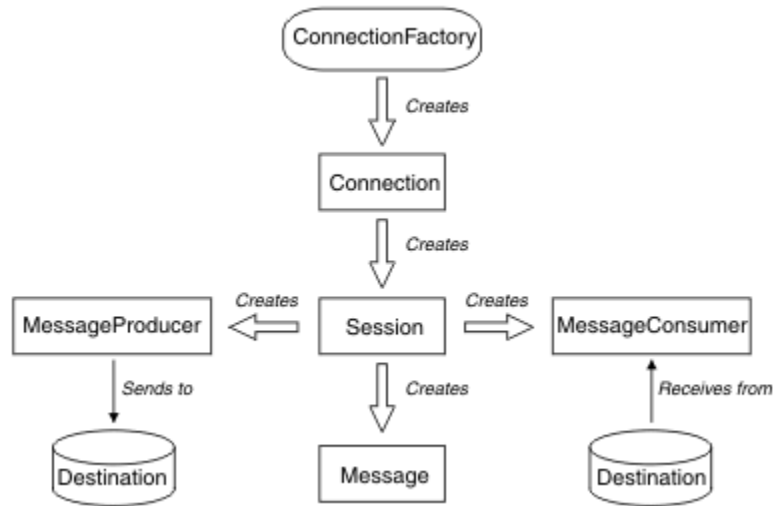
## MessageProducer

Bir uygulama, iletileri bir hedefe göndermek için bir ileti üreticisi kullanır.

## MessageConsumer

Bir uygulama, bir hedefe gönderilen iletileri almak için bir ileti tüketicisi kullanır.

Şekil 9 sayfa 126 , bu nesneleri ve bunların ilişkilerini gösterir.



Şekil 9. JMS nesneleri ve ilişkileri

Bir Hedef, ConnectionFactoryya da Connection nesnesi, çok iş parçacıklı bir uygulamanın farklı iş parçacıkları tarafından eşzamanlı olarak kullanılabilir, ancak bir Oturum, MessageProducerya da MessageConsumer nesnesi farklı iş parçacıkları tarafından koşut zamanlı olarak kullanılamaz. Bir Oturum, MessageProducerya da MessageConsumer nesnesinin koşut zamanlı olarak kullanılmamasını sağlamanın en basit yolu, her iş parçacığı için ayrı bir Oturum nesnesi yaratmamaktır.

JMS , iki ileti sistemini destekler:

- Noktadan Noktaya İleti Sistemi
- Yayınlama/abone olma ileti alışverişi

Bu ileti alışverişi biçimleri de *ileti sistemi etki alanları* olarak da adlandırılır ve her iki ileti sistemi stilini de bir uygulamada birleştirebilirsiniz. Noktadan noktaya iletişim alanında, hedef bir kuyruktır ve yayınlama/abone olma etki alanında, hedef bir konudur.

JMS 1.1 tarihinden önce JMS sürümleriyle, noktadan noktaya iletişim etki alanı için programlama bir arabirim ve yöntem kümesini kullanır ve yayınlama/abone olma etki alanı için programlama başka bir

küme kullanır. İki set birbirine benzer, ama ayrı. JMS 1.1' tan, hem ileti sistemi etki alanlarını destekleyen, ortak bir arabirim ve yöntem kümesi kullanabilirsiniz. Ortak arabirimler, her ileti sistemi etki alanı için bir etki alanı bağımsız görünümü sağlar. Çizelge 14 sayfa 127 , JMS etki alanı bağımsız arabirimlerini ve bunlara karşılık gelen etki alanlarını özel arabirimlerini listeler.

<i>Çizelge 14. JMS etki alanı bağımsız ve etki alanına özgü arabirimler</i>		
<b>Etki alanı bağımsız arabirimleri</b>	<b>Noktadan noktaya etki alanına ilişkin etki alanına özgü arabirimler</b>	<b>Yayınlama/abone olma etki alanı için etki alanına özgü arabirimler</b>
ConnectionFactory	QueueConnectionÜreticisi	TopicConnectionÜreticisi
Bağlantı	QueueConnection	TopicConnection
Hedef	Kuyruk	Konu
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

JMS 2.0 , etki alanına özgü tüm arabirimleri korur ve bu nedenle var olan uygulamalar bu arabirimleri kullanmaya devam edebilir. For new applications, however, consider using the domain independent interfaces of JMS 1.1 or the simplified API of JMS 2.0.

IBM MQ classes for JMS içinde, JMS nesnelere IBM MQ kavramlarına aşağıdaki şekillerde ilgilidir:

- Bağlantı nesnesinin, bağlantıyı yaratmak için kullanılan bağlantı üreticisinin özelliklerinden türetilmiş özellikleri vardır. 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.
- Bir oturum nesnesi, oturumun işlem kapsamını tanımlayan bir IBM MQ bağlantı tanıtıcısını sarmadığı.
- Bir MessageProducer nesnesi ve her biri bir IBM MQ nesne tanıtıcısını sarmalayan bir MessageConsumer nesnesinden.

When using IBM MQ classes for JMS, all the normal rules of IBM MQ apply. Özellikle, bir uygulamanın uzak bir kuyruğa ileti gönderebileceği, ancak yalnızca, uygulamanın bağlı olduğu kuyruk yöneticisinin sahip olduğu bir kuyruktan bir ileti alabileceği unutulmamalı.

JMS belirtimi, ConnectionFactory ve Hedef nesnelere denetlenmesini bekler. Bir denetimci, denetlenen nesnelere merkezi bir havuzda yaratır ve bakımını yapar; JMS uygulaması bu nesnelere Java Naming and Directory Interface (JNDI) kullanarak alır.

IBM MQ classes for JMS' ta, Hedef arabirimin somutlaması, Kuyruk ve Konu 'nın soyut bir üst sınıfıdır; dolayısıyla, Hedef eşgörünümü bir Kuyruk nesnesi ya da Konu nesnesidir. Etki alanı bağımsız arabirimleri, bir kuyruğu ya da konuyu hedef olarak kabul eder. Bir MessageProducer ya da MessageConsumer nesnesine ilişkin ileti alışverişi etki alanı, hedefin bir kuyruk mu, yoksa bir konu mu tarafından belirlenir.

Bu nedenle, IBM MQ classes for JMS içinde aşağıdaki tiplerin nesnelere denetleyebilirler:

- ConnectionFactory
- QueueConnectionÜreticisi
- TopicConnectionÜreticisi
- Kuyruk
- Konu
- XAConnectionFactory
- XAQueueConnectionÜreticisi

- XATopicConnectionÜreticisi

## İlgili kavramlar

IBM MQ Java dil arabirimleri

[“JMS 2.0 işlevselliğini kullanma” sayfa 299](#)

JMS 2.0 , IBM MQ classes for JMS' a birçok yeni işlev alanı sunar.

## JMS ileti

JMS iletileri bir üstbilginin, özelliklerden ve bir gövdeden oluşur. JMS , beş ileti gövdesi tipini tanımlar.

JMS iletileri aşağıdaki kısımlardan oluşur:

### Üstbilgi

Tüm iletiler aynı üstbilgi alanları kümesini destekler. Üstbilgi alanları, iletileri tanımlamak ve yönlendirmek için hem istemciler, hem de sağlayıcılar tarafından kullanılan değerleri içerir.

### Özellikler

Her ileti, uygulama tanımlı özellik değerlerini desteklemek için yerleşik bir olanak içerir. Özellikler, uygulama tanımlı iletilere süzgeç uygulamak için verimli bir mekanizma sağlar.

### Gövde

JMS , kullanılmakta olan ileti alışverişi biçemlerinin çoğunluğunu kapsayan beş ileti gövdesi tipini tanımlar:

#### Akış

Java temel değerleri akışı. Doldurulan ve sıralı bir şekilde okunuyor.

#### Eşlem

Adların dizgiler ve değerler olduğu ad-değer çiftleri kümesi, Java ilkel tipleridir. Girişlere sırayla ya da rasgele ad temelinde erişilebilir. Girişlerin sırası tanımsız.

#### Metin

A message containing a java.lang.String.

#### Nesne

Diziselleştirilebilir bir Java nesnesi içeren bir ileti

#### Bayt

Yorumlanmamış byte akışıdır. Bu ileti tipi, bir güvenin var olan bir ileti biçimiyle eşleşmesi için tam olarak kodlanır.

JMSCorrelationID üstbilgi alanı, bir iletiyi başka bir iletiyi bağlamak için kullanılır. Bu ileti genellikle, bir yanıt iletilisini istek isteyen iletilisiyle bağlantıdır. JMSCorrelationID , sağlayıcıya özgü bir ileti tanıtıcısını, uygulamaya özgü bir dizgiyi ya da bir sağlayıcı-yerel bayt [] değerini barınabiliyor.

### JMSiçindeki ileti seçicileri

İletiler, uygulama tanımlı özellik değerleri içerebilir. An application can use message selectors to have a JMS provider filter messages.

İleti, uygulama tanımlı özellik değerlerini desteklemek için yerleşik bir olanak içerir. Sonuç olarak, bu, uygulamaya özgü üstbilgi alanlarını bir iletiye eklemek için bir mekanizma sağlar. Properties allow an application, using message selectors, to have a JMS provider select or filter messages on its behalf, using application-specific criteria. Uygulama tanımlı özellikler aşağıdaki kurallara uymalıdır:

- Özellik adları, bir ileti seçici tanıtıcısına ilişkin kurallara uymalıdır.
- Özellik değerleri Boole, bayt, short, int, long, float, double, ve String olabilir.
- JMSX ve JMS\_ ad örnekleri ayrılmıştır.

Özellik değerleri, bir ileti göndermeden önce ayarlanır. Bir istemci bir ileti aldığı anda, ileti özellikleri salt okunurdur. Bir istemci bu noktada özellikleri ayarlama girişiminde bulunursa, bir MessageNotWriteableException yayınlanır. clearProperties çağrılırsa, özellikler artık hem okunabilecek, hem de yazılacak şekilde yazılabilir.

Bir özellik değeri, ileti gövdesindeki bir değeri yineleyebilir. JMS , bir özelliğe yapılmamanın yapılabileceği bir ilke tanımlamaz. Ancak, uygulama geliştiricilerin, JMS sağlayıcılarının bir ileti gövdesindeki verileri ileti



özelliklerinde verilerden daha verimli bir şekilde işleyeceğini dikkate almaları gerekir. En iyi performans için, uygulamaların ileti özelliklerini yalnızca bir ileti üstbilgisini özelleştirmeleri gerektiğinde kullanması gerekir. Bunu yapmanın birincil nedeni, uyarlanmış ileti seçimini desteklemesidir.

Bir JMS ileti seçici, bir istemcinin ileti üstbilgisini kullanarak, ilgilendiği iletileri belirtmesine olanak sağlar. Yalnızca, seçiciyle eşleşen üstbilgileri içeren iletiler teslim edilir.

İleti seçicileri, ileti gövdesi değerlerine gönderme yapamazlar.

Bir ileti seçici, ileti üstbilgisi alanı ve özellik değerleri, seçicide karşılık gelen tanıtıcılarının yerine geçtiğinde, seçici doğru olarak değerlendirildiğinde bir iletiyle eşleşir.

İleti seçici, sözdizimi, SQL92 koşullu ifade sözdiziminin bir alt kümesine dayalı olan bir Dizedir. Bir ileti seçicinin değerlendirdiği sıra, öncelik düzeyi içinde soldan sağa doğru olur. Bu siparişi değiştirmek için araçları kullanabilirsiniz. Önceden tanımlı seçici hazır bilgileri ve işleç adları burada büyük harfle yazılır; ancak, büyük/küçük harf duyarlı değildir.

## Bir ileti seçicisinin içeriği

Bir ileti seçici şunları içerebilir:

- Hazır Bilgiler
  - Dizgi hazır bilgisi tırnak içine alınır. Çift tırnak işareti, bir tırnak işaretini temsil eder. 'Literal' ve 'literal' (hazır bilgi) örnekleri verilebilir. Java dizgi hazır bilgileri gibi, bunlar Unicode karakter kodlamasını kullanır.
  - Tam sayısal hazır bilgi, 57, -957 ve +62 gibi ondalık bir nokta içermeyen bir sayısal değerdir. Java uzunluğundaki sayılar desteklenir.
  - Yaklaşık sayısal hazır bilgi, bilimsel gösterimde ( 7E3 ya da -57.9E2gibi) sayısal bir değerdir ya da 7., -95.7ya da +6.2gibi bir ondalık değer. Java double (Çift) aralığındaki numaralar desteklenir.
  - Boole tipi hazır bilgiler TRUE ve FALSE değerini verir.
- Tanıtıcılar:
  - An identifier is an unlimited length sequence of Java letters and Java digits, the first of which must be a Java letter. Bir harf, Character.isJavaLetter yönteminin true (doğru) değerini döndürdüğü herhangi bir karakterdir. Bu, \_ ve \$' ı içerir. Bir harf ya da rakam, Character.isJavaLetterOrBasamağın true değerini döndürdüğü herhangi bir karakterdir.
  - Tanıtıcılar NULL, TRUE ya da FALSE adları olamaz.
  - Tanıtıcılar NOT, AND, OR, BETWE, LIKE, IN ya da IS olamaz.
  - Tanıtıcılar, üstbilgi alanı başvuruları ya da özellik başvurularıdır.
  - Tanıtıcılar büyük ve küçük harfe duyarlıdır.
  - İleti üstbilgisi alanı başvuruları şu şekilde kısıtlanmıştır:
    - JMSDeliveryMode
    - JMSönceliği
    - JMSMessageID
    - JMSTimestamp
    - JMSCorrelationID
    - JMSTypeJMSMessageID, JMSTimestamp, JMSCorrelationIDve JMSType değerleri boş değerli olabilir ve bu durumda boş değer (NULL) olarak işlem görür.
  - JMSX ile başlayan herhangi bir ad, JMStanımlı bir özellik adıdır.
  - JMS\_ ile başlayan herhangi bir ad, sağlayıcıya özgü bir özellik adıdır.

- JMS ile başlamayan herhangi bir ad, uygulamaya özgü bir özellik adıdır. Bir iletide var olmayan bir özelliğe ilişkin başvuru varsa, değeri NULL (boş değerli) olur. Böyle bir değer varsa, bu değer, ilgili özellik değeridir.
- Beyaz alan, Java için tanımlanandır: boşluk, yatay sekme, form besleme ve satır sonlandırıcı.
- İfadeler:
  - Seçici, koşullu bir ifadedir. Gerçek eşleşmeleri değerlendiren bir seçici; yanlış ya da bilinmeyen olarak değerlendirilen bir seçici eşleşmez.
  - Aritmetik ifadeler, kendilerinden, aritmetik işlemlerden, tanıticılardan (sayısal hazır bilgi olarak kabul edilen bir değere sahip) ve sayısal hazır bilgiler içerir.
  - Koşullu ifadeler kendilerinden, karşılaştırma işlemlerinden ve mantıksal işlemlerden oluşur.
- İfadelerin değerlendirilen sıralamayı ayarlamak için standart destek pazarlama () desteklenir.
- Mantıksal işlemler öncelik sırasına göre: NOT, AND, OR.
- Karşılaştırma işlemleri: =, >, >=, <, <=, <> (eşit değil).
  - Yalnızca aynı tipteki değerler karşılaştırılabilir. Bir kural dışı durum, tam sayısal değerleri ve yaklaşık sayısal değerleri karşılaştırmak için geçerli bir değerdir. (Gerekli tür dönüştürme işlemi, Java sayısal yükseltme kurallarına göre tanımlanır.) Farklı tipleri karşılaştırma girişiminde bulunulursa, seçici her zaman false olur.
  - Dizgi ve Boole karşılaştırması = ve <> ile kısıtlanmıştır. İki dizgi, yalnızca aynı karakter dizisini içeriyorsa eşittir.
- Aritmetik işlemler öncelik sırasına göre:
  - +,-birli.
  - \*,/, çarpma ve bölme.
  - +,-, toplama ve çıkarma.
  - Boş değer (NULL) olan aritmetik işlemler desteklenmez. Bunlar denenirse, tam seçici her zaman false olur.
  - Aritmetik işlemler Java sayısal promosyonu kullanmalıdır.
- arithmetic-expr1 [ NOT ] BETWEEN arithmetic-expr2 ve arithmetic-expr3 karşılaştırma işleci:
  - 15-19 yaş arası, yaş >= 15 ve yaş <= 19 olarak eşittir.
  - 15-19 yaş değerleri, < 15 YA DA age > 19 yaşının eşdeğeridir.
  - BETWEEN işleminin ifadelerinden herhangi biri NULL (boş) ise, işlemin değeri false olur. BETWEEN işleminin NOT NULL ifadelerinden herhangi biri NULL (boş değer) ise, işlemin değeri true olur.
- tanıtıcı [ NOT ] IN (string-literal1, string-literal2, ...) tanıtıcısı bir Dize ya da NULL değerine sahip olan karşılaştırma işleci.
  - Country IN ('UK ',' US ',' France '), 'UK için 've' Peru için 'yanlış ' için geçerlidir. Bu ifade (Country = 'UK') OR (Country = 'US ') YA DA (Country = 'France') ifadesine eşdeğerdir.
  - Country NOT IN ('UK ',' US ',' France ') 'UK için yanlış ve' Peru için doğru '. NOT ((Ülke = 'UK ') YA DA (Ülke = 'ABD') YA DA (Ülke = 'Fransa ')) ifadelerine eşdeğerdir.
  - IN ya da NOT IN işleminin tanıtıcısı NULL ise, işlemin değeri bilinmez.
- Tanıtıcı [ NOT ] LIKE örüntü-değeri [ ESCAPE escape-character ] karşılaştırma işleci, burada tanıtıcının dizgi değeri var. pattern-value bir dizgi hazır bilgisidir; burada \_ herhangi bir tek karakter için ve% herhangi bir karakter dizisi anlamına gelir (boş sıra dahil). Diğer tüm karakterler kendileri için geçerli. İsteğe bağlı çıkış karakteri tek bir karakter dizisi hazır bilgisidir; örüntüdeki \_ ve% karakterlerinin özel anlamlarından kaçmak için kullanılan karakter.
  - '12%3' gibi bir telefon 123 ve 12993 için geçerlidir ve 1234 için false değerini içerir.
  - '\_se' gibi sözcük, "kaybetmek" için doğrudur ve "gevşek" için "false".
  - '\\\_%' ESCAPE '\\', "\_foo" için "true" ve "bar" için false değerini içerir.
  - '12%3', 123 ve 12993 için yanlış ve 1234 için doğru (true) olarak değil.

- LIKE ya da NOT LIKE işleminin tanıtıcısı NULL ise, işlemin değeri bilinmiyor demektir.
- tanıtıcı, boş değerli bir üstbilgi alanı değeri ya da eksik bir özellik değeri için boş değerli (NULL) karşılaştırma işleci sınamaları.
  - prop\_name IS NULL.
- tanıtıcı, boş değerli olmayan bir üstbilgi alan değeri ya da bir özellik değeri varlığı için NOT NULL karşılaştırma işleci sınamaları DEĞİL.
  - prop\_name BOŞ DEĞİL.

## Bir ileti seçici örneği

Aşağıdaki ileti seçici, ileti tipi araba, mavi renk ve ağırlığı 2500 lbs 'den büyük olan iletileri seçer:

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

## NULL özellik değerleri

Yukarıdaki listede belirtildiği gibi, özellik değerleri NULL (boş değerli) olabilir. NULL değerler içeren seçici ifadelerinin değerlendirilmesi, SQL 92 NULL anlambilimi ile tanımlanır. Aşağıdaki liste, bu anlambilimi hakkında kısa bir açıklama sağlar:

- SQL, bir NULL değerini bilinmeyen olarak değerlendirir.
- Bilinmeyen bir değere sahip 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.

## Special behavior of JMSMessageID and JMSCorrelationID

The IBM MQ classes for JMS contain optimizations when selecting messages from a queue based on either JMSMessageID or JMSCorrelationID.

Bir uygulama, formun bir seçicisini belirtiyorsa:

```
JMSMessageID= 'ID:message_id'
```

Burada *message\_id* , standart bir IBM MQ ileti tanıtıcısı içeren bir dizgidir; daha sonra JMS için IBM MQ sınıfları, belirtilen ileti tanıtıcısına sahip iletiyi almak için **MatchOption** MQMO\_MATCH\_MSG\_IDusesınıfını kullanır.

Örneğin, bir kuyruktan 414D51207061756C745639314C545320C57C1A5F25ECE602 ileti tanıtıcısına sahip bir ileti almak için, uygulamanın şu ileti seçiciyi kullanması gerekir:

```
JMSMessageID= ' ID:414D51207061756C745639314C545320C57C1A5F25ECE602 '
```

Benzer bir şekilde, uygulama, biçimi olan bir seçici belirtiyorsa:

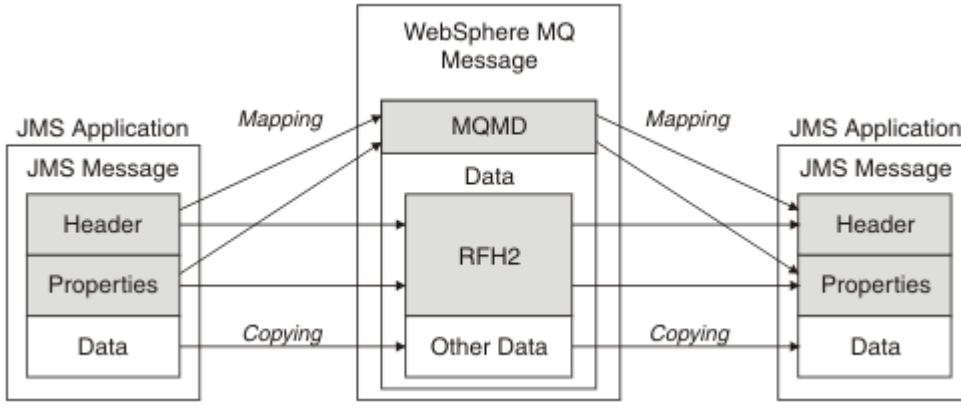
```
JMSCorrelationID = 'ID:correlation_id'
```

Burada *correlation\_id* , standart bir IBM MQ ilinti tanıtıcısı içeren bir dizgidir; JMS için IBM MQ sınıfları, belirtilen ilinti tanıtıcısına sahip iletiyi kuyruktan almak için **MatchOption** MQMO\_MATCH\_CORIL\_ID değerini kullanır.

In the following example, a message selector is used to get a message that has a correlation identifier of 414D51207061756C745639314C545320846E5B5F25B1CC02:

```
JMSCorrelationID= ' ID:414D51207061756C745639314C545320846E5B5F25B1CC02 '
```





Şekil 10. İletilerin MQRFH2 üstbilgisi kullanılarak JMS ile IBM MQ arasında nasıl dönüştürülebileceğini

Yapılar iki şekilde dönüştürülebiliyor:

### Eşleme

MQMD 'nin JMS alanıyla eşdeğeri olan bir alanı içerdiği durumlarda, JMS alanı MQMD alanına eşlenir. Ek MQMD alanları JMS özellikleri olarak gösterilir; JMS uygulaması, JMS dışı bir uygulama ile iletişim kurarken bu alanları ayarlamaya ya da ayarlamaya gerek duyabilir.

### Kopyalama

MQMD eşdeğeri olmayan bir yerde, JMS üstbilgi alanı ya da özelliği, MQRFH2 içinde bir alan olarak dönüştürülebilecek, geçirilemez.

### MQRFH2 üstbilgisi ve JMS

Bu konu grubunda, ileti içeriğiyle ilişkili JMS sözel verileri taşıyan MQRFH Sürüm 2 üstbilgisi açıklanır. MQRFH Sürüm 2 üstbilgisi genişletilebilir olduğundan, JMS ile doğrudan ilişkili olmayan ek bilgiler de taşıyabilir. Ancak bu bölüm yalnızca JMStarafından kullanımını kapsar. Tam tanım için bkz. [MQRFH2 -Kural ve biçimleme üstbilgisi 2](#).

Üstbilginin iki bölümü, sabit bir kısmı ve değişken kısmı vardır.

### Sabit bölüm

Sabit bölüm, standart IBM MQ üstbilgi örüntünden modellenir ve aşağıdaki alanlardan oluşur:

#### StrucId (MQCHAR4)

Yapı tanıtıcısı.

MQRFH\_STRUC\_ID değeri olmalıdır (değer: "RFH ") (başlangıç değeri).

MQRFH\_STRUC\_ID\_ARRAY (değer: "R", "F", "H", " ") tanımlıdır.

#### Sürüm (MQUZE)

Yapı sürüm numarası.

MQRFH\_VERSION\_2 (değer: 2) (başlangıç değeri) olmalıdır.

#### StrucLength (MQUZE)

NameValueVeri alanları da içinde olmak üzere toplam MQRFH2 uzunluğu.

StrucLength olarak ayarlanan değer, birden çok 4 olmalıdır (bunu başarmak için NameValueVeri alanlarındaki veriler boşluk karakterleriyle doldurulabilir).

#### Kodlama (MQUZE)

Veri kodlaması.

Encoding of any numeric data in the portion of the message following the MQRFH2 (the next header, or the message data following this header).

#### CodedCharSetId (MQHOMER)

Kodlanmış karakter takımı tanıtıcısı.

Representation of any character data in the portion of the message following the MQRFH2 (the next header, or the message data following this header).

### **Biçim (MQCHAR8)**

Biçim adı.

Format name for the portion of the message following the MQRFH2.

### **İşaretler (MQUZE)**

Bayraklar.

MQRFH\_NO\_FLAGS = 0. İşaret ayarlanmadı.

### **NameValueCCSID (MQUZE)**

Bu üstbilgide bulunan NameValueVeri karakteri dizgilerine ilişkin kodlanmış karakter takımı tanıtıcısı (CCSID). NameValueVerileri, üstbilgide bulunan diğer karakter dizgilerinden farklı olan bir karakter kümesinde kodlanabilir (StrucID ve Format).

NameValueCCSID 'si 2 baytlık bir Unicode CCSID 'se (1200, 13488 ya da 17584), Unicode 'un bayt sırası MQRFH2'indeki sayısal alanların bayt sıralamasını aynıdır. (Örneğin, Sürüm, StrucLengthve NameValueCCSID ' nin kendisi.)

NameValueCCSID değeri aşağıdaki tablodan değerleri alır:

<i>Çizelge 15. NameValueCCSID alanı için olası değerler</i>	
<b>CCSID</b>	<b>Anlamı</b>
1200	UTF-16, en son Unicode sürümü destekleniyor
13488	UTF-16, Unicode sürüm 2.0 altkütmesi
17584	UTF-16, Unicode sürüm 3.0 altkütmesi (Euro simgesini içerir)
1208	UTF-8, en son Unicode sürümü destekleniyor

### **Değişken kısmı**

Değişken kısmı sabit porsiyondan sonra gelir. Değişken kısmı, değişken sayıda MQRFH2 klasörü içeriyor. Her klasör, bir değişken öge sayısı ya da özellik içerir. Klasörler grubuna ilişkin özellikler. JMS tarafından yaratılan 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, JMSObjectMessageya da boş değer olarak tanımlar.

mcd klasörü, MQRFH2'içeren bir JMS iletisinde her zaman bulunur.

Her zaman IBM Integration Bus' dan gönderilen MQRFH2 içeren bir iletide bulunur. Bir iletinin etki alanını, biçimini, tipini ve ileti kümesini açıklar.

<i>Çizelge 16. mcd özellik adı, eşanlamlı, veri tipi ve klasör</i>			
<b>Özellik eşanlamlısı</b>	<b>Özellik adı</b>	<b>Veri tipi</b>	<b>Klasör</b>
	mcd.Msd	string	<mcd><Msd>messageDomain</Msd></mcd>
	mcd.Set	string	<mcd><Set>messageDomain</Set></mcd>
	mcd.Type	string	<mcd><Type>messageDomain</Type></mcd>

Çizelge 16. mcd özellik adı, eşanlamlı, veri tipi ve klasör (devamı var)			
Özellik eşanlamlısı	Özellik adı	Veri tipi	Klasör
	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 ' de bulunur MQRFH2.

### usr klasörü

usr , iletiyle ilişkili uygulama tanımlı JMS özelliklerini içerir. usr klasörü, yalnızca bir uygulama uygulama tanımlı bir özellik ayarladıysa bulunur.

### mnext klasörü

mnext 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 17. mnext özellik adı, eşanlamlı, veri tipi ve klasör			
Özellik eşanlamlısı	Özellik adı	Veri tipi	Klasör
JMSArmCorrelator	mnext.Arm	string	<mnext><Arm>armCorrelator</Arm></mnext>
JMSRMCorrelator	mnext.Wrm	string	<mnext><Wrm>wrmCorrelator</Wrm></mnext>
JMSDeliveryTime	mnext.Dlt	i8	<mnext><Dlt>DeliveryTime</Dlt></mnext>
JMSDeliveryDelay	mnext.Dly	i8	<mnext><Dly>DeliveryTime</Dly></mnext>

mnext klasörüne kendi özelliklerinizi eklemeyin.

### mmps klasörü

mmps , yalnızca IBM MQ yayınlama/abone olma tarafından kullanılan özellikleri içerir. Klasör yalnızca uygulama, tümleşik yayınlama/abone olma özelliklerinden en az birini ayarladıysa bulunur.

Çizelge 18. mmps özellik adı, eşanlamlı, veri tipi ve klasör			
Özellik eşanlamlısı	Özellik adı	Veri tipi	Klasör
MQTopicString	mmps.Top	string	<mmps><Top>topicString</Top></mmps>
MQSubscriberData	mmps.Sud	string	<mmps><Sud>subscriberUserData...</Sud></mmps>
MQIsRetained	mmps.Ret	boolean	<mmps><Ret>isRetained</Ret></mmps>

Çizelge 18. mqps özellik adı, eşanlamlı, veri tipi ve klasör (devamı var)			
Özellik eşanlamlısı	Özellik adı	Veri tipi	Klasör
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>
MQPubStrIn tData	mqpse.Sid	string	<mqps><Sid>publicationData</Sid></mqps>
MQPubFormat	mqpse.Pfmt	i8	<mqps><Pfmt>messageFormat</Pfmt></mqps>

mqps klasörüne kendi özelliklerinizi eklemeyin.

Çizelge 19 sayfa 136 , özellik adlarının tam listesini gösterir.

Çizelge 19. JMStarafından kullanılanMQRFH2 klasörleri ve özellikleri				
JMS alan adı	Java tip	MQRFH2 klasör adı	Özellik adı	Tip/değerler
JMSHedef	Hedef	JMS	Dst	dizgi
JMSSüresi	long	JMS	ÜS	i8
JMSönceliği	int	JMS	Pri	i4
JMSDeliveryMode	int	JMS	Dlv	i4
JMSCorrelationID	Dizgi	JMS	Cid	dizgi
JMSReplyTo	Hedef	JMS	Rto	dizgi
JMSTimestamp	long	JMS	Tms	i8
JMSType	Dizgi	MCD	Tip, Küme, Fmt	dizgi
JMSXGroupID	Dizgi	JMS	GID	dizgi
JMSXGroupSeq	int	JMS	Seq	i4
xxx (kullanıcı tanımlı)	Herhangi	USR	xxx	Herhangi Biri
		MCD	MSD	jms_yok jms_metin jms_byte jms_map jms_akımı jms_nesnesi

#### NameValueUzunluk (MQUZE)

Bu uzunluk alanını hemen izleyen NameValueveri dizgisinin bayt cinsinden uzunluğu (kendi uzunluğunu kapsamaz).



## NameValueVerileri (MQCHARn)

Önceki NameValueUzunluk alanı tarafından bayt cinsinden uzunluğu verilen tek bir karakter dizisi. Bir özellik dizisini tutan bir klasör içerir. Her özellik, adı klasör adı olan bir XML öğesinin içinde yer alan bir ad/tip/değer üçlüdür:

```
<foldername>  
triplet1 triplet2 ..... tripletn </foldername>
```

Kapanış </foldername> etiketi, dolgu karakteri olarak boşluklarla izlenebilir. Her bir kırpma, 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. İçerilirse, dt= etiketinden önce bir ya da daha çok boşluk karakteri eklenmelidir.

### name

Özelliğin adı; bkz. [Çizelge 19 sayfa 136](#).

### datatype

katlama işleminden sonra, [Çizelge 20 sayfa 137](#) içinde listelenen veri tiplerinden biri eşleşmelidir.

### value

[Çizelge 20 sayfa 137](#) içindeki tanımlamaları kullanarak, aktarılabilecek değerlerin bir dizgi gösterimidir.

Boş değer, aşağıdaki sözdizimi kullanılarak kodlanır:

```
<name dt='datatype' xsi:nil='true'></name>
```

xsi:nil='false' kullanmayın.

Çizelge 20. Özellik veri tipleri	
Veri tipi	Tanımlama
dizgi	< ve & dışında herhangi bir karakter dizisi
boolean	0 ya da 1 karakteri ( 0 = false, 1 = true)
bin.hex	Sekizlileri temsil eden onaltılık basamaklar
i1	A number, expressed using digits 0 . . 9, with optional sign (no fractions or exponent). Dahil -128-127 aralığında (bu değerler de içinde olmak üzere) olmalıdır
i2	A number, expressed using digits 0 . . 9, with optional sign (no fractions or exponent). -32768 ile 32767 arasında (bu değerler de içinde olmak üzere) yalan olmalıdır
i4	A number, expressed using digits 0 . . 9, with optional sign (no fractions or exponent). -2147483648 ile 2147483647 (dahil) arasında yalan olmalıdır
i8	A number, expressed using digits 0 . . 9, with optional sign (no fractions or exponent). -9223372036854775808 ile 9223372036854775807 (dahil) aralığında yatmalı
int	A number, expressed using digits 0 . . 9, with optional sign (no fractions or exponent). i8 ile aynı aralıkta olmalıdır. Gönderenin belirli bir duyarlılığı özellik ile ilişkilendirmek istemiyorsa, i * tiplerinden birinin yerine kullanılabilir
r4	Kayar noktalı sayı, büyüklük < = 3.40282347E+38, > = 1.175E-37 , 0 . . 9 basamakları kullanılarak ifade edilir, isteğe bağlı işaret, isteğe bağlı kesirli basamaklar, isteğe bağlı üstel
r8	Kayar noktalı sayı, büyüklük < = 1.7976931348623E+308, > = 2.225E-307 , 0 . . 9 basamakları kullanılarak ifade edilir, isteğe bağlı işaret, isteğe bağlı kesirli basamaklar, isteğe bağlı üstel

Bir dizgi değeri boşluk içerebilir. Bir dizgi değerinde aşağıdaki kaçış dizilerini kullanmanız gerekir:

- & karakteri için&amp;
- < karakteri için&lt;

Aşağıdaki kaçış dizilerini kullanabilirsiniz, ancak bunlar gerekli değildir:

- > karakteri için&gt;
- ' karakteri için&apos;
- " karakteri için&quot;

*İlgili MQMD alanlarıyla birlikte JMS alanları ve özellikleri*

These tables show the MQMD fields equivalent to JMS header fields, JMS properties, and JMS provider-specific properties.

Çizelge 21 sayfa 138 , JMS üstbilgi alanlarını listeler ve Çizelge 22 sayfa 138 , doğrudan MQMD alanlarıyla eşlenen JMS özelliklerini listeler. Çizelge 23 sayfa 138 , sağlayıcıya özgü özellikleri ve eşlendikleri MQMD alanlarını listeler.

JMS üstbilgi alanı	Java tip	MQMD alanı	C tipi
JMSDeliveryMode	int	Kalıcılık	MQLONG
JMSSüresi	long	Son kullanma tarihi	MQLONG
JMSönceliği	int	Öncelik	MQLONG
JMSMessageID	Dizgi	MsgID	MQBYTE24
JMSTimestamp	long	PutDate PutTime	MQCHAR8 MQCHAR8
JMSCorrelationID	Dizgi	CorrelId	MQBYTE24

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

JMS sağlayıcıya özgü özellik	Java tip	MQMD alanı	C tipi
JMS_IBM_Report_Exception	int	Rapor	MQLONG
JMS_IBM_Report_Son Kullanma	int	Rapor	MQLONG
JMS_IBM_Report_COA	int	Rapor	MQLONG
JMS_IBM_Report_COD	int	Rapor	MQLONG

Çizelge 23. JMS sağlayıcısına özgü özellikler-MQMD alanlarıyla eşleniyor (devamı var)

JMS sağlayıcıya özgü özellik	Java tip	MQMD alanı	C tipi
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_Disard_Msg	int	Rapor	MQLONG
JMS_IBM_MsgType	int	MsgType	MQLONG
JMS_IBM_Geri	int	Geribildirim	MQLONG
JMS_IBM_Biçimi	Dizgi	Biçim "1" sayfa 139	MQCHAR8
JMS_IBM_PutApplTipi	int	PutApplTipi	MQLONG
JMS_IBM_kodlama	int	Kodlama	MQLONG
JMS_IBM_Character_Set	Dizgi	CodedCharacterSetId "2" sayfa 139	MQLONG
JMS_IBM_PutDate	Dizgi	PutDate	MQCHAR8
JMS_IBM_PutTime	Dizgi	PutTime	MQCHAR8
JMS_IBM_Last_Msg_In_Group	boolean	MsgFlags	MQLONG

**Not:**

1. JMS\_IBM\_Format, ileti gövdesinin biçimini temsil eder. This can be defined by the application setting the JMS\_IBM\_Format property of the message (note that there is an 8 character limit), or can default to the IBM MQ format of the message body appropriate to the JMS message type. JMS\_IBM\_Format, yalnızca ileti RFH ya da RFH2 bölümleri içermiyorsa MQMD Biçimi alanıyla eşlenir. Tipik bir iletide, ileti gövdesinden hemen önce gelen RFH2 ' nin biçim alanıyla eşlenir.
2. JMS\_IBM\_Character\_Set property value is a String value that contains the Java character set equivalent for the numeric CodedCharacterSetId value. MQMD alanı CodedCharacterSetId , JMS\_IBM\_Character\_Set özelliği tarafından belirtilen Java karakter kümesi dizgisinin eşdeğerini içeren sayısal bir değerdir.

*JMS alanlarını IBM MQ alanlarıyla eşleyerek (giden iletiler)*

Bu çizelgeler, JMS üstbilgi ve özellik alanlarının, gönderme () ya da yayınlama () sırasında MQMD ve MQRFH2 alanlarına nasıl eşlendiğini gösterir.

Çizelge 24 sayfa 140 , gönderme () ya da yayınlama () sırasında JMS üstbilgi alanlarının MQMD/RFH2 alanlarıyla nasıl eşlendiğini gösterir. Çizelge 25 sayfa 140 , gönderme () ya da yayınlama () sırasında JMS özelliklerinin MQMD/RFH2 alanlarına nasıl eşlendiğini gösterir. Çizelge 26 sayfa 141 , gönderme () ya da yayınlama () sırasında JMS sağlayıcıya özgü özelliklerin MQMD alanlarıyla nasıl eşlendiğini gösterir.

İleti nesnesine göre ayarlanmış olarak işaretlenmiş alanlar için, iletilen değer, gönderme () ya da yayınlama () işleminden hemen önce JMS iletisinde tutulan değerdir. The value in the JMS message is left unchanged by the operation.

Gönderme Yöntemine Göre Ayarla işaretli alanlar için, gönderme () ya da yayınlama () gerçekleştirildiğinde bir değer atanır ( JMS iletisinde tutulan herhangi bir değer yoksa). JMS iletindeki değer, kullanılan değeri göstermek üzere güncelleştirilir.

Alma olarak işaretlenen alanlar iletilmez ve gönderilerek () ya da yayınlama () ile değiştirilmeden bırakılır.

Çizelge 24. Giden ileti alanı eşlemesi			
JMS üstbilgi alanı adı	İletim için kullanılan MQMD alanı	Üstbilgi	Ayarlayan
JMSHedef		MQRFH2	Yöntemi Gönder
JMSDeliveryMode	Kalıcılık	MQRFH2	Yöntemi Gönder
JMSSüresi	Son kullanma tarihi	MQRFH2	Yöntemi Gönder
JMSönceliği	Öncelik	MQRFH2	Yöntemi Gönder
JMSMessageID	MsgID		Yöntemi Gönder
JMSTimestamp	PutDate/PutTime		Yöntemi Gönder
JMSCorrelationID	CorrelId	MQRFH2	İleti Nesnesi
JMSReplyTo	ReplyToQ/ReplyToQMgr	MQRFH2	İleti Nesnesi
JMSType		MQRFH2	İleti Nesnesi
JMSRedird			Yalnızca alma

**Not:**

1. MQMD alanı CodedCharacterSetId , JMS\_IBM\_Character\_Set özelliği tarafından belirtilen Java karakter kümesi dizgisinin eşdeğerini içeren sayısal bir değerdir.

Çizelge 25. Giden ileti JMS özellik eşlemesi			
JMS özellik adı	İletim için kullanılan MQMD alanı	Üstbilgi	Ayarlayan
JMSXUserID	UserIdentifier		Yöntemi Gönder
JMSXAppID	PutApplAdı		Yöntemi Gönder
JMSXDeliveryCount			Yalnızca alma
JMSXGroupID	GroupId	MQRFH2	İleti Nesnesi
JMSXGroupSeq	MsgSeqNumarası	MQRFH2	İleti Nesnesi

**Not:**

Bu özellikler JMS belirtimi tarafından salt okunur olarak tanımlanır ve JMS sağlayıcısı tarafından isteğe bağlı olarak ayarlanır (bazı durumlarda isteğe bağlı olarak).

IBM MQ classes for JMS ' ta bu özelliklerden ikisi uygulama tarafından geçersiz kılınabilir. Bunu yapmak için, hedefin aşağıdaki özellikleri ayarlayarak uygun şekilde yapılandırıldığından emin olun:

1. WMQConstants.WMQ\_MQMD\_MESSAGE\_CONTEXT özelliğini WMQConstants.WMQ\_MDCTX\_SET\_ALL\_CONTEXTolarak ayarlayın.
2. WMQConstants.WMQ\_MQMD\_WRITE\_ENABLED özelliğini trueolarak ayarlayın.

Aşağıdaki özellikler uygulama tarafından geçersiz kılınabilir:

**JMSXAppID**

İletideki WMQConstants.JMS\_IBM\_MQMD\_PUTAPPLNAME özelliği ayarlanarak bu özellik geçersiz kılınabilir; değer bir Java dizgisi olmalıdır.

**JMSXGroupID**

Bu özellik, iletide WMQConstants.JMS\_IBM\_MQMD\_GROUPID özelliği ayarlanarak geçersiz kılınabilir; bu değer bir byte dizisi olmalıdır.

Çizelge 26. 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_Son Kullanma	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_Disard_Msg	Rapor		İleti Nesnesi
JMS_IBM_MsgType	MsgType		İleti Nesnesi
JMS_IBM_Geri	Geribildirim		İleti Nesnesi
JMS_IBM_Biçimi	Biçim		İleti Nesnesi
JMS_IBM_PutApplTipi	PutApplTipi		Yöntemi Gönder
JMS_IBM_kodlama	Kodlama		İleti Nesnesi
JMS_IBM_Character_Set	CodedCharacterSetId		İleti Nesnesi
JMS_IBM_PutDate	PutDate		Yöntemi Gönder
JMS_IBM_PutTime	PutTime		Yöntemi Gönder
JMS_IBM_Last_Msg_In_Group	MsgFlags		İleti Nesnesi

Gönderme () ya da yayınlama () için JMS üstbilgisi alanları eşleniyor

Bu notlar, gönderme () ya da yayınlama () sırasında JMS alanlarının eşlemeleriyle ilgilidir.

#### JMSDestination- MQRFH2

Bu, hedef nesnenin belirgin özelliklerini diziselleştiren bir dizgi olarak saklanır; böylece, alıcı bir JMS , eşdeğer bir hedef nesneyi yeniden oluşturabilirler. MQRFH2 alanı URI olarak kodlanır (URI gösterimine ilişkin ayrıntılar için [“Birörnek kaynak tanıtıcıları \(URI ' ler\)” sayfa 199 ' a](#) bakın).

#### JMSReplyTo - MQMD.ReplyToQ, ReplyToQMgr, MQRFH2

Kuyruk adı MQMD.ReplyToQ alanı ve kuyruk yöneticisi adı ReplyToQMgr alanlarına kopyalanır. Hedef uzantı bilgileri (hedef nesnede saklanan diğer yararlı ayrıntılar), MQRFH2 alanına kopyalanır. MQRFH2 alanı URI olarak kodlanır (URI gösterimine ilişkin ayrıntılar için bkz. [“Birörnek kaynak tanıtıcıları \(URI ' ler\)” sayfa 199](#) ).

#### JMSDeliveryMode - MQMD.Persistence

The JMSDeliveryMode value is set by the send() or publish() Method or MessageProducer, unless the Destination Object overrides it. JMSDeliveryMode değeri, MQMD.Persistence alanı aşağıdaki gibi olur:

- JMS değeri PERSISTENT, MQPER\_PERDEAL ile eşdeğerdir
- JMS değeri NON\_PERSISTENT, MQPER\_NOT\_PERDUAL ile eşdeğerdir.

MQQueue persistence özelliği WMQConstants.WMQ\_PER\_QDEFolarak ayarlanmadıysa, teslim kipi değeri de MQRFH2içinde kodlanır.

#### JMSExpiration ile MQMD.Expiry, MQRFH2

JMSExpiration, süre bitimi (geçerli zamanın ve etkin zamanın toplamını) saklar, ancak MQMD, zaman zaman yaşamasını sağlar. Ayrıca, JMSExpiration milisaniyedir, ancak MQMD.Expiry saniyenin onda biri cinsinden ifade edilir.

- Gönderme () yöntemi, yaşamak için sınırsız bir zaman belirtiyorsa, MQMD.Expiry MQE\_UNSIINSıA olarak ayarlanır ve MQRFH2' da JMSExpiration kodlanmaz.

- If the send() method sets a time to live that is less than 214748364.7 seconds (about 7 years), the time to live is stored in MQMD.Expiry, and the expiration time (in milliseconds), is encoded as an i8 value in the MQRFH2.
- Gönderme () yöntemi bir saati 214748364.7 saniyeden daha uzun bir süre ayarlarsa, MQMD.Expiry MQEI\_UNESSINA olarak ayarlıdır. Milisaniye cinsinden gerçek süre bitimi, MQRFH2 içinde bir i8 değeri olarak kodlanır.

### **JMSPriority- MQMD.Priority**

JMSPriority değerini (0-9), MQMD öncelik değerine (0-9) doğrudan eşleyin. JMSPriority varsayılan olmayan bir değere ayarlandıysa, öncelik düzeyi MQRFH2 içinde de kodlanır.

### **MQMD.MessageID' danJMSMessageID**

JMS ' tan gönderilen tüm iletiler, IBM MQ tarafından atanan benzersiz ileti tanıtıcılarına sahiptir. Atanan değer, MQMD.MessageId alanı, MQPUT çağrısından sonra ve JMSMessageID alanında uygulamaya geri geçirilir. IBM MQ messageId , 24 baytlık bir ikili değerdir, ancak JMSMessageID bir dizgidir. JMSMessageID , ikili messageId değerinden 48 onaltılı karakterden oluşan bir sıraya dönüştürüldü; örnek olarak karakter tanıtıcısı: JMS , ileti tanıtıcıları üretmesini geçersiz kılmak için ayarlanabilen bir ipucu sağlar. Bu ipucu yok sayılır ve tüm durumlarda benzersiz bir tanıtıcı atanır. Bir gönderinin () üzerine yazılmadan önce JMSMessageID alanına ayarlanan herhangi bir değer üzerine yazılır.

Gerekliyse, MQMD.MessageID, bunu [“Reading and writing the message descriptor from an IBM MQ classes for JMS application” sayfa 216](#) içinde açıklanan IBM MQ JMS uzantılarından biriyle yapabilirsiniz.

### **JMSTimestamp- MQRFH2**

Bir gönderme sırasında, JMSTimestamp alanı JVM ' nin saatinde göre ayarlanır. Bu değer MQRFH2' ye ayarlanır. Bir gönderinin () üzerine yazılmadan önce JMSTimestamp alanına ayarlanan herhangi bir değer üzerine yazılır. Ayrıca, JMS\_IBM\_PutDate ve JMS\_IBM\_PutTime özelliklerine de bakın.

### **JMSType- MQRFH2**

Bu dizgi, MQRFH2 mcd.Type alanına ayarlanır. URI biçiminde olması durumunda, mcd.Set ve mcd.Fmt alanlarını da etkileyebilir.

### **JMSCorrelationID - MQMD.CorrelId, MQRFH2**

JMSCorrelationID aşağıdakilerden birini tutabilir:

#### **Sağlayıcıya özgü ileti tanıtıcısı**

This is a message identifier from a message previously sent or received, and so should be a string of 48 lowercase hexadecimal digits that are prefixed with ID: The prefix is removed, the remaining characters are converted into binary, and then they are set into the MQMD.CorrelId field. No CorrelId value is encoded in the MQRFH2.

#### **Sağlayıcı-yerel bayt [] değer**

The value is copied into the MQMD.CorrelId field - padded with nulls, or truncated to 24 bytes if necessary. No CorrelId value is encoded in the MQRFH2.

#### **Uygulamaya özgü bir dizgi**

Değer MQRFH2' ye kopyalanır. Dizginin ilk 24 baytı UTF8 biçiminde, MQMD.CorrelID.

### *Mapping JMS property fields*

Bu notlar, IBM MQ iletilerindeki JMS özellik alanlarının eşlenmesine başvurur.

### **JMSXUserID from MQMD UserIdentifier**

JMSXUserID , gönderme çağrısından geri dönüş olarak ayarlandı.

### **JMSXAppID from MQMD PutApplName**

JMSXAppID , gönderme çağrısından geri dönüş olarak ayarlandı.

### **JMSXGroupID - MQRFH2 (noktadan noktaya iletişim)**

Noktadan noktaya iletişim iletileri için, JMSXGroupID , MQMD GroupID alanına kopyalanır. JMSXGroupID örnek tanıtıcısı ile başlıyorsa, ikili olarak dönüştürülür. Ters durumda, bu dizgi bir UTF8 dizgisi olarak kodlanır. Gerekliyse, değer doldurulur ya da kesilir ve 24 byte uzunluğunda bir değer kesilir. MQMF\_MSG\_IN\_GROUP işareti ayarlıdır.

### **JMSXGroupID - MQRFH2 (yayınlama/abone olma)**

Yayınlama/abone olma iletileri için, JMSXGroupID bir dizgi olarak MQRFH2 içine kopyalanır.

### **JMSXGroupSeq MQMD MsgSeqSayı (noktadan noktaya iletişim)**

Noktadan noktaya iletişim iletileri için, JMSXGroupSeq , MQMD MsgSeqSayı alanına kopyalanır. MQMF\_MSG\_IN\_GROUP işareti ayarlıdır.

### **JMSXGroupSeq MQMD MsgSeqNumarası (yayınlama/abone olma)**

Yayınlama/abone olma iletileri için, JMSXGroupSeq , MQRFH2 içine 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 MsgType ile eşlenir. Uygulama, JMS\_IBM\_MsgType için belirttik 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, YYYYAAAGG ' nin IBM MQ Tarih biçimindeki sekiz karakterden oluşan bir dizidir. 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, HHMMSSSTH ' 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 yoksayılr.

**IBM MQ alanlarını JMS alanlarıyla eşleyerek (gelen iletiler)**

Bu çizelgeler, JMS üstbilgi ve özellik alanlarının, get () ya da alma () zamanındaki MQMD ve MQRFH2 alanlarına nasıl eşlendiğini gösterir.

Çizelge 27 sayfa 144 shows how JMS header fields are mapped onto MQMD/MQRFH2 fields at get() or receive() time. Çizelge 28 sayfa 145 shows how JMS property fields are mapped onto MQMD/MQRFH2 fields at get() or receive() time. Çizelge 29 sayfa 145 , JMS sağlayıcısına özgü özelliklerin nasıl eşlendiğini gösterir.

Çizelge 27. Gelen ileti JMS üstbilgi alanı eşlemesi		
JMS üstbilgi alanı adı	MQMD alanı alındı	MQRFH2 alanı şu alandan alındı:
JMSHedef		jms.Dst ya da mqps.Top "1" sayfa 145
JMSDeliveryMode	Kalıcılık "2" sayfa 145	jms.Dlv "2" sayfa 145
JMSSüresi		jms.Exp
JMSönceliği	Öncelik	
JMSMessageID	MsgID	
JMSTimestamp	PutDate "2" sayfa 145 PutTime "2" sayfa 145	jms.Tms "2" sayfa 145
JMSCorrelationID	CorrelId "2" sayfa 145	jms.Cid "2" sayfa 145



Çizelge 27. Gelen ileti JMS üstbilgi alanı eşlemesi (devamı var)

JMS üstbilgi alanı adı	MQMD alanı alındı	MQRFH2 alanı şu alandan alındı:
JMSReplyTo	ReplyToQ "2" sayfa 145 ReplyToQMGr "2" sayfa 145	jms.Rto "2" sayfa 145
JMSType		mcd.Type, mcd.Set, mcd.Fmt
JMSRedird	BackoutCount	

**Not:**

1. Hem jms.Dst hem de mqps.Top ayarlandıysa, jms.Dst içindeki değer kullanılır.
2. MQRFH2 'den ya da MQMD' den alınan değerlere sahip olan özellikler için, her ikisi de kullanılabilir durumda ise, MQRFH2 ' deki ayar kullanılır.
3. JMS\_IBM\_Character\_Set property value is a String value that contains the Java character set equivalent for the numeric CodedCharacterSetId value.

Çizelge 28. Gelen ileti özelliği eşlemesi

JMS özellik adı	MQMD alanı alındı	MQRFH2 alanı şu alandan alındı:
JMSXUserID	UserIdentifier	
JMSXAppID	PutApplAdı	
JMSXDeliveryCount	BackoutCount	
JMSXGroupID	GroupId "1" sayfa 145	jms.Gid "1" sayfa 145
JMSXGroupSeq	MsgSeqNumara "1" sayfa 145	jms.Seq "1" sayfa 145

**Not:**

1. MQRFH2 'den ya da MQMD' den alınan değerlere sahip olan özellikler için, her ikisi de kullanılabilir durumda ise, MQRFH2 ' deki ayar kullanılır. Bu özellikler yalnızca MQMF\_MSG\_IN\_GROUP ya da MQMF\_LAST\_MSG\_IN\_GROUP ileti işaretleri ayarlandıysa, MQMD değerlerinden ayarlanır.

Çizelge 29. Gelen ileti sağlayıcıya özgü JMS özellik eşlemesi

JMS özellik adı	MQMD alanı alındı	MQRFH2 alanı şu alandan alındı:
JMS_IBM_Report_Exception	Rapor	
JMS_IBM_Report_Son Kullanma	Rapor	
JMS_IBM_Report_COA	Rapor	
JMS_IBM_Report_COD	Rapor	
JMS_IBM_Report_PAN	Rapor	
JMS_IBM_Report_NAN	Rapor	
JMS_IBM_Report_Pass_MsgID	Rapor	
JMS_IBM_Report_Pass_Correl_ID	Rapor	
JMS_IBM_Report_Disard_Msg	Rapor	

Çizelge 29. Gelen ileti sağlayıcıya özgü JMS özellik eşlemesi (devamı var)

JMS özellik adı	MQMD alanı alındı	MQRFH2 alanı şu alandan alındı:
JMS_IBM_MsgType	MsgType	
JMS_IBM_Geri	Geribildirim	
JMS_IBM_Biçimi	Biçim	
JMS_IBM_PutApplTipi	PutApplTipi	
JMS_IBM_kodlama "1" sayfa 146	Kodlama	
JMS_IBM_Character_Set "1" sayfa 146	CodedCharacterSetId	
JMS_IBM_PutDate	PutDate	
JMS_IBM_PutTime	PutTime	
JMS_IBM_Last_Msg_In_Group	MsgFlags	

1. Yalnızca gelen ileti bir Bytes İletisi ise ayarlanır.

#### Exchanging messages between a JMS application and a traditional IBM MQ application

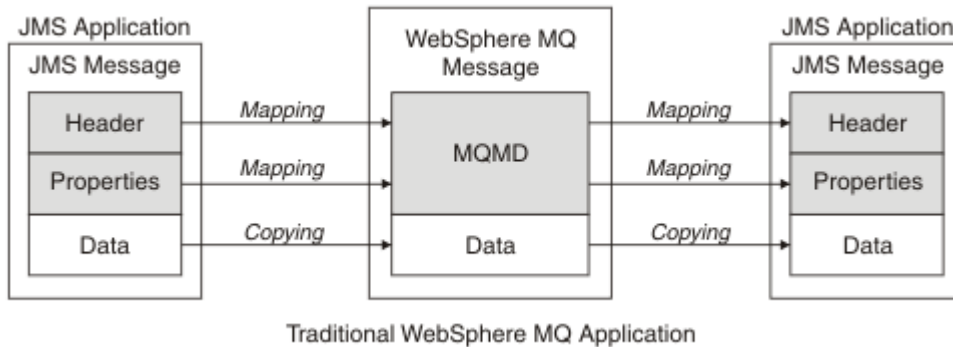
This topic describes what happens when a JMS application exchanges messages with a traditional IBM MQ application that cannot process the MQRFH2 header.

Şekil 11 sayfa 146 , eşlemeyi gösterir.

The administrator indicates that the JMS application is communicating with a traditional IBM MQ application by setting the TARGCLIENT property of the destination to MQ. Bu, MQRFH2 üstbilgisinin üretilmemesine neden olmadığını gösterir. Bu işlem yapılmazsa, alma uygulaması MQRFH2 üstbilgisini işleyebilmelidir.

The mapping from JMS to MQMD targeted at a traditional IBM MQ application is the same as mapping from JMS to MQMD targeted at a JMS application. If IBM MQ classes for JMS receives an IBM MQ message with the MQMD *Format* field set to anything other than MQFMT\_RFH2, data is being received from a non-JMS application. Biçim MQFMT\_STRING ise, ileti bir JMS metin iletisi olarak alınır. Ters durumda, bu ileti bir JMS byte iletisi olarak alınır. MQRFH2 olmadığı için, yalnızca MQMD ' de iletilen JMS özellikleri geri yüklenebilir.

IBM MQ classes for JMS , MQRFH2 üstbilgisi olmayan bir ileti alırsa, iletinin JMSReplyTo üstbilgi alanından türetilmiş olan Kuyruk ya da Konu nesnesinin TARGCLIENT özelliği varsayılan olarak MQ olarak ayarlanır. Bu, kuyruğa ya da konuya gönderilen bir yanıt iletisinin MQRFH2 üstbilgisine de sahip olmadığı anlamına gelir. Bir yanıt iletisinde MQRFH2 üstbilgisi dahil olmak üzere bu davranışı, özgün iletinin bir MQRFH2 üstbilgisi varsa, bağlantı üreticisinin TARGCLIENTMATCHESLE özelliğini NO olarak ayarlayarak değiştirebilirsiniz.



Şekil 11. JMS iletilerinin, MQRFH2 üstbilgisi içermeyen IBM MQ iletilerine nasıl dönüştürülebileceği

## JMS ileti gövdesi

Bu konu, ileti gövdesinin kodlamasıyla ilgili bilgileri içerir. Kodlama, JMS iletisinin tipine bağlıdır.

### ObjectMessage

ObjectMessage , normal şekilde Java Runtime tarafından diziselleştirilmiş bir nesnedir.

### TextMessage

TextMessage , kodlanmış bir dizgidir. Giden bir ileti için, dizgi hedef nesne tarafından verilen karakter kümesiyle kodlanır. Bu, varsayılan değer olarak UTF8 kodlamasına ayarlanır ( UTF8 kodlaması, iletinin ilk karakteriyle başlar; başlangıçta uzunluk alanı yoktur). Ancak, IBM MQ classes for JMS tarafından desteklenen başka herhangi bir karakter kümesini belirtmek mümkün olabilir. Bu tür karakter kümeleri daha çok JMS dışı bir uygulamaya ileti gönderdiğinizde kullanılır.

Karakter kümesi çift baytlık bir küme ( UTF16da içinde olmak üzere), hedef nesnenin tamsayı kodlama belirtimi byte 'ın sırasını belirler.

Gelen ileti, iletide belirtilen karakter kümesi ve kodlamayla yorumlanır. Bu belirtiler son IBM MQ üstbilgisinde (ya da üstbilgi yoksa MQMD) bulunur. JMS iletileri için, son üstbilgi genellikle MQRFH2' dir.

### BytesMessage

A BytesMessage is, by default, a sequence of bytes as defined by the JMS 1.0.2 specification and associated Java documentation.

Uygulamanın kendisi tarafından birleştirilen bir giden ileti için, hedef nesnenin kodlama özelliği, iletide yer alan tamsayı ve kayar noktalı alanların kodlamaları geçersiz kılmak için kullanılabilir. Örneğin, kayan noktalı değerlerin IEEE biçimi yerine S/390 içinde saklanabileceğini 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 (BytesMessage) alınırsa ve değiştirilmeden yeniden gönderilirse, gövdesi, alındığı için bayt for byte 'ı iletir. Hedef nesnenin kodlama özelliğinin gövde üzerinde etkisi yok. BytesMessage içinde açık olarak gönderilebilen tek dize benzeri varlık, bir UTF8 dizesidir. Bu, Java UTF8 biçiminde kodlanır ve 2 baytlık bir alanla başlar. Hedef nesnenin karakter kümesi özelliğinin giden bir BytesMessage öğesinin kodlaması üzerinde hiçbir etkisi yoktur. Gelen bir IBM MQ iletisinde karakter kümesi değeri, bu iletinin yorumlanmasında JMS BytesMessage olarak bir etkisi yoktur.

Java olmayan uygulamaların Java UTF8 kodlamasını tanıması beklenmez. Bu nedenle, bir JMS uygulamasının metin verileri içeren bir BytesMessage göndermesi için uygulamanın kendi dizelerini bayt dizilerine dönüştürmesi ve bu bayt dizilerini BytesMessage' a yazması gerekir.

### MapMessage

MapMessage , şu şekilde kodlanan XML ad/tip/değer üçüzleri içeren bir dizgidir:

```
<map>
  <elt name="elementname1" dt="datatype1">value1</elt>
  <elt name="elementname2" dt="datatype2">value2</elt>
  ...
</map>
```

Burada datatype , [Çizelge 20 sayfa 137](#) içinde listelenen veri tiplerinden biridir. Varsayılan veri türü string' dir ve bu nedenle dizgi öğeleri için dt=" string" özniteliği atanır.

Bir eşlem iletisinin gövesini oluşturan XML dizesini kodlamak ya da yorumlamak için kullanılan karakter kümesi, bir metin iletisi için geçerli olan kurallara göre belirlenir.

IBM MQ classes for JMS öncesi sürümleri, aşağıdaki biçimde bir eşlem iletisinin gövdeini kodladığından 5.3 ' dan önceki sürümlere göre kodlanır:

```
<map>
  <elementname1 dt="datatype1">value1</elementname1>
  <elementname2 dt="datatype2">value2</elementname2>
```

```
</map>
```

5.3 and later versions of IBM MQ classes for JMS can interpret either format, but versions of IBM MQ classes for JMS earlier than 5.3 cannot interpret the current format.

If an application needs to send map messages to another application that is using a version of IBM MQ classes for JMS earlier than 5.3, the sending application must call the connection factory method `setMapNameStyle(WMQConstants.WMQ_MAP_NAME_STYLE_COMPATIBLE)` to specify that the map messages are sent in the previous format. Varsayılan olarak, tüm eylem iletileri yürürlükteki biçimde gönderilir.

### StreamMessage

Bir `StreamMessage`, bir eyleme ileti gibidir, ancak öge adları olmadan:

```
<stream>
  <elt dt="datatype1">value1</elt>
  <elt dt="datatype2">value2</elt>
  ...
</stream>
```

Burada `datatype`, Çizelge 20 sayfa 137’inde listelenen veri tiplerinden biridir. Varsayılan veri türü `string`’dir ve bu nedenle dizgi öğeleri için `dt="string"` özniteliği atlanır.

`StreamMessage` gövdeyi oluşturan XML dizisini kodlamak ya da yorumlamak için kullanılan karakter kümesi, `TextMessage` için geçerli olan kurallardan sonra belirlenir.

`MQRFH2`.format alanı aşağıdaki gibi ayarlanır:

#### MQFMT\_NONE

`ObjectMessage`, `BytesMessage` için ya da gövdesi olmayan iletiler için.

#### MQFMT\_STRING

`TextMessage`, `StreamMessage` ya da `MapMessage` için.

### JMS ileti dönüştürme

İleti gönderirken ve alırken JMS ' ta ileti verisi dönüştürme işlemi gerçekleştirilir. IBM MQ , çoğu veri dönüştürme işlemi 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 tokacı değiş tokuş edildiğinde metin dönüştürülmektedir.

Daha karmaşık ileti alışverişleri yapmayı planlıyorsanız, aşağıdaki konular ilginizi çekmektedir. Karmaşık ileti alışverişi şunları içerir:

- Transferring non-text messages between an IBM MQ application and a JMS application.
- Metin verilerinin bayt biçiminde değiş tokuş edilmesine neden olur.
- Uygulamadaki metnin dönüştürülmesi.

### JMS İleti Verileri

Veri dönüştürme, iki JMS uygulaması arasında bile metin ve sayısal verileri uygulamalar arasında değiş tokuş etmek için gereklidir. Metin ve sayıların iç gösterimi kodlanmalıdır, bu nedenle bir iletilerde aktarılabilirler. Kodlama, sayıların ve metnin nasıl temsil edildiği hakkında bir karar sağlar. IBM MQ manages the encoding of text and numbers in JMS messages, except for `JMSObjectMessage`, see “`JMSObjectMessage`” sayfa 155. Üç ileti özniteliğini kullanır. Üç öznitelik şunlardır: `CodedCharacterSetId`, `Encoding`, ve `Format`.

These three message attributes are normally stored in the JMS header, `MQRFH2`, fields of a JMS message. İleti tipi, JMS ileti tipi yerine bir `MQise`, öznitelikler ileti tanımlayıcısında ( `MQMD`) saklanır. Öznitelikler, JMS ileti verilerini dönüştürmek için kullanılır. JMS ileti verileri, bir IBM MQ iletinin ileti verileri kısmında aktarılır.

## JMS İleti Özellikleri

JMS message properties, such as JMS\_IBM\_CHARACTER\_SET, are exchanged in the MQRFH2 header part of a JMS message, unless the message has been sent without an MQRFH2. Yalnızca JMSTextMessage ve JMSBytesMessage , MQRFH2 olmadan gönderilebilir. Bir JMS özelliği, ileti tanımlayıcısında IBM MQ ileti özelliği olarak saklanırsa, MQMD, MQMD dönüştürmesinin bir parçası olarak dönüştürülür. Bir JMS özelliği MQRFH2' de saklanırsa, MQRFH2 . NameValueCCSID tarafından belirtilen karakter kümesinde saklanır. Bir ileti gönderildiğinde ya da alındığında, ileti özellikleri JVM ' deki iç gösterimlerine ve bunların iç gösterimlerine dönüştürülmektedir. Dönüştürme, ileti tanımlayıcısı ya da MQRFH2 . NameValueCCSID karakter takımından ve karakter takımından olmalıdır. Sayısal veriler metne dönüştürülür.

## JMS ileti dönüştürme

Aşağıdaki konularda, dönüştürme gerektiren daha karmaşık iletiler değiş tokuş etmeyi planlıyorsanız, yararlı olan örnekler ve görevler yer alır.

### *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 MQ tarafından otomatik olarak gerçekleştirilir.

İleti dönüştürmeye nasıl yaklaşılabacağına 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 149](#)
2. [“Uygulama verilerini dönüştürme” sayfa 150](#)
3. [“Kuyruk yöneticisi veri dönüştürme” sayfa 151](#)
4. [“İleti kanalı veri dönüştürme” sayfa 151](#)

### **Dönüştürme nerede gerçekleştirilmelidir?**

[“İleti dönüştürmeye ilişkin bir yaklaşım seçilmesi: alıcı iyi” sayfa 151](#) bölümünde, "alıcının iyi yaptığı" olağan yaklaşımı açıklanmaktadır. "Günlük nesnesi" ' yi iyi yapar, JMS veri dönüştürmesi için de geçerlidir.

## JMS istemcisi veri dönüştürme

JMS istemci<sup>1</sup>Veri dönüştürme, bir hedefe gönderilirken JMS iletilerinde Java temel öğelerinin ve nesnelere bayta dönüştürülmesi ve alındığında yeniden dönüştürülmesi işlemidir. JMS istemcisi veri dönüştürme işlemi JMSMessage sınıflarının yöntemlerini kullanır. Yöntemler, [Çizelge 30 sayfa 152](#) içindeki 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.

<sup>1</sup> "JMS İstemci" , istemcide ya da bağ tanımlama kipinde çalışan JMS arabirimini gerçekleştiren IBM MQ classes for JMS ' yi ifade eder.

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 yoksayılar. 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 nesnelere doğrudan 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 istemci ileti dönüştürme ve kodlama](#)" sayfa 161.

## Uygulama verilerini dönüştürme

Bir JMS istemci uygulaması, `java.nio.charset.Charset` sınıfını kullanarak belirtik karakter verileri dönüştürmesi gerçekleştirebilir; [Şekil 14 sayfa 154](#) ve [Şekil 15 sayfa 154](#) içindeki 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<sup>2</sup>. 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 istemcisinin JMS 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 "MQSTR" olarak belirtmediyse, ileti `JMSBytesMessage` olarak oluşturulur. `JMSBytesMessage` içinde biçimlendirilmiş ileti verilerini almak için bir yöntem dizisi çağırmanız gerekir. Yöntemlerin, alanların iletiye yazıldığı sırayla çağırılması gerekir. Alanlar sayısalysa, 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 kayıtlı 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. "[Exchanging a formatted record with a non-JMS application](#)" sayfa 169.

<sup>2</sup> Bir özel durum: `writeUTF` kullanılarak yazılan veriler 2 baytlık bir alanla başlar

## Kuyruk yöneticisi veri dönüştürme

Bir JMS istemci programı bir ileti aldığı anda, kuyruk yöneticisi tarafından kod sayfası dönüştürme işlemi gerçekleştirilebilir. Dönüştürme, bir C programı için gerçekleştirilen dönüştürmeyle aynıdır. C programı MQGMO\_CONVERT ögesini MQGET GetMessageOptions değıştirge seçeneđi olarak ayarlar; bkz. [Şekil 13 sayfa 154](#). Bir kuyruk yöneticisi, ileti alan bir JMS istemci programı için dönüştürme gerçekleştirir; WMQ\_RECEIVE\_CONVERSION hedef özelliđi WMQ\_RECEIVE\_CONVERSION\_QMGR olarak ayarlanırsa, JMS istemci programı hedef özelliđi de ayarlayabilir; bkz. [Şekil 12 sayfa 151](#).

```
((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 alışverişı sırasında gelir. İletideki Format alanı tanımlıysa ve hedef karakter kümesi ya da kodlama ileten 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 gerçekleştirilmesi gerekmez. JMS dışı bir alan program, iletinin gerekli kod sayfasında ve kodlamada teslim edildiğinden emin olmak için dönüştürme çıkışına dayanır. Gönderen bir JMS istemcisi ve JMS olmayan bir alıcıyla, bu örnek IBM MQ için 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 üzerinde **CVTMQMDTA** olarak adlandırılır. Bkz. ["Exchanging a formatted record with a non-JMS application"](#) sayfa 169.

## İleti kanalı veri dönüştürme

IBM MQ Gönderen, Sunucu, Küme Alıcısı ve Küme Gönderen kanallarının bir ileti dönüştürme seçeneđi vardır, CONVERT. Bir iletinin içeriđi, isteđe bađlı 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 diđer dönüştürme biçimlerini kullanmak mümkün değilse kullanılır.

## İleti dönüştürmeye ilişkin bir yaklaşım seçilmesi: "alıcı iyi"

IBM MQ uygulama tasarımında kod dönüştürmeye ilişkin olađan 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. Örneđin, alıcı altyapıda dođru 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 nesnelere oluşturmak için yeniden dönüştürülür.

JMS istemci uygulamaları ile C gibi bir dilde yazılan uygulamalar arasındaki fark, Java ' in veri dönüştürmesi gerçekleştirilmesi 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 Windows üzerinde `windows-1252` ve UNIX üzerinde `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` ya da `JMSObjectMessage` ileti tiplerinden birini seçin. Yapılacak başka dönüştürme görevi yok.

Kayıt biçimi kullanan JMS 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 `JMSByteMessage` komutunu kullanın. Sayı ve bayt yazmak için `JMSByteMessage` yöntemlerini ve `Charset` sınıfı metni bayt dizilerine belirttik 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. ["Exchanging a formatted record with a non-JMS application"](#) sayfa 169. JMS dışı uygulamalarla değiş tokuş edilen iletileri dönüştürme örnekleri için.

## Örnekler

### İleti tipleri ve dönüştürme tipleri tablosu

Çizelge 30. İleti tipleri ve dönüştürme tipleri				
Dönüştürme tipi				
İleti tipi	Metin	Sayısal	Diğer	Yok
JMSObjectMessage				getObject setObject



Çizelge 30. İleti tipleri ve dönüştürme tipleri (devamı var)

İleti tipi	Dönüştürme tipi			
	Metin	Sayısal	Diğer	Yok
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

## C programından veri dönüştürme çağrılıyor

```
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 parçasığı

## JMSBytesMessage içinde metin gönderme ve alma

Şekil 14 sayfa 154 içindeki kod, BytesMessage içinde 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 154 içinde `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 istemci ileti dönüştürme ve kodlama

JMS istemci ileti dönüştürme ve kodlamasını yapmak için kullandığınız yöntemler, her dönüştürme tipinin kod örnekleriyle birlikte listelenir.

### Kuyruk yöneticisi veri dönüştürme

Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alan JMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

## İlgili görevler

### Exchanging a formatted record with a non-JMS application

Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları izleyin ve JMSBytesMessage kullanılarak JMS dışı bir uygulama ile ileti alışverişi yapabilen bir JMS istemci uygulaması. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değişimi, bir veri dönüştürme çıkışı çağrılmadan ya da çağrılmadan gerçekleşebilir.

## İlgili başvurular

### JMS ileti tipleri ve dönüştürme

İleti tipi seçimi, ileti dönüştürmeye yaklaşımınızı etkiler. The interaction of message conversion and message type is described for the JMS message types, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessage, and JMSBytesMessage.

### *JMS ileti tipleri ve dönüştürme*

İleti tipi seçimi, ileti dönüştürmeye yaklaşımınızı etkiler. The interaction of message conversion and message type is described for the JMS message types, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessage, and JMSBytesMessage.

## JMSObjectMessage

JMSObjectMessage , bir nesne ve başvuruda bulunduğu tüm nesnelere, JVM tarafından bayt akışına diziselleştirilmiş bir nesne içerir. Metin, UTF-8'a serileştirilmiş ve 65534 bayttan daha fazla olmayan dizgiler ya da karakter dizileriyle sınırlanır. An advantage of JMSObjectMessage is that applications are not involved in any data conversion issues as long as they use only the methods and attributes of the object. JMSObjectMessage , uygulama programcısı olmayan karmaşık nesnelere için, bir iletinin bir iletiye nasıl kodlanacağını göz önünde bulundurarak karmaşık nesnelere için veri dönüştürme olanağı sağlar. The disadvantage of using JMSObjectMessage is it can be exchanged only with other JMS applications. Diğer JMS ileti tiplerinden birini seçerek, JMS iletilerini JMS dışı uygulamalarla değiş tokuş etmek mümkündür.

“JMSObjectMessage gönderme ve alma” sayfa 157 , bir iletide değiştirilmekte olan bir String nesnesini gösterir.

Bir JMS istemci uygulaması, JMSObjectMessage ' u yalnızca JMStipi bir gövde içeren bir iletide alabilirler. Hedef, bir JMS stil gövdesi belirtmelidir.

## JMSTextMessage

JMSTextMessage , tek bir metin dizgisi içerir. When a text message is sent, the text Format is set to "MQSTR" , WMQConstants.MQFMT\_STRING. Metnin CodedCharacterSetId değeri, hedefi için tanımlanan kodlanmış karakter takımı tanıtıcısı olarak ayarlanır. Metin, IBM MQ tarafından CodedCharacterSetId ile kodlanır. CodedCharacterSetId ve Format alanları, ileti tanımlayıcısında (MQMD) ya da MQRFH2'daki JMS alanlarına ayarlanır. İleti bir WMQ\_MESSAGE\_BODY\_MQ ileti gövdesi stiline sahip olarak tanımlandıysa ya da gövde stili belirtilmediyse, ancak hedef hedef WMQ\_TARGET\_DEST\_MQ ise, ileti tanımlayıcı alanları ayarlanır. Otherwise the message has a JMS RFH2 and the fields are set in the fixed part of the MQRFH2.

Bir uygulama, bir hedef için tanımlanan kodlanmış karakter takımı tanıtıcısını geçersiz kılabilir.

JMS\_IBM\_XX\_ENCODE\_CASE\_ONE character\_set ileti özelliğini kodlanmış bir karakter kümesi tanıtıcısı olarak ayarlamalıdır; örneğin, “JMSTextmessage gönderme ve alma” sayfa 158 içindeki örneğe bakın.

JMS istemcisi, consumer.receive yöntemi kuyruk yöneticisi dönüşümünü çağırdığında isteğe bağlıdır. Queue manager conversion is enabled by setting the destination property WMQ\_RECEIVE\_CONVERSION to WMQ\_RECEIVE\_CONVERSION\_QMGR. Kuyruk yöneticisi, iletiyi JMS istemcisine aktarmadan önce ileti için belirtilen JMS\_IBM\_CHARACTER\_SET ' dan gelen metin iletisini dönüştürür. Hedef farklı bir WMQ\_RECEIVE\_CCSDolmadıkça, dönüştürülen iletinin karakter kümesi 1208(UTF-8) olarak ayarlanır. JMSTextMessage ' a gönderme yapan iletide CodedCharacterSetId , hedef karakter kümesi tanıtıcısı olarak güncellenir. The text is decoded from the target character set into Unicode by the getText method; see the example in “JMSTextmessage gönderme ve alma” sayfa 158.

Bir JMSTextMessage , JMS MQRFH2 üstbilgisi olmadan bir MQstili ileti gövdesinde gönderilebilir. Uygulama tarafından geçersiz kılınmadıkça, hedef özniteliklerin

değeri WMQ\_MESSAGE\_BODY ve WMQ\_TARGET\_DEST ileti gövdesi biçimini belirler.

The application can override the values set on the destination by calling `destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ)` or `destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ)`.

If you send a `JMSTextMessage` with an MQ style body by sending it to a destination with WMQ\_MESSAGE\_BODY set to WMQ\_MESSAGE\_BODY\_MQ, you cannot receive it as a `JMSTextMessage` from the same destination. All messages received from a destination with WMQ\_MESSAGE\_BODY set to WMQ\_MESSAGE\_BODY\_MQ are received as a `JMSBytesMessage`. İletiyi `JMSTextMessage` olarak almaya çalışırsanız, bir kural dışı durum oluşmasına neden olur: `ClassCastException: com.ibm.jms.JMSBytesMessage cannot be cast to javax.jms.TextMessage`.

**Not:** Bir `JMSBytesMessage` içindeki metin, JMS istemcisi tarafından dönüştürülmez. İstemci yalnızca iletideki metni bayt dizisi olarak alabilir. Kuyruk yöneticisi dönüşümü etkinleştirildiyse, metin kuyruk yöneticisi tarafından dönüştürülür, ancak JMS istemcisi bunu yine de `JMSBytesMessage` içinde bayt dizisi olarak almaktadır.

Bir `JMSTextMessage` değerinin MQ ya da JMS gövdesi stiliyle gönderilip gönderilmediğini denetlemek için WMQ\_TARGET\_DEST özelliğinin kullanılması genellikle daha iyi olur. Daha sonra, iletiyi WMQ\_TARGET\_DEST değerine ayarlanmış bir hedeften WMQ\_TARGET\_DEST\_MQ ya da WMQ\_TARGET\_DEST\_JMSolarak ayarlayabilirsiniz. WMQ\_TARGET\_DEST , alıcı üzerinde hiçbir etkiye sahip değildir.

## JMSMapMessage ve JMSStreamMessage

Bu iki JMS ileti tipi benzerdir. You can read and write primitive types to the messages using methods based on the `DataInputStream` and `DataOutputStream` interfaces; see [“İleti tipleri ve dönüştürme tipleri tablosu” sayfa 160](#). Ayrıntılar [“JMS istemci ileti dönüştürme ve kodlama” sayfa 161](#) içinde açıklanmıştır. Her temel öge etiketlenmiş; bkz. [“JMS ileti gövdesi” sayfa 147](#).

Sayısal veriler okunuyor ve XML metni olarak kodlanmış iletiye yazıldı. Hedef özelliğe hiçbir başvuru yapılmadı, JMS\_IBM\_ENCODING. Metin verileri, `JMSTextMessage` içindeki metinle aynı şekilde işlem görür. Örneğin, [Şekil 20 sayfa 158](#) içindeki örnek tarafından oluşturulan ileti içeriğine bakmanız gerekiyorsa, tüm ileti verileri, 37 karakter kümesi değeriyle gönderildiği için EBCDIC ' de olur.

Bir `JMSMapMessage` ya da `JMSStreamMessage` içine birden çok öge gönderebilirsiniz.

Verilerin tek tek öğelerini bir `JMSMapMessage`'den ya da bir `JMSStreamMessage`'den konumuna göre alabilirsiniz. İletide saklanan `CodedCharacterSetId` değeri kullanılarak bir alma ya da okuma yöntemi çağrıldığında, her öğenin kodu çözülür. Öğeyi almak için kullanılan yöntem, gönderilen tip için farklı bir tip döndürürse, tip dönüştürülür. Tip dönüştürülemezse, kural dışı durum oluşur. Ayrıntılar için [Sınıf JMSStreamMessage](#) başlıklı konuya bakın. [“Sending data in a JMSStreamMessage and JMSMapMessage” sayfa 158](#) içindeki örnek, tip dönüşümünü gösterir ve `JMSMapMessage` içeriğini sıradan dışarı çıkarır.

`JMSMapMessage` ve `JMSStreamMessage` için MQRFH2. format alanı, "MQSTR" olarak ayarlıdır. If the destination property WMQ\_RECEIVE\_CONVERSION is set to WMQ\_RECEIVE\_CONVERSION\_QMGR, the message data is converted by the queue manager before being sent to the JMS client. İletinin MQRFH2. `CodedCharacterSetId` değeri, hedefin WMQ\_RECEIVE\_CCSDID ' idir. The MQRFH2. Encoding is Native. If WMQ\_RECEIVE\_CONVERSION is WMQ\_RECEIVE\_CONVERSION\_CLIENT\_MSG the `CodedCharacterSetId` and Encoding of the MQRFH2 is the value set by the sender.

Bir JMS istemcisi uygulaması, yalnızca JMS stili gövdesi olan ve MQ stilinde bir gövde belirtmeyen bir hedefte `JMSMapMessage` ya da `JMSStreamMessage` ' i alabilir.

## JMSBytesMessage

Bir `JMSBytesMessage` birden çok ilkel tip içerebilir. You can read and write primitive types to the messages using methods based on the `DataInputStream` and `DataOutputStream` interfaces; see [“İleti tipleri ve dönüştürme tipleri tablosu” sayfa 160](#). Ayrıntılar [“JMS ileti tipleri ve dönüştürme” sayfa 155](#) içinde açıklanmıştır.

The encoding of numeric data in the message is controlled by the value of `JMS_IBM_ENCODING` that is set before writing numeric data to the `JMSBytesMessage`. Bir uygulama, `JMS_IBM_ENCODING`ileti özelliğini ayarlayarak `JMSBytesMessage` için tanımlanan varsayılan `Native` kodlamasını geçersiz kılabilir.

Text data can be read and written in UTF-8 using the `readUTF` and `writeUTF`, or in Unicode using the `readChar` and `writeChar` methods. `CodedCharacterSetId`' yi kullanan hiçbir yöntem yok. Diğer bir seçenek olarak, JMS istemcisi, `Charset` sınıfını kullanarak metni bayt olarak kodlayabilir ve byte olarak kodlayabilir. It transfers the bytes between the JVM and message without the IBM MQ classes for JMS performing any conversion; see [“JMSBytesMessage içinde metin gönderme ve alma” sayfa 158](#).

MQ uygulamasına gönderilen bir `JMSBytesMessage` , genellikle JMS MQRFH2 üstbilgisi olmadan bir MQstili ileti gövdesinde gönderilir. Bir JMS uygulamasına gönderilirse, ileti gövdesi biçimi tipik olarak JMS olur. Uygulama tarafından geçersiz kılınmadıkça, hedef özniteliklerin değeri `WMQ_MESSAGE_BODY` ve `WMQ_TARGET_DEST` ileti gövdesi biçimini belirler. The application can override the values set on the destination by calling `destination.setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ)` or `destination.setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ)`.

Bir MQ stil gövdesi ile bir `JMSBytesMessage` gönderdiğinizde, iletiyi bir MQ ya da JMS ileti gövdesi stilini tanımlayan bir hedeften alabilirsiniz. JMS stili gövdesi içeren bir `JMSBytesMessage` gönderdiğinizde, iletiyi JMS ileti gövdesi stilini tanımlayan bir hedeften almanız gerekir. Bunu yapmazsanız, MQRFH2 , kullanıcı ileti verilerinin bir parçası olarak kabul edilir; bu, beklediğiniz gibi olmayabilir.

İletin bir MQ ya da JMS gövdesi biçimi olup olmadığı, `WMQ_TARGET_DEST`ayarından etkilenmez.

İleti daha sonra, kuyruk yöneticisi tarafından, ileti verileri için bir `Format` sağlandıysa ve kuyruk yöneticisi veri dönüştürme etkinleştirilmişse, ileti daha sonra dönüştürülebilirdi. İleti verilerinin biçimini belirtmekten başka bir şey için biçim alanını kullanmayın ya da boş bırakın, `MQConstants.MQFMT_NONE`

Bir `JMSBytesMessage`' ta birden çok öge gönderebilirsiniz. İleti için tanımlanan kodlamayı kullanarak ileti gönderildiğinde, her sayısal öge dönüştürülür.

You can retrieve the individual items of data from `JMSBytesMessage`. İletiyi yaratmak için yazma yöntemleriyle aynı sırada okuma yöntemlerini çağırın. İletide saklanan `Encoding` değeri kullanılarak ileti çağırıldığında, her sayısal öge dönüştürülür.

`JMSMapMessage` ve `JMSStreamMessage`' ten farklı olarak, `JMSBytesMessage` yalnızca uygulama tarafından yazılan verileri içerir. Bir `JMSMapMessage` ve `JMSStreamMessage` içindeki öğeleri tanımlamak için kullanılan XML etiketleri gibi ileti verilerinde ek veri saklanmaz. Bu nedenle, diğer uygulamalar için biçimlenmiş iletileri aktarmak için `JMSBytesMessage` simgesini kullanın.

Converting between `JMSBytesMessage` and `DataInputStream` and `DataOutputStream` is useful in some applications. Code based on the example, [“DataInputStream ve DataOutputStream kullanarak ileti okuma ve yazma” sayfa 159](#), is necessary to use the `com.ibm.mq.header` package with JMS.

## Örnekler

### JMSObjectMessagegönderme 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

## JMSTextmessagegönderme ve alma

Metin iletisi, farklı karakter kümelerinde metin içeremez. Örnekte, farklı karakter kümelerindeki metin, iki farklı ileti halinde gönderilir.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Şekil 17. Metin iletisini hedef tarafından tanımlanan karakter kümesiyle 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. ccşid 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

## Sending data in a JMSStreamMessage and JMSMapMessage

```
StreamMessage smo = session.createStreamMessage();
smo.writeString("256");
smo.writeInt(512);
smo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(smo);
...
MapMessage mmo = session.createMapMessage();
mmo.setString("First", "256");
mmo.setInt("Second", 512);
mmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
producer.send(mmo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println("Stream: First as float " + smi.readFloat() +
    " Second as String " + smi.readString());
...
Stream: First as float: 256.0, Second as String: 512
...
MapMessage mmi = (MapMessage)consumer.receive();
System.out.println("Map: Second as String " + mmi.getString("Second") +
    " First as double " + mmi.getDouble("First"));
...
Map: Second as String: 512, First as double: 256.0
```

Şekil 20. Send data in JMSStreamMessage and JMSMapMessage

## JMSByteMessage içinde metin gönderme ve alma

Şekil 21 sayfa 159 içindeki kod, ByteMessage içinde 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 159 içinde `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 ileti okuma ve yazma

Şekil 23 sayfa 159 içindeki kod, `DataOutputStream` kullanarak 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 bir `JMSBytesMessage` gönderin

`JMS_IBM_ENCODING` özelliğini belirleyen deyim açıklama satırı olarak geçersiz kılınmaktadır. Bu deyim, doğrudan bir `JMSBytesMessage`'a yazıyorsa, ancak `DataOutputStream`'a yazarken herhangi bir etkisi yoktur. `DataOutputStream`'a yazılan sayılar Native kodlamasında kodlanır. `JMS_IBM_ENCODING` ayarının bir etkisi yok.

Şekil 24 sayfa 160 içindeki kod, `DataStream` kullanan 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 kod sayfası özelliği ( JMS\_IBM\_CHARACTER\_SET) kullanılarak yazdırılır. Giriş JMS\_IBM\_CHARACTER\_SET , bir Java kod sayfasıdır ve sayısal kodlanmış karakter takımı tanıtıcısı değil.

### İleti tipleri ve dönüştürme tipleri tablosu

Çizelge 31. İleti tipleri ve dönüştürme tipleri				
İleti tipi	Dönüştürme 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 31. İleti tipleri ve dönüştürme tipleri (devamı var)

İleti tipi	Dönüştürme 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 MQ tarafından otomatik olarak gerçekleştirilir.

JMS istemci ileti dönüştürme ve kodlama

JMS istemci ileti dönüştürme ve kodlamasını yapmak için kullandığınız yöntemler, her dönüştürme tipinin kod örnekleriyle birlikte listelenir.

Kuyruk yöneticisi veri dönüştürme

Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alan JMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

### İlgili görevler

Exchanging a formatted record with a non-JMS application

Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları izleyin ve JMSBytesMessage kullanılarak JMS dışı bir uygulama ile ileti alışverişi yapabilen bir JMS istemci uygulaması. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değişimi, bir veri dönüştürme çıkışı çağrılmadan ya da çağrılmadan gerçekleşebilir.

*JMS istemci ileti dönüştürme ve kodlama*

JMS istemci ileti dönüştürme ve kodlamasını yapmak için kullandığınız yöntemler, her dönüştürme tipinin kod örnekleriyle birlikte listelenir.

Dönüştürme ve kodlama, Java primitives ya da objects are read ya da written to ve from JMS iletileri olduğunda ortaya çıkar. Dönüştürme, kuyruk yöneticisi veri dönüştürme ve uygulama verileri dönüşümünden ayırt edilmesi için JMS istemcisi veri dönüştürme adı olarak adlandırılır. The conversion

takes place strictly when data is read from or written to a JMS message. Metin, iç 16 bit Unicode gösterimine ya da iç 16 bit 'e dönüştürülür.<sup>3</sup>İletilerde metin için kullanılan karakter kümesine. Sayısal veriler, ileti için tanımlanan kodlamaya dönüştürülmüş ve Java ilkel sayısal tiplere dönüştürülür. Dönüştürmenin gerçekleştirilip gerçekleştirilmeyeceğini ve hangi dönüştürme tipinin gerçekleştirileceğini, JMS ileti tipine ve okuma ya da yazma işlemine bağlıdır.

Çizelge 32 sayfa 162 , farklı JMS ileti tiplerine ilişkin okuma ve yazma yöntemlerini, gerçekleştirilen dönüştürme tipine göre kategorilere ayırır. Dönüştürme tipleri, çizelgenin ardından gelen metinde açıklanır.

<i>Çizelge 32. İleti tipleri ve dönüştürme tipleri</i>				
	<b>Dönüştürme tipi</b>			
<b>İleti tipi</b>	<b>Metin</b>	<b>Sayısal</b>	<b>Diğer</b>	<b>Yok</b>
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

<sup>3</sup> Bazı Unicode gösterimi 16 bitten fazlasını gerektirir. Bir Java SE başvurusuna bakın.

Çizelge 32. İleti tipleri ve dönüştürme tipleri (devamı var)

İleti tipi	Dönüştürme tipi			
	Metin	Sayısal	Diğer	Yok
JMSMapMessage	getString setString	getDouble getFloat getInt getLong getShort setDouble setFloat setInt setLong setShort	getBoolean getObject setBoolean setObject	getBytes readChar setByte setBytes setChar

### Metin

Hedef için varsayılan CodedCharacterSetId , 1208, UTF-8' dir. Varsayılan olarak, metin Unicode 'dan dönüştürülür ve bir UTF-8 metin dizisi olarak gönderilir. Alma işlemi sırasında metin, istemci tarafından alınan iletteki kodlanmış karakter kümesinden Unicode 'a dönüştürülür.

setText ve writeString yöntemleri, metni Unicode 'dan hedef için tanımlanan karakter kümesinden dönüştürür. Bir uygulama, JMS\_IBM\_CHAREACTER\_SETileti özelliğini ayarlayarak hedef karakter kümesini geçersiz kılabilir. JMS\_IBM\_XX\_ENCODE\_CASE\_ONE charter\_set, bir ileti gönderirken, sayısal kodlanmış karakter takımı tanıtıcısı olmalıdır<sup>4</sup>.

“JMSTextmessagegönderme ve alma” sayfa 165 içindeki kod parçacıklar iki ileti gönderir. Biri hedef için tanımlanan karakter kümesinde, diğeri de uygulama tarafından tanımlanan 37 karakter kümesiyle gönderilir.

getText ve readString yöntemleri, iletteki metni, iletide tanımlı karakter kümesinden Unicode 'a dönüştürür. The methods use the code page defined in the message property, JMS\_IBM\_XX\_ENCODE\_CASE\_ONE charter\_set. İleti, bir MQtipi iletirse ve MQRFH2olmadığında, kod sayfası MQRFH2. CodedCharacterSetId ' tan eşlenir. İleti bir MQtipi iletirse, MQRFH2değeri yoksa, kod sayfası MQMD. CodedCharacterSetId' tan eşlenir.

Şekil 29 sayfa 166 içindeki kod parçacığı, hedefe gönderilen iletiyi alır. İletideki metin IBM037 kod sayfasından Unicode 'a geri dönüştürüldü.

**Not:** Metnin kodlanmış karakter takımı 37 'ye dönüştürülmesini denetmenin basit bir yolu, IBM MQ Explorer 'in kullanılmasıdır. Kuyruğa göz atın ve ileti alınmadan önce iletinin özelliklerini gösterin.

Contrast the code snippet in Şekil 28 sayfa 165 with the incorrect code snippet in Şekil 25 sayfa 163. In the incorrect snippet the text string is converted twice, once by the application, and again by IBM MQ.

```
TextMessage tmo = session.createTextMessage();
tmo.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, 37);
tmo.setText(new String("Sent in EBCDIC character set 37".getBytes(CCSID.getCodepage(37))));
producer.send(tmo);
```

Şekil 25. Kod sayfası dönüşümü yanlış

writeUTF yöntemi, metni Unicode 'dan 1208 'e, UTF-8' a dönüştürür. Metin dizilimi, 2 baytlık bir uzunlukla önceden ortaya çıktı. Metin dizilimin uzunluk üst sınırı 65534 byte 'tır. readUTF yöntemi,

<sup>4</sup> JMS\_IBM\_CHARACTER\_SET iletisi alınırken, Java CharSet kod sayfası adı yer almaktadır.

writeUTF yöntemi tarafından yazılan bir iletide bir öğeyi okur. Bu, writeUTF yöntemi tarafından yazılan bayt sayısını tam olarak okur.

## Sayısal

Bir hedefe ilişkin varsayılan sayısal kodlama: Native. Java için Native kodlama sabiti, tüm platformlar için aynı olan 273, x '00000111' değerine sahiptir. Alma sırasında, iletteki sayılar doğru biçimde sayısal Java primitiflere dönüştürülür. Dönüştürme, iletide tanımlı olan kodlamayı ve okuma yönteminin döndürdüğü tipi kullanır.

The send method converts numbers that are added to a message by the set and write into the numeric encoding defined for the destination. Hedef kodlama, bir uygulama tarafından ileti özelliği JMS\_IBM\_ENCODING ; bir ileti için geçersiz kılınabilir. örneğin:

```
message.setIntProperty(WMQConstants.JMS_IBM_ENCODING,  
WMQConstants.WMQ_ENCODING_INTEGER_REVERSED);
```

get ve read sayısal yöntemleri, iletteki sayıları, iletide tanımlanan sayısal kodlamadaki değerleri dönüştürür. Sayıları, read ya da get yöntemi tarafından belirtilen tipe dönüştürüyorlar; bkz. ENCODING özelliği. Yöntemler, JMS\_IBM\_ENCODING içinde tanımlanan kodlamayı kullanır. Bu kodlama MQRFH2. Encoding 'den eşlenir; ancak, ileti bir MQ-type iletisi değilse ve MQRFH2' den (.) sahip değildir. İleti bir MQ tipi iletisiyse, MQRFH2 değeri yoksa, yöntemler MQMD. Encoding içinde tanımlanan kodlamayı kullanır.

Şekil 30 sayfa 166 içindeki örnek, bir uygulamanın hedef biçimdeki bir sayıyı kodlamasını ve bir JMSStreamMessage' a göndermesini gösterir. Şekil 30 sayfa 166 içindeki örneği, Şekil 31 sayfa 166 içindeki örneğe göre karşılaştırın. Fark, JMS\_IBM\_ENCODING ' un JMSBytesMessage olarak ayarlanması gerekir.

**Not:** Sayının doğru olarak kodlandığından emin olmak için basit bir yöntem, IBM MQ Explorer 'ı kullanmanın bir yoludur. Kuyruğa göz atın ve tüketmeden önce iletinin özelliklerini gösterin.

## Diğer

The boolean methods encode true and false as x '01' and x '00' in a JMSByteMessage, JMSStreamMessage, and JMSMapMessage.

UTF yöntemleri, Unicode 'u UTF-8 metin dizilerine kodlayıp kodu çözer. Dizilimler, 65536 karakterden kısa ve 2 byte 'lık uzunluk alanlarından önce gelir.

Nesne yöntemleri, ilkel tipleri nesne olarak tamamlar. Sayısal ve metin tipleri, temel tipler, sayısal ve metin yöntemleri kullanılarak okunmuyorsa ya da yazılmışsa, kodlanır ya da dönüştürülür.

## Yok

readByte, readBytes, readUnsignedByte, writeByte ve writeBytes yöntemleri, uygulama ile ileti arasında dönüştürme yapılmadan tek baytlar ya da bayt dizilerini alır ya da kokardır. readChar ve writeChar yöntemleri, uygulama ile ileti arasında dönüştürme yapılmadan 2 byte Unicode karakter alır ve yerleştirir.

Using the readBytes and writeBytes methods, the application can perform its own code point conversion, as in “JMSBytesMessage içinde metin gönderme ve alma” sayfa 166.

IBM MQ does not perform any code page conversion in the client as the message is a JMSBytesMessage, and because the readBytes and writeBytes methods are used. Bununla birlikte, bayt metni gösteriyorsa, uygulama tarafından kullanılan kod sayfasının, hedefin kodlanmış karakter takısıyla eşleştiğinden emin olun. İleti, kuyruk yöneticisi dönüştürme çıkışı tarafından yeniden dönüştürülebilirdi. Another possibility is that the receiving JMS client program might follow the convention of converting any byte arrays representing text in the message into strings or characters using the JMS\_IBM\_CHARACTER\_SET property in the message.

Bu örnekte istemci, dönüştürmesi 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 seçmiş olabilir ve sonra iletinin JMS\_IBM\_XX\_ENCODE\_CASE\_ONE character\_set özelliğinde karşılık gelen kodlanmış karakter kümesini ayarlayabilir. The IBM MQ classes for Java use JMS\_IBM\_XX\_ENCODE\_CASE\_ONE character\_set to set the CodedCharacterSetId field in the JMS properties in the MQRFH2, or in the message descriptor, MQMD:

```
String codePage = CCSID.getCodepage(37);
message.setIntProperty(WMQConstants.JMS_IBM_CHARACTER_SET, codePage);
```

If a byte array is written into a JMSStringMessage or JMSMapMessage, IBM MQ classes for JMS does not perform data conversion, as the bytes are typed as hexadecimal data not as text in the JMSStringMessage and JMSMapMessage.

Uygulamanızdaki karakterler byte 'ları gösteriyorsa, okunabilmek ve iletiye yazmak için kod noktalarını dikkate almalısınız. Şekil 26 sayfa 165 içindeki kod, hedef kodlanmış karakter kümesini kullanma kuralından sonra gelir. JVM 'yi JVM için varsayılan karakter kümesini kullanarak oluşturursanız, bayt içeriği platforma bağlıdır. A JVM on Windows typically has a default CharSet of windows-1252, and UNIX, UTF-8. Windows ile UNIX arasındaki değişim, metin alışverişi için bayt olarak belirttik bir kod sayfası seçmenizi gerektirir.

```
StreamMessage smo = producer.session.createStreamMessage();
smo.writeBytes("123".getBytes(CCSID.getCodepage(((MQDestination) destination)
.getIntProperty(WMQConstants.WMQ_CCSID))));
```

Şekil 26. Hedef karakter takımı kullanılarak JMSStreamMessage içindeki bir dizeyi gösteren byte yazılması

## Örnekler

### JMSTextmessagegönderme ve alma

Metin iletisi, farklı karakter kümelerinde metin içeremez. Örnekte, farklı karakter kümelerindeki metin, iki farklı ileti halinde gönderilir.

```
TextMessage tmo = session.createTextMessage();
tmo.setText("Sent in the character set defined for the destination");
producer.send(tmo);
```

Şekil 27. Metin iletisini hedef tarafından tanımlanan karakter kümesiyle 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

<sup>5</sup> 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 hedefe ilişkin tanımlar. Bir JMSBytesMessage 'nin JMS\_IBM\_ENCODING özelliğini, hedef için belirtilen değere ayarlamamış olmanız gerektiğini fark edin.

---

```
StreamMessage smo = session.createStreamMessage();
smo.writeInt(256);
producer.send(smo);
...
StreamMessage smi = (StreamMessage)consumer.receive();
System.out.println(smi.readInt());
...
256
```

Şekil 30. JMSStreamMessage içindeki hedef kodlamayı kullanarak bir sayı gönderme

---

```
BytesMessage bmo = session.createBytesMessage();
bmo.writeInt(256);
int encoding = ((MQDestination) (destination)).getIntProperty(
    (WMQConstants.WMQ_ENCODING)
    (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. JMSBytesMessage içindeki hedef kodlamayı kullanarak bir sayı gönderme

---

### JMSBytesMessage içinde metin gönderme ve alma

Şekil 32 sayfa 166 içindeki kod, BytesMessage içinde 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 167 içinde 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. 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 33. JMSBytesMessage olanağından String alınması

## İlgili kavramlar

### JMS ileti dönüştürme yaklaşımları

JMS uygulama tasarımcılarına bir dizi veri dönüştürme yaklaşımı aşıktır. 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 MQ tarafından otomatik olarak gerçekleştirilir.

### Kuyruk yöneticisi veri dönüştürme

Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alan JMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

## İlgili görevler

### Exchanging a formatted record with a non-JMS application

Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları izleyin ve JMSBytesMessage kullanılarak JMS dışı bir uygulama ile ileti alışverişi yapabilen bir JMS istemci uygulaması. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değişimi, bir veri dönüştürme çıkışı çağrılmadan ya da çağrılmadan gerçekleştirilebilir.

## İlgili başvurular

### JMS ileti tipleri ve dönüştürme

İleti tipi seçimi, ileti dönüştürmeye yaklaşımınızı etkiler. The interaction of message conversion and message type is described for the JMS message types, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessage, and JMSBytesMessage.

### Kuyruk yöneticisi veri dönüştürme

Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alan JMS 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, Encoding ve 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\_CONVERSION hedef özelliği WMQ\_RECEIVE\_CONVERSION\_QMGR ya da WMQ\_RECEIVE\_CONVERSION\_CLIENT\_MSG olarak 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-8 kodlamasından 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 nesnelere 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-8 biçiminde teslim edilmesiyle ilgili sorunlar ortaya çıktı.

JMS kodu, [Şekil 36 sayfa 169](#) içindeki 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 parçasığı

## 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 JMStreamMessage olarak 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 MQ tarafından otomatik olarak gerçekleştirilir.

### JMS istemci ileti dönüştürme ve kodlama

JMS istemci ileti dönüştürme ve kodlamasını yapmak için kullandığınız yöntemler, her dönüştürme tipinin kod örnekleriyle birlikte listelenir.

### “Veri dönüştürme çıkışı çağrılıyor” sayfa 1015

Veri dönüştürme çıkışı, bir MQGET çağrısının işlenmesi sırasında denetimi alan, kullanıcı tarafından yazılmış bir çıkıştır.

## İlgili görevler

### Exchanging a formatted record with a non-JMS application

Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları izleyin ve JMSBytesMessage kullanılarak JMS dışı bir uygulama ile ileti alışverişi yapabilen bir JMS istemci uygulaması. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değişimi, bir veri dönüştürme çıkışı çağrılmadan ya da çağrılmadan gerçekleşebilir.

## İlgili başvurular

### JMS ileti tipleri ve dönüştürme

İleti tipi seçimi, ileti dönüştürmeye yaklaşımınızı etkiler. The interaction of message conversion and message type is described for the JMS message types, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMStreamMessage, and JMSBytesMessage.

### *Exchanging a formatted record with a non-JMS application*

Bir veri dönüştürme çıkışı tasarlamak ve oluşturmak için bu görevde önerilen adımları izleyin ve JMSBytesMessage kullanılarak JMS dışı bir uygulama ile ileti alışverişi yapabilen bir JMS istemci

uygulaması. JMS dışı bir uygulamayla biçimlendirilmiş bir iletinin değişimi, bir veri dönüştürme çıkışı çağrılmadan ya da çağrılmadan gerçekleşebilir.

## Başlamadan önce

You might be able to design a simpler solution to exchanging messages with a non-JMS application using a `JMSTextMessage`. Bu görevdeki adımları takip etmeden önce bu olasılığı ortadan kaldırın.

## Bu görev hakkında

Bir JMS istemcisi, diğer JMS istemcileriyle değiş tokuş edilen JMS iletilerinin biçimlendirilmesi ayrıntılarında yer almıyorsa, bu istemci daha kolay yazılır. İleti tipi `JMSTextMessage`, `JMSMapMessage`, `JMSStreamMessage`ya da `JMSObjectMessage` olduğu sürece, IBM MQ iletinin biçimlendirilmesi ayrıntılarının sonrasına bakar. IBM MQ , farklı platformlardaki kod sayfalarındaki ve sayısal kodlamadaki farklılıklarla ilgilidir.

JMS dışı uygulamalarla ileti alışverişi yapmak için bu ileti tiplerini kullanabilirsiniz. Bunu yapmak için, bu iletilerin IBM MQ classes for JMS tarafından nasıl oluşturulacağını anlamamız gerekir. İletileri yorumlamak için JMS dışı bir uygulamayı değiştirebilirsiniz; bkz. [“Mapping JMS messages onto IBM MQ messages” sayfa 132.](#)

Bu ileti tiplerinden birini kullanmanın yararı, JMS istemci programlamasıdır. Bu programlama, ileti alışverişi yaptığı uygulama tipine bağlı değildir. Bir dezavantaj, başka bir program için bir değişiklik gerektirmesi ve diğer programı değiştiremeyebilirsiniz.

Alternatif bir yaklaşım, var olan ileti biçimleriyle başa çıkabilen bir JMS istemcisi uygulaması yazmadır. Sık sık varolan iletiler değişmez biçimdir ve biçimlenmemiş veri, metin ve sayılardan oluşan bir karışım içerir. Use the steps in this task, and the example JMS client in [“Writing classes to encapsulate a record layout in a JMSBytesMessage” sayfa 173](#), as a starting point for building a JMS client that can exchange formatted records with non-JMS applications.

## Yordam

1. Kayıt düzenini tanımlayın ya da önceden tanımlı IBM MQ üstbilgi sınıflarından birini kullanın.

Önceden tanımlı IBM MQ üstbilgilerini işlemek için [IBM MQ ileti üstbilgilerini işleme](#) başlıklı konuya bakın.

[Şekil 37 sayfa 171](#) , veri dönüştürme yardımcı programı tarafından işlenebilen, kullanıcı tanımlı, sabit uzunluklu kayıt düzenine bir örnektir.

2. Veri dönüştürme çıkışını yaratın.

Veri dönüştürme çıkışı yazmak için [Writing a data-conversion exit program](#) (Veri dönüştürme çıkış programı yazma) yönergelerini izleyin.

To try out the example in [“Writing classes to encapsulate a record layout in a JMSBytesMessage” sayfa 173](#), name the data conversion exit MYRECORD.

3. Kayıt yerleşim düzenini sarmak ve kayıt göndermek ve almak için Java sınıflarını yazın. Alabileceğiniz iki yaklaşım şunlardır:

- Bu okumaya bir sınıf yazın ve kaydı içeren `JMSBytesMessage` ' ı yazın; bkz. [“Writing classes to encapsulate a record layout in a JMSBytesMessage” sayfa 173.](#)
- Kaydın veri yapısını tanımlamak için `com.ibm.mq.header.Header` sınıfını genişleten bir sınıf yazın; bkz. [Yeni üstbilgi tipleri için sınıflar oluşturma.](#)

4. İletileri değiş tokuş etmek için hangi kodlanmış karakter takısının belirleneceğine karar verin.

Bkz. [İleti dönüştürmesine bir yaklaşım seçilmesi: Günlük nesnesi iyi duruma gelir.](#)

5. Configure the destination to exchange MQ-type messages, without a JMS MQRFH2 header.

Hem gönderme, hem de alma hedefi, MQ tipi iletileri değiş tokuş edecek şekilde yapılandırılmalıdır. Aynı hedefi her iki gönderme ve alma için de kullanabilirsiniz.

Uygulama, hedef ileti gövdesi özelliğini geçersiz kılabilir:

```
((MQDestination)destination).setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

“Writing classes to encapsulate a record layout in a JMSBytesMessage” sayfa 173 içindeki örnek, hedef ileti gövdesi özelliğini geçersiz kılar ve bir MQstili iletinin gönderilmesine olanak sağlar.

## 6. Test the solution with JMS and non-JMS applications

Veri dönüştürme çıkışını sınamak için kullanılabilecek yararlı araçlar şunlardır:

- amqsgetc0.c örnek programı, JMS istemcisi tarafından gönderilen bir iletinin alınmasını test etmek için kullanışlıdır. Şekil 38 sayfa 172 içinde, örnek üstbilgisini ( RECORD.h) kullanmak için önerilen değişiklikleri görün. Bu değişikliklerle amqsgetc0.c, örnek JMS istemcisi tarafından gönderilen bir iletiyi alır ( TryMyRecord.java;). bkz. “Writing classes to encapsulate a record layout in a JMSBytesMessage” sayfa 173.
- Örnek IBM MQ göz atma programı ( amqsbcbg0.c), ileti üstbilgisinin içeriğini, JMS üstbilgisini ( MQRFH2) ve ileti içeriğini incelemek için kullanışlıdır.
- The **rfhutl** program, previously available in SupportPac IH03, allows test messages to be captured and stored in files, and then used to drive Message Flows. Çıkış iletileri çeşitli biçimlerde de okunabilir ve görüntülenebilir. Bu biçimler, iki XML tipini ve bir COBOL copybook ile eşleşmeyi içerir. Veriler EBCDIC ya da ASCII ' de olabilir. İleti gönderilmeden önce iletiye bir RFH2 üstbilgisi eklenebilir.

Değiştirilen amqsgetc0.c örnek programını kullanarak ileti almaya çalışırsanız ve neden kodu 2080 ile bir hata alırsanız, iletinin MQRFH2 olup olmadığını denetleyin. Bu değişiklikler, iletinin MQRFH2no ' u belirten 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 bildirme

```

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);

```

- Modify the MQGET call to use 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 değiştirme,

#### 1. Başlangıç:

```

buffer[messlen] = '\0';          /* add terminator */
printf("message <%s>\n", buffer);

```

#### 2. Hedef:

```

/* buffer[messlen] = '\0';          add terminator */
printf("ccsid <%d>, flags <%d>, message <%32.32s>\n \0",
      md.CodedCharSetId, record.Flags, record.RecordData);

```

Şekil 38. Modify amqsget0.c

## 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 MQ tarafından otomatik olarak gerçekleştirilir.

JMS istemci ileti dönüştürme ve kodlama

JMS istemci ileti dönüştürme ve kodlamasını yapmak için kullandığınız yöntemler, her dönüştürme tipinin kod örnekleriyle birlikte listelenir.

Kuyruk yöneticisi veri dönüştürme

Kuyruk yöneticisi veri dönüşümü, JMS istemcilerinden ileti alan JMS dışı uygulamalar tarafından her zaman kullanılabilir. İleti alan JMS istemcileri, isteğe bağlı olan kuyruk yöneticisi veri dönüştürmesini de kullanır.

### **İlgili başvurular**

JMS ileti tipleri ve dönüştürme

İleti tipi seçimi, ileti dönüştürmeye yaklaşımınızı etkiler. The interaction of message conversion and message type is described for the JMS message types, JMSObjectMessage, JMSTextMessage, JMSMapMessage, JMSStreamMessage, and JMSBytesMessage.

Dönüştürme-çıkış kodu yaratmak için kullanılan yardımcı program

*Writing classes to encapsulate a record layout in a JMSBytesMessage*

Bu görevin amacı, bir JMSBytesMessage içindeki veri dönüştürmenin ve sabit kayıt yerleşim düzeninin nasıl birleştirileceğini, örneğin, nasıl birleştirileceğini keşfetmek için kullanılır. Bu görevde, bir JMSBytesMessage içinde örnek bir kayıt yapısı değiştirmek için bazı Java sınıfları oluşturursunuz. Diğer kayıt yapılarını değiştirmek için sınıf yazmak için örneği değiştirebilirsiniz.

A JMSBytesMessage is the best choice of JMS message type to exchange mixed data type records with non-JMS programs. It has no additional data inserted into the message body by the JMS provider. Bu nedenle, bir JMS istemci programı var olan bir IBM MQ programıyla çalışabiliyorsa, kullanılacak ileti tipinin en iyi seçimidir. Bir JMSBytesMessage kullanılması ana sınıma, diğer program tarafından beklenen kodlama ve karakter kümesiyle eşleşen bir değer sağlar. Bir çözüm, kaydı sarmalayan bir sınıf oluşturmaktan başka bir şey değildir. A class that encapsulates reading and writing a JMSBytesMessage, for a specific record type, makes it easier to send and receive fixed-format records in a JMS program. Arabirimin soyut bir sınıftaki soysal yönlerini yakalayarak, çözümün büyük bölümü farklı kayıt biçimleri için yeniden kullanılabilir. Soyut soysal sınıfı genişleten sınıflarda farklı kayıt biçimleri uygulanabilir.

Alternatif bir yaklaşım, `com.ibm.mq.headers.Header` sınıfını genişletebilmektedir. `Header` sınıfının, daha açıklayıcı bir şekilde kayıt biçimi oluşturmak için `addMQLONG` gibi yöntemleri vardır. `Header` sınıfının kullanılmasının bir dezavantajı ise, öznitelikler daha karmaşık bir yorumlayıcı arabirimi kullanır ve ayarlanıyor. Her iki yaklaşım da aynı sayıda uygulama kodunda sonuç elde eder.

Bir JMSBytesMessage , her bir kayıt aynı biçimi, kodlanmış karakter kümesini ve kodlamayı kullanmadığı sürece, bir iletide MQRFH2' un yanı sıra yalnızca tek bir biçimi sarsabilir. Bir JMSBytesMessage 'in biçim, kodlama ve karakter kümesi, MQRFH2' in ardından gelen tüm iletilerin özellikleridir. Örnek, JMSBytesMessage ' un yalnızca bir kullanıcı kaydı içerdiği varsayımına yazılır.

### **Başlamadan önce**

1. Beceri düzeyiniz: Java programlama ve JMS ile ilgili bilgi sahibi olmanız gerekir. Java geliştirme ortamını ayarlama hakkında herhangi bir yönerge sağlanmaz. It is advantageous to have written a program to exchange a JMSTextMessage, JMSStreamMessage, or JMSMapMessage. Daha sonra, JMSBytesMessage kullanarak bir ileti alışverişi arasındaki farkları görebilirsiniz.
2. Örnek, IBM WebSphere MQ 7.0 gerektirir.
3. Örnek, Eclipse çalışma ortamının Java perspektifi kullanılarak yaratılmıştı. JRE 6.0 ya da üstünü gerektirir. You can use the Java perspective in IBM MQ Explorer to develop and run the Java classes. Alternatif olarak, kendi Java geliştirme ortamınızı kullanın.
4. IBM MQ Explorer 'in kullanılması, test ortamını ayarlamayı ve hata ayıklamayı, komut satırı yardımcı programlarını kullanmaktan daha basitleştirir.

## Bu görev hakkında

İki sınıf oluşturma yoluyla yönlendirildiniz: RECORD ve MyRecord. Bu iki sınıf birlikte, sabit biçimli bir kaydı sarsalıyor. Öznitelikleri elde etmek ve ayarlamak için yöntemleri var. Get (alma) yöntemi, kaydı JMSBytesMessage 'den okur ve put yöntemi bir kaydı JMSBytesMessage' e yazar.

Görevin amacı, yeniden kullanabileceğiniz bir üretim kalitesi sınıfı yaratmamaktır. Kendi sınıflarınıza başlamak için görevdeki örnekleri kullanmayı tercih edebilirsiniz. The purpose of the task is to provide you with guidance notes, primarily about using character sets, formats, and encoding, when using a JMSBytesMessage. Sınıfları oluşturmadaki her adım, bazen gözden kaçırılan JMSBytesMessage' u kullanma yönlerinin açıklanandır ve açıklardır.

RECORD sınıfı soyut (abstract) ve bir kullanıcı kaydı için bazı ortak alanları tanımlar. Ortak alanlar, göz yakalayıcısına, sürüme ve uzunluk alanına sahip olmanın standart IBM MQ üstbilgi düzenine modellenir. Birçok IBM MQ üstbilgisinde bulunan kodlama, karakter kümesi ve biçim alanları atılır. Başka bir üstbilgi kullanıcı tanımlı biçimi izleyemez. RECORD sınıfını genişleten MyRecord sınıfı, kaydı ek kullanıcı alanlarıyla birlikte genişleterek gerçekleştirir. Sınıflar tarafından yaratılan bir JMSBytesMessage'değeri, kuyruk yöneticisi veri dönüştürme çıkışı tarafından işlenebilir.

“Örneği çalıştırmak için kullanılan sınıflar” sayfa 180 includes a full listing of RECORD and MyRecord. Ayrıca, RECORD ve MyRecord' yi sınamak için ek "iskeleler" sınıflarının listelemeleri de yer alır. Ek sınıflar şunlardır:

### TryMyRecord

RECORD ve MyRecord' yi test etmek için kullanılan ana program.

### EndPoint

Tek bir sınıftaki JMS bağlantısını, hedefini ve oturumunu kaplayan soyut bir sınıf. Arabirimi yalnızca RECORD ve MyRecord sınıflarının test edilmesi gereksinimlerini karşılar. Bu, JMS uygulamaları yazmak için oluşturulmuş bir tasarım örneği değildir.

**Not:** Endpoint sınıfı, bir hedef oluşturduktan sonra bu kod satırını içerir:

```
((MQDestination)destination).setReceiveConversion  
(WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR);
```

V7.0'da V7.0.1.5' de kuyruk yöneticisi dönüşümünü açmak gerekir. Varsayılan değer olarak devre dışı bırakılır. V7.0' da en çok V7.0.1.4 kuyruk yöneticisi dönüşümü varsayılan olarak etkindir ve bu kod satırı bir hataya neden olur.

### MyProducer ve MyConsumer

Classes that extend EndPoint, and create a MessageConsumer and MessageProducer, connected and ready to accept requests.

Tüm sınıflar bir JMSBytesMessage' de veri dönüştürmeyi nasıl kullanacağını anlamak için, tüm sınıfları oluşturabileceğiniz ve deneyebileceğiniz eksiksiz bir uygulama oluşturur.

## Yordam

1. Bir IBM MQ üstbilgisindeki standart alanları bir varsayılan oluşturucuyla sarmalamak için bir soyut sınıf yaratın. Daha sonra, üstbilgiyi gereksinimlerinize göre uyarlamak için sınıfı genişletiyorsunuz.

```
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. The attributes, structID to nextFormat, are listed in the order they are laid out in a standard IBM MQ message header.
  - b. The attributes, format, messageEncoding, and messageCharset, describe the header itself, and are not part of the header.
  - c. Kaydın kodlanmış karakter takımı tanıtıcısını ya da karakter kümesini saklanıp saklanmayacağınıza karar vermeniz gerekir. Java , karakter kümeleri ve IBM MQ iletileri, kodlanmış karakter takımı tanıtıcılarını kullanır. Örnek kod karakter kümelerini kullanır.
  - d. int is serialized to MQLONG by IBM MQ. MQLONG , 4 bayttır.
2. Özel öznitelikler için alıcıları ve ayarlayıcıları yaratın.
- a) Alıcıları yarat ya da oluşturun:

```

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) Ayarlayıcılar yarat 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. Create a constructor to create a RECORD instance from a JMSBytesMessage.

```

public RECORD(BytesMessage message) throws JMSEException, 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. The messageCharset and messageEncoding, are captured from the message properties, as they override the values set for the destination. format güncellenmedi. Örnek, hata denetimi yapmaz. Record(BytesMessage) oluşturucusu çağrılırsa, JMSBytesMessage 'in RECORD tipinde bir ileti olduğu varsayılır. The line "setStructID(new String(structID, getMessageCharset()))" sets the eye catcher.

- b. RECORD eşgörünümünde belirlenen varsayılan değerlerin güncellenmesi sırasında, bu yöntemi tamamlayan kodun satırlarında, satırdaki alanları diziselden geri çevirme işlemi gerçekleştirin.
4. Üstbilgi alanlarını bir JMSBytesMessage değerine yazmak için bir put yöntemi yaratın.

```
protected BytesMessage put(MyProducer myProducer) throws IOException,
    JMSEException, 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 encapsulates the JMS Connection, Destination, Session, and MessageProducer in a single class. MyConsumer, used later on, encapsulates the JMS Connection, Destination, Session, and MessageConsumer in a single class.
- b. Bir JMSBytesMessage için, kodlama Native' den başka bir kodsda, kodlamada kodlamanın ayarlanması gerekir. The destination encoding is copied to the message encoding attribute, JMS\_IBM\_CHARACTER\_SET, and saved as an attribute of the RECORD class.
- i) "setMessageEncoding(myProducer.getEncoding());", hedef kodlamayı almak için "((MQDestination) destination).getIntProperty(WMQConstants.WMQ\_ENCODING);" i çağırır.
- ii) "Bytes.setIntProperty(WMQConstants.JMS\_IBM\_ENCODING, getMessageEncoding());", ileti kodlamasını ayarlar.
- c. Metni bayt olarak dönüştürmek için kullanılan karakter takımı hedeften alınır ve RECORD sınıfının bir özneliği olarak kaydedilir. Bir JMSBytesMessage yazılırken IBM MQ classes for JMS tarafından kullanılmadığı için iletide bu ad belirlenmez.

"messageCharset = myProducer.getCharset();" çağrılar

```
public String getCharset() throws UnsupportedEncodingException,
    JMSEException {
    return CCSID.getCodepage(getCCSID());
}
```

It gets the Java character set from a coded character set identifier.

"CCSID.getCodepage(ccsid)" is in the package com.ibm.mq.headers. The ccsid is obtained from another method in MyProducer, which queries the destination:

```
public int getCCSID() throws JMSEException {
    return ((MQDestination) destination)
        .getIntProperty(WMQConstants.WMQ_CCSID);
}
```

- d. "myProducer.setMQClient(true);", istemci tipine ilişkin hedef ayarı geçersiz kılar ve bunu IBM MQ MQI client' a zorlar. Bir yönetim yapılandırma hatasını gizlediğinden, bu kod satırını atlamayı tercih edebilirsiniz.



"myProducer.setMQClient(true);" çağrılarını:

```
((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ); }  
if (!getMQDest()) setMQBody();
```

The code has the side-effect of setting the IBM MQ body style to unspecified, if it must override a setting of JMS.

**Not:**

IBM MQ classes for JMS , iletinin biçimini, kodlamasını ve karakter kümesi tanıttıcısını ileti tanımlayıcısına, MQMD'ye ya da JMS üstbilgisine, MQRFH2' a yazdır. Bu, iletinin bir IBM MQ biçemi gövdesi olup olmadığına bağlıdır. MQMD alanlarını el ile ayarlamayın.

İleti tanımlayıcı özelliklerini el ile ayarlamak için bir yöntem vardır. JMS\_IBM\_MQMD\_\* özelliklerini kullanır. You must set the destination property, WMQ\_MQMD\_WRITE\_ENABLED to set the JMS\_IBM\_MQMD\_\* properties:

```
((MQDestination)destination).setMQMDWriteEnabled(true);
```

Özellikleri okumak için hedef özelliğini ( WMQ\_MQMD\_READ\_ENABLED) ayarlamanız gerekir.

JMS\_IBM\_MQMD\_\* ' ı yalnızca tüm ileti bilgi yükü üzerinde tam denetim ele aldıysanız kullanın. JMS\_IBM\_\* özelliklerinden farklı olarak, JMS\_IBM\_MQMD\_\* özellikleri IBM MQ classes for JMS ' in bir JMS iletisinin nasıl oluşturulacağını denetmez. JMS iletisinin özellikleriyle çakışan ileti tanımlayıcı özellikleri yaratmak olanaklıdır.

e. Yöntemi, sınıftaki öznelikleri, iletteki alanlar olarak serileştiren kod satırlarını içerir.

Dizgi öznelikleri boşluklarla dolduruldu. Dizgiler, kayıt için tanımlanan karakter takımı kullanılarak bayt 'a dönüştürülür ve ileti alanlarının uzunluğuna kısaltılır.

5. İçer aktarma deyimlerini ekleyerek sınıfı tamamlayın.

```
package com.ibm.mq.id;  
import java.io.IOException;  
import java.io.Serializable;  
import java.io.UnsupportedEncodingException;  
import javax.jms.BytesMessage;  
import javax.jms.JMSException;  
import com.ibm.mq.constants.MQConstants;  
import com.ibm.mq.headers.MQDataException;  
import com.ibm.msg.client.wmq.WMQConstants;
```

6. Ek alanları dahil etmek için RECORD sınıfını 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 = "ABCDEFGHJKLMNOPQRSTUVWXYZ012345";  
  
    public MyRecord() {  
        super();  
        super.setStructID(STRUCT_ID);  
        super.setHeaderFormat(FORMAT);  
        super.setStructLength(super.getStructLength() + MQLONG_LENGTH  
            + DATA_LENGTH);  
    }  
}
```

**Not:**

a. RECORD alt sınıfı, MyRecord, gözün yakalayıcıya, biçimini ve uzunluğunun özelleştirilmesini sağlar.

7. Alıcıları ve ayarlayıcıları yaratın ya da oluşturun.

a) Alıcıları yarat:

```
public int getFlags() { return flags; }
public String getRecordData() { return recordData; } .
```

b) Ayarlayıcıları yaratın:

```
public void setFlags(int flags) {
    this.flags = flags; }
public void setRecordData(String recordData) {
    this.recordData = recordData; }
}
```

8. Create a constructor to create a MyRecord instance from a JMSBytesMessage.

```
public MyRecord(BytesMessage message) throws JMSEException, 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, RECORD sınıfı tarafından önce okunur.
  - b. The recordData text is converted to String using the character set property of the message.
9. Bir tüketiciden ileti almak ve yeni bir MyRecord yönetim ortamı yaratmak için durağan bir yöntem yaratın.

```
public static MyRecord get(MyConsumer myConsumer) throws JMSEException,
    MQDataException, IOException {
    BytesMessage message = (BytesMessage) myConsumer.receive();
    return new MyRecord(message);
}
```

**Not:**

- a. In the example, for brevity, the MyRecord(BytesMessage) constructor is called from the static get method. Genellikle, yeni bir MyRecord yönetim ortamı yaratmaktan ileti almayı birbirinden ayırabilirsiniz.
10. Müşteri alanlarını, ileti üstbilgisi içeren bir JMSBytesMessage 'a eklemek için bir put yöntemi oluşturun.

```
public BytesMessage put(MyProducer myProducer) throws JMSEException,
    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. Koddaki yöntem çağrıları, MyRecord sınıfındaki öznelikleri, iletteki alanlar olarak diziselleştirir.

- 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 javax.jms.BytesMessage;
import javax.jms.JMSEException;
import com.ibm.mq.headers.MQDataException;
```

## Sonuçlar

Sonuçlar:

- TryMyRecord sınıfını çalıştırdıktan sonra elde edilen sonuçlar:

- İletinin kodlanmış karakter takımı 37 'de gönderilmesi ve kuyruk yöneticisi dönüştürme çıkışısının kullanılması:

```
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, kuyruk yöneticisi dönüştürme çıkışı kullanılarak değil, 37 kodlu karakter kümesinde 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 546 CCSID IBM037
```

- The results from modifying the TryMyRecord class not to receive the message, and instead receiving it using the modified amqsget0.c sample. The modified sample accepts a formatted record; see [Şekil 38 sayfa 172](#) in “Exchanging a formatted record with a non-JMS application” [sayfa 169](#).

- İletinin kodlanmış karakter takımı 37 'de gönderilmesi ve kuyruk yöneticisi dönüştürme çıkışısının kullanılması:

```
Sample AMQSGET0 start
ccsid <850>, flags <1>, message <ABCDEFGHIJKLMNOPQRSTUVWXYZ012345>
no more messages
Sample AMQSGET0 end
```

- İleti, kuyruk yöneticisi dönüştürme çıkışı kullanılarak değil, 37 kodlu karakter kümesinde gönderiliyor.

```
Sample AMQSGET0 start
MQGET ended with reason code 2110
ccsid <37>, flags <1>, message <---+--ãÃ++ÐÊËËiÐÎÐ+ÔÔööµpÙ-±=¼¶§>
no more messages
Sample AMQSGET0 end
```

Örnek ve deneyi farklı kod sayfaları ve bir veri dönüştürme çıkışı ile denemek için. Java sınıflarını oluşturun, IBM MQ' u yapılandırın ve ana programı çalıştırın; TryMyRecord ; bkz. [Şekil 39 sayfa 180](#).

1. Örneği çalıştırmak için IBM MQ ve JMS ' yi yapılandırın. The instructions are for running the example on Windows.

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 Administration programı buradan çalıştırılmalıdır. Yol, *MQ\_INSTALLATION\_PATH*\java\bin.

e. Aşağıdaki JMS tanımlarını JMSQM1Q1.txt adlı bir dosyada 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 Gezginini 'ni kullanarak yarattığınız tanımlara göz atabilir, bunları değiştirebilir ve bunlara göz atabilirsiniz.

3. TryMyRecord komutunu çalıştırın.

### Örneği çalıştırmak için kullanılan sınıflar

The classes listed in figures [Şekil 39 sayfa 180](#) to [Şekil 44 sayfa 184](#) are also available in a compressed file; download [jm25529\\_.zip](#) or [jm25529\\_.tar.gz](#).

```
package com.ibm.mq.id;  
public class TryMyRecord {  
    public static void main(String[] args) throws Exception {  
        MyProducer producer = new MyProducer();  
        MyRecord outrec = new MyRecord();  
        System.out.println("Out flags " + outrec.getFlags() + " text "  
            + outrec.getRecordData() + " Encoding "  
            + producer.getEncoding() + " CCSID " + producer.getCCSID()  
            + " MQ " + producer.getMQDest());  
        outrec.put(producer);  
        System.out.println("Out flags " + outrec.getFlags() + " text "  
            + outrec.getRecordData() + " Encoding "  
            + producer.getEncoding() + " CCSID " + producer.getCCSID()  
            + " MQ " + producer.getMQDest());  
        MyRecord inrec = MyRecord.get(new MyConsumer());  
        System.out.println("In flags " + inrec.getFlags() + " text "  
            + inrec.getRecordData() + " Encoding "  
            + inrec.getMessageEncoding() + " CCSID "  
            + inrec.getMessageCharset());  
    }  
}
```

Şekil 39. TryMyRecord

```

package com.ibm.mq.id;
import java.io.IOException;
import java.io.Serializable;
import java.io.UnsupportedEncodingException;
import javax.jms.BytesMessage;
import javax.jms.JMSEException;
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 JMSEException, 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,
        JMSEException, 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; }
}

```

Şekil 40. RECORD

```

package com.ibm.mq.id;
import java.io.IOException;
import javax.jms.BytesMessage;
import javax.jms.JMSEException;
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 JMSEException, 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 JMSEException,
        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 JMSEException,
        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; }
}

```

Şekil 41. MyRecord

```

package com.ibm.mq.id;
import java.io.UnsupportedEncodingException;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSEException;
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, JMSEException {
        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,
        JMSEException {
        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 JMSEException {
        return (((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_CCSSID)); }
    public String getCharset() throws UnsupportedEncodingException,
        JMSEException {
        return CCSID.getCodepage(getCCSID()); }
    public int getEncoding() throws JMSEException {
        return (((MQDestination) destination)
            .getIntProperty(WMQConstants.WMQ_ENCODING)); }
    public boolean getMQDest() throws JMSEException {
        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 JMSEException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_CCSSID,
            ccsid); }
    public void setEncoding(int encoding) throws JMSEException {
        ((MQDestination) destination).setIntProperty(WMQConstants.WMQ_ENCODING,
            encoding); }
    public void setMQBody() throws JMSEException {
        ((MQDestination) destination)
            .setMessageBodyStyle(WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED); }
    public void setMQBody(boolean mqbody) throws JMSEException {
        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 JMSEException {
        if (mqclient){
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
            if (!getMQDest()) setMQBody();
        }
        else
            ((MQDestination) destination).setTargetClient(WMQConstants.WMQ_TARGET_DEST_JMS); }
}

```

Şekil 42. EndPoint

```

package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageProducer;
import javax.naming.NamingException;
public class MyProducer extends Endpoint {
    public MessageProducer producer;
    public MyProducer() throws NamingException, JMSException {
        super();
        producer = session.createProducer(destination); }
    public MyProducer(String cFactory, String dest) throws NamingException,
        JMSException {
        super(cFactory, dest);
        producer = session.createProducer(destination); }
    public void send(Message message) throws JMSException {
        producer.send(message); }
}

```

Şekil 43. MyProducer

```

package com.ibm.mq.id;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageConsumer;
import javax.naming.NamingException;
public class MyConsumer extends Endpoint {
    public MessageConsumer consumer;
    public MyConsumer() throws NamingException, JMSException {
        super();
        consumer = session.createConsumer(destination);
        connection.start(); }
    public MyConsumer(String cFactory, String dest) throws NamingException,
        JMSException {
        super(cFactory, dest);
        consumer = session.createConsumer(destination);
        connection.start(); }
    public Message receive() throws JMSException {
        return consumer.receive(); }
}

```

Şekil 44. MyConsumer

## **IBM MQ classes for JMS uygulamasındaki bağlantı fabrikalarını ve hedeflerini oluşturma ve yapılandırma**

An IBM MQ classes for JMS application can create connection factories and destinations by retrieving them as administered objects from a Java Naming and Directory Interface (JNDI) namespace, by using the IBM JMS extensions, or by using the IBM MQ JMS extensions. Bir uygulama, bağlantı fabrikalarının ve hedeflerin özelliklerini ayarlamak için IBM JMS uzantılarını ya da IBM MQ JMS uzantılarını da kullanabilir.

Bağlantı fabrikaları ve hedefler, JMS uygulamasının mantık akışındaki noktaları başlıyabiliyor. Bir uygulama, ileti alışverişi sunucusuyla bağlantı yaratmak için ConnectionFactory nesnesini kullanır ve iletilerin gönderileceği hedef ya da ileti göndermek için hedef olarak bir Kuyruk ya da Konu nesnesi kullanır. Bu nedenle, uygulamanın en az bir bağlantı üreticisi ve bir ya da daha çok hedef oluşturması gerekir. Bir bağlantı üreticisi ya da hedefi yarattıktan sonra, uygulamanın bir ya da daha çok özelliğini ayarlayarak nesnenin konfigürasyonunu tanımlamanız gerekebilir.

Özet olarak, bir uygulama bağlantı fabrikalarını ve hedeflerini aşağıdaki şekillerde oluşturabilir ve yapılandırabilir:

### **Yönetilen nesnelere almak için JNDI kullanılıyor**

Bir denetimci IBM MQ JMS yönetim aracını kullanarak Configuring objects using the JMS administration tool içinde açıklandığı şekilde ya da JNDI ad alanında yönetilen nesnelere olarak bağlantı üreticileri ve hedefleri oluşturmak ve yapılandırmak için Configuring JMS objects using IBM MQ Explorer' de açıklandığı gibi IBM MQ Explorer yönetim aracını kullanabilir. Daha sonra, uygulama, yönetilen nesnelere JNDI ad alanından alabilir. Having retrieved an administered object, the application can, if required, set or change one or more of its properties by using either the IBM JMS extensions or the IBM MQ JMS extensions.



## IBM JMS uzantılarını kullanma

Bir uygulama, yürütme sırasında dinamik olarak bağlantı fabrikaları ve hedefler oluşturmak için IBM JMS uzantılarını kullanabilir. Uygulama önce bir `JmsFactoryFactory` nesnesi yaratır ve sonra bağlantı üreticileri ve hedefler yaratmak için bu nesnenin yöntemlerini kullanır. Bir bağlantı üreticisi ya da hedefi yarattıktan sonra, uygulama, özelliklerini ayarlamak için `JmsPropertyBağlam` arabiriminden edinilen yöntemleri kullanabilir. Diğer bir seçenek olarak, uygulama hedefi oluştururken bir hedef için bir ya da daha fazla özellik belirtmek için bir URL 'yi (uniform resource identifier; URI) kullanabilir.

## IBM MQ JMS uzantılarını kullanma

Uygulama, yürütme sırasında dinamik olarak bağlantı üreticileri ve hedefler yaratmak için IBM MQ JMS uzantılarını da kullanabilir. Uygulama, bağlantı fabrikaları ve hedefler oluşturmak için sağlanan oluşturucuları kullanır. Bir bağlantı üreticisi ya da hedefi yarattıktan sonra, uygulama, nesnenin özelliklerini ayarlamak için nesne yöntemlerini kullanabilir. Diğer bir seçenek olarak, uygulama hedefi yaratırken bir hedefe ilişkin bir ya da daha çok özellik belirtmek için bir URI kullanabilir.

## İlgili görevler

### JMS kaynaklarının yapılandırılması

#### *Bir JMS uygulamasındaki denetimli nesnelere almak için JNDI olanağını kullanma*

To retrieve administered objects from a Java Naming and Directory Interface (JNDI) namespace, a JMS application must create an initial context and then use the `lookup()` method to retrieve the objects.

Bir uygulamanın yönetilen nesnelere JNDI ad alanından alabilmesi için, denetimcinin önce yönetilen nesnelere yaratması gerekir. Yönetici, bir JNDI ad alanında yönetilen nesnelere yaratmak ve bu nesnelere korumak için IBM MQ JMS yönetim aracını ya da IBM MQ Explorer olanağını kullanabilir. Daha fazla bilgi için bkz. [JNDI ad alanında bağlantı fabrikalarını ve hedeflerini yapılandırma](#).

Bir uygulama sunucusu, tipik olarak, yönetilen nesnelere için kendi havuzunu ve nesnelere yaratmak ve bakımını yapmak için kendi araçlarını sağlar.

Bir JNDI ad alanından denetlenen nesnelere almak için, aşağıdaki örnekte gösterildiği gibi, bir uygulamanın öncelikle başlangıç bağlamı yaratması gerekir:

```
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, `url` ve `icf` dizgi değişkenleri aşağıdaki anlamlara sahiptir:

### **url**

Dizin hizmetinin URL (uniform resource locator; URL adresi). URL adresi aşağıdaki biçimlerden birine sahip olabilir:

- `ldap://hostname/contextName` , for a directory service based on an LDAP server
- `file://directoryPath` , yerel dosya sistemine dayalı bir dizin hizmeti için

### **icf**

İlk bağlam üreticisinin sınıf adı; aşağıdaki değerlerden biri olabilir:

- `com.sun.jndi.ldap.LdapCtxFactory`, for a directory service based on an LDAP server
- `com.sun.jndi.fscontext.ReffsContextFactory`, yerel dosya sistemine dayalı bir dizin hizmeti için

Bir JNDI paketi ve bir LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) hizmet sağlayıcısının bazı birleşimlerinin, LDAP hata 84 'ünün oluşmasına neden olduğunu unutmayın. To resolve this problem, insert the following line of code before the call to InitialDirContext():

```
environment.put(Context.REFERRAL, "throw");
```

İlk bağlam elde edildikten sonra, uygulama, aşağıdaki örnekte gösterildiği gibi, lookup () yöntemini kullanarak, JNDI ad alanından denetimli nesnelere ulaşabilir:

```
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 nesnelere ulaşır:

- myCFadına bağlanan bir ConnectionFactory nesnesi
- myQadına bağlı bir Kuyruk nesnesi
- myTadına bağlı bir Konu nesnesi

JNDI kullanımına ilişkin ek bilgi için, Oracle Corporation tarafından sağlanan JNDI belgelerine bakın.

### İlgili görevler

[Configuring JMS objects using IBM MQ Explorer](#)

[Yönetim aracını kullanarak JMS nesnelere yapılandırma](#)

[Configuring JMS resources in WebSphere Application Server](#)

### IBM JMS uzantılarını kullanma

IBM MQ classes for JMS contains a set of extensions to the JMS API called the IBM JMS extensions. Bir uygulama, yürütme sırasında dinamik olarak bağlantı fabrikaları ve hedefler yaratmak ve IBM MQ classes for JMS nesnelere özelliklerini ayarlamak için bu uzantıları kullanabilir. Uzantılar, herhangi bir ileti alışverişi sağlayıcısıyla birlikte kullanılabilir.

IBM JMS uzantıları, aşağıdaki paketlerdeki bir arabirim ve sınıflardan oluşan bir kümedir:

- com.ibm.msg.client.jms
- com.ibm.msg.client.services

Paketler, *MQ\_INSTALLATION\_PATH*/java/libiçinde yer alan com.ibm.mqjms.jar içinde bulunabilir.

Bu uzantılar aşağıdaki işlevi sağlar:

- Bir Java Adlandırma ve Dizin Arabirimi (JNDI) ad alanından denetlenen nesnelere ulaşmak yerine, yürütme sırasında dinamik olarak bağlantı üreticileri ve varış noktaları yaratmak için kullanılan bir fabrikada kullanılan bir mekanizma
- IBM MQ classes for JMS nesnelere özelliklerini ayarlamak için kullanılan yöntemler kümesi
- Bir sorunla ilgili ayrıntılı bilgi edinme yöntemleriyle ilgili kural dışı durum sınıfları kümesi
- İzlemeyi denetlemeye ilişkin yöntemler kümesi
- IBM MQ classes for JMSile ilgili sürüm bilgilerini edinmeye ilişkin bir yöntem kümesi

With regard to creating connection factories and destinations dynamically at run time, and setting and getting their properties, the IBM JMS extensions provide an alternative set of interfaces to the IBM MQ JMS extensions. Ancak, IBM MQ JMS uzantıları IBM MQ ileti alışverişi sağlayıcısına özgü, IBM JMS uzantıları IBM MQ 'a özgü değildir ve [IBM MQ class for JMS mimarisini](#) de açıklanan katmanlı mimari içinde herhangi bir ileti alışverişi sağlayıcısıyla birlikte kullanılabilir.

The interface `com.ibm.msg.client.wmq.WMQConstants` contains the definitions of constants, which an application can use when setting the properties of IBM MQ classes for JMS objects using the IBM JMS extensions. Arabirim, herhangi bir ileti alışverişi sağlayıcısından bağımsız IBM MQ ileti alışverişi sağlayıcısı ve JMS değişmezleri için sabit değerler içerir.

Aşağıdaki içe aktarma deyimlerinin çalıştırıldığı varsayılarak, aşağıdaki kod örnekleri verilmiştir:

```
import com.ibm.msg.client.jms.*;
import com.ibm.msg.client.services.*;
import com.ibm.msg.client.wmq.WMQConstants;
```

## Bağlantı üreticileri ve hedefleri oluşturma

Bir uygulamanın IBM JMS uzantılarını kullanarak bağlantı üreticileri ve hedefleri oluşturabilmesi için önce bir `JmsFactoryFactory` nesnesi yaratması gerekir. Bir `JmsFactoryFactory` nesnesi yaratmak için, uygulama `JmsFactoryFactory` sınıfının `getInstance()` yöntemini çağırır. Aşağıdaki örnekte gösterildiği gibi:

```
JmsFactoryFactory ff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
```

`getInstance()` çağırısındaki parametre, seçilen ileti alışverişi sağlayıcısı olarak IBM MQ ileti alışverişi sağlayıcısını tanımlayan bir sabittir. Daha sonra uygulama, bağlantı fabrikaları ve hedefler yaratmak için `JmsFactoryFactory` nesnesini kullanabilir.

Bir bağlantı üreticisi yaratmak için, uygulama `JmsFactoryFactory` nesnesinin `createConnectionFactory()` yöntemini çağırır. Aşağıdaki örnekte gösterildiği gibi:

```
JmsConnectionFactory factory = ff.createConnectionFactory();
```

Bu deyim, tüm özellikleri için varsayılan değerlere sahip bir `JmsConnectionFactory` nesnesi yaratır; bu, uygulamanın bağ tanımları kipindeki varsayılan kuyruk yöneticisine bağlandığı anlamına gelir. Bir uygulamanın istemci kipinde bağlanmasını istiyorsanız ya da varsayılan kuyruk yöneticisi dışında bir kuyruk yöneticisine bağlanmak istiyorsanız, bu bağlantıyı yaratmadan önce uygulamanın `JmsConnectionFactory` nesnesinin uygun özelliklerini ayarlaması gerekir. Bunun nasıl yapacağına ilişkin bilgi için bkz. [“IBM MQ classes for JMS nesnelerinin özelliklerinin ayarlanması” sayfa 188.](#)

`JmsFactoryFactory` sınıfı, aşağıdaki tiplere ilişkin bağlantı üreticileri yaratmak için de yöntemler içerir:

- `JmsQueueConnectionFactory`
- `JmsTopicConnectionFactory`
- `JmsXAConnectionFactory`
- `JmsXAQueueConnectionFactory`
- `JmsXATopicConnectionFactory`

Bir kuyruk nesnesi yaratmak için, uygulama `JmsFactoryFactory` nesnesinin `createQueue()` yöntemini çağırır. Aşağıdaki örnekte gösterildiği gibi:

```
JmsQueue q1 = ff.createQueue("Q1");
```

Bu deyim, tüm özellikleri için varsayılan değerleri içeren bir `JmsQueue` nesnesi yaratır. Nesne, yerel kuyruk yöneticisine ait olan Q1 adlı bir IBM MQ kuyruğunu temsil eder. Bu kuyruk, yerel bir kuyruk, bir diğer ad kuyruğu ya da uzak kuyruk tanımlaması olabilir.

`createQueue()` yöntemi, bir parametre olarak bir kuyruk tekstili kaynak tanıtıcısını (URI) da kabul edebilir. Kuyruk URI 'si, bir IBM MQ kuyruğunun adını ve isteğe bağlı olarak, kuyruğa sahip olan kuyruk yöneticisinin adını ve `JmsQueue` nesnesinin bir ya da daha fazla özelliğini belirten bir dizgidir. Aşağıdaki deyimde bir kuyruk URI örneği yer almaktadır:

```
JmsQueue q2 = ff.createQueue("queue://QM2/Q2?persistence=2&priority=5");
```

The JmsQueue object created by this statement represents an IBM MQ queue called Q2 that is owned by queue manager QM2, and all messages sent to this destination are persistent and have a priority of 5. Kuyruk URI 'leri hakkında daha fazla bilgi için bkz. “Birörnek kaynak tanıtıcıları (URI 'ler)” sayfa 199. Bir JmsQueue nesnesinin özelliklerini ayara alternatif bir yol için bkz. “IBM MQ classes for JMS nesnelere özelliklerinin ayarlanması” sayfa 188.

Bir konu nesnesi oluşturmak 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 değerleri içeren bir JmsTopic nesnesi yaratır. Nesne, Sport/Football/Results adlı bir konuyu gösterir.

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 dizidir. Aşağıdaki deyimler bir konu URI örneğini içerir:

```
String s1 = "topic://Sport/Tennis/Results?persistence=1&priority=0";  
JmsTopic t2 = ff.createTopic(s1);
```

Bu deyimler tarafından yaratılan JmsTopic nesnesi, Sport/Tennis/Results adlı bir konuyu gösterir 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 199. Bir JmsTopic nesnesinin özelliklerini ayara alternatif bir yöntem için bkz. “IBM MQ classes for JMS nesnelere özelliklerinin ayarlanması” sayfa 188.

Bir uygulama, bir bağlantı üreticisi ya da hedefi yarattıktan sonra, bu nesne yalnızca seçilen ileti sistemi sağlayıcısıyla kullanılabilir.

## IBM MQ classes for JMS nesnelere özelliklerinin ayarlanması

To set the properties of IBM MQ classes for JMS objects using the IBM JMS extensions, an application uses the methods of the com.ibm.msg.client.JmsPropertyContext interface.

Her Java veri tipi için, JmsPropertyBağlam arabirimi, o veri tipine sahip bir özelliğin değerini ayarlamak için bir yöntem ve bu veri tipine sahip bir özelliğin değerini almak için bir yöntem içerir. Örneğin, bir uygulama setIntProperty () yöntemini, bir tamsayı değeri olan bir özelliği ayarlamayı çağırır ve getIntProperty () yöntemini, bir tamsayı değeri olan bir özelliği almak için çağırır.

com.ibm.mq.jms paketindeki sınıfların eşgörünümleri de JmsPropertyBağlam arabiriminin yöntemlerini devralır. Bu nedenle, bir uygulama bu yöntemleri kullanarak MQConnectionFactory, MQQueue ve MQTopic nesnelere ilişkin özellikleri ayarlar.

Bir uygulama bir IBM MQ classes for JMS nesnesi yarattığında, varsayılan değerleri olan özellikler otomatik olarak ayarlanır. Bir uygulama bir özelliği ayarladığında, yeni değer özelliğin sahip olduğu önceki değer yerini alır. Bir özellik ayarlandıktan sonra silinemez, ancak değeri değiştirilebilir.

Bir uygulama, özellik için geçerli olmayan bir değere özellik ayarlamaya çalışırsa, IBM MQ classes for JMS bir JMSEnvironment kural dışı durumu yayınlar. Bir uygulama, ayarlanmamış bir özelliği alma girişiminde bulunursa, davranış JMS belirtiminde anlatıldığı gibidir. IBM MQ classes for JMS , temel veri tipleri için bir NumberFormatkural dışı durumu kural dışı durumu yayınladı ve başvuru veri tipleri için boş değer döndürür.

Bir uygulama, bir IBM MQ classes for JMS nesnesinin önceden tanımlanmış özelliklerine ek olarak, kendi özelliklerini de ayarlayabilir. Bu uygulama tanımlı özellikler IBM MQ classes for JMS tarafından yoksa idi.

IBM MQ classes for JMS nesnelere özellikleri hakkında daha fazla bilgi için bakınız: [Properties of IBM MQ classes for JMS objects](#).

Aşağıdaki kod, özellikleri IBM JMS uzantılarını kullanarak nasıl ayarlayabilmeye 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");
```

The effect of setting these properties is that the application connects to queue manager QM1 in client mode, using an MQI channel called QM1.SVR. Kuyruk yöneticisi, anasistem adı HOST1olan bir sistemde çalışıyor ve kuyruk yöneticisinin dinleyicisi 1415 numaralı kapıda dinlemede. Bu bağlantı ve bunun altındaki oturumlarla ilişkili diğer kuyruk yöneticisi bağlantıları, onlarla ilişkili uygulama adına "Uygulamam" adını verdi.

**Not:** z/OS altyapılarında çalışan kuyruk yöneticileri uygulama adlarını ayarlamamaktadır ve bu nedenle bu ayar yoksayılr.

The JmsPropertyContext interface also contains the setObjectProperty() method, which an application can use to set properties. Yöntemin ikinci parametresi, özelliğin değerini sarmalayan bir nesnedir. Örneğin, aşağıdaki kod, 1415 değerini soyan bir Tamsayı nesnesi yaratır ve daha sonra, bağlantı üreticisinin PORT özelliğini 1415 değerine ayarlamak için setObjectProperty () çağrılarını çağırır:

```
Integer port = new Integer(1415);
factory.setObjectProperty(WMQConstants.WMQ_PORT, port);
```

Bu nedenle, bu kod aşağıdaki deyimle eşdeğerdir:

```
factory.setIntProperty(WMQConstants.WMQ_PORT, 1415);
```

Tersi durumda, getObjectProperty () yöntemi, bir özelliğin değerini sarmalayan bir nesneyi döndürür.

## Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi

Bir uygulama bir IBM MQ classes for JMS nesnesinin özelliğini ayarlamak ya da almak için JmsPropertybağlam arabirimi yöntemini kullandığında, özelliğin değeri bir veri tipinden başka bir veri tipinden örtük olarak dönüştürülebilir.

Örneğin, aşağıdaki deyim JmsQueue nesnesinin PRIORITY özelliğini ayarlar. q1:

```
q1.setStringProperty(WMQConstants.WMQ_PRIORITY, "5");
```

PRIORITY özelliğinin bir tamsayı değeri vardır; dolayısıyla, setStringProperty () çağrısı örtük olarak "5" (kaynak değer) dizgisini (kaynak değer) tamsayı 5 'e (hedef değer) dönüştürür ve daha sonra, PRIORITY özelliğinin değeri olur.

Bunun tersine, aşağıdaki deyim JmsQueue nesnesinin PRIORITY özelliğini alır. 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) dönüştürüldü.

IBM MQ classes for JMS tarafından desteklenen dönüştürmeler [Çizelge 33 sayfa 189](#) içinde gösterilir.

<i>Çizelge 33. Bir veri tipinden diğerine dönüştürme desteklenir</i>	
<b>Kaynak veri tipi</b>	<b>Desteklenen hedef veri tipleri</b>
boolean	Dizgi

Çizelge 33. Bir veri tipinden diğerine dönüştürme desteklenir (devamı var)

Kaynak veri tipi	Desteklenen hedef veri tipleri
Byte	int, long, short, String
DAMGA	Dizgi
double	Dizgi
kayan nokta	çift, Dizgi
int	uzun, Dizgi
long	Dizgi
short	int, long, String
Dizgi	boolean, byte, double, float, int, long, short

Desteklenen dönüştürmeleri yöneten genel kurallar aşağıdaki gibidir:

- Sayısal değerler, dönüştürme sırasında bir veri tipinden diğerine dönüştürülebilecek şekilde dönüştürülebilirler. For example, a value with data type `int` can be converted into a value with data type `long`, but cannot be converted into a value with data type `short`.
- Herhangi bir veri tipindeki bir değer bir dizgiye dönüştürülebilir.
- Bir dize, herhangi bir diğer veri türüne ( `char` dışında) dönüştürülebilmektedir. sağlanan dizgi, dönüştürme için doğru biçimde olmalıdır. Bir uygulama, doğru biçimde olmayan bir dizgiyi dönüştürmeye çalışırsa, IBM MQ classes for JMS bir `NumberFormatException` dışı durumu kural dışı durumu atar.
- Bir uygulama, desteklenmeyen bir dönüştürme girişiminde bulunursa, IBM MQ classes for JMS bir `MessageFormatException` dışı durumu kural dışı durumu yayınlar.

Bir veri tipinden başka bir veri tipine dönüştürme için belirli kurallar şunlardır:

- Bir boole değerini dizgiye dönüştürürken, `true` değeri "true"dizgisine dönüştürülür ve `false` değeri "false"dizgisine dönüştürülür.
- Bir dizgiyi bir boole değerine dönüştürürken, "doğru" dizgisi (büyük-küçük harfe duyarlı değil) `true` değerine dönüştürülür ve "yanlış" dizgisi (büyük/küçük harfe duyarlı değildir) `false` değerine dönüştürülür. Diğer herhangi bir dizgi `false`'a dönüştürülür.
- Bir dizgiyi `byte`, `int`, `long`ya da `short` veri tipli bir değere dönüştürürken, dizginin şu biçimde olması gerekir:

[ boşluk ] [ *sign* ] *haneler*

Dizginin bileşenlerinin anlamları aşağıdaki gibidir:

#### **boşluklar**

İsteğe bağlı olarak baştaki boş karakterler.

#### **İşaret**

İsteğe bağlı artı işareti (+) ya da eksi işareti (-).

#### **Rakamlar**

Bitişik basamaklar dizisi (0-9). En az bir basamak var olmalıdır.

Basamaklar sırasından sonra, dizgi sayı olmayan diğer karakterleri içerebilir, ancak bu karakterlere ilk girildiği anda dönüştürme durakları durur. Dizilimin ondalık bir tamsayıyı temsil ettiği varsayılır.

Dizgi doğru biçimde değilse, IBM MQ classes for JMS bir `NumberFormatException` dışı durumu kural dışı durumu yayınlıyor.

- Bir dizgiyi `double` ya da `float` veri tipine sahip bir değere dönüştürürken, dizginin şu biçimde olması gerekir:

[ boşluk ] [ *işaret* ] *basamaklar* [ *e\_char* [ *e\_işareti* ] *e\_basamakları* ]

Dizginin bileşenlerinin anlamları aşağıdaki gibidir:

**boşluklar**

İsteğe bağlı olarak baştaki boş karakterler.

**İşaret**

İsteğe bağlı artı işareti (+) ya da eksi işareti (-).

**Rakamlar**

Bitişik basamaklar dizisi (0-9). En az bir basamak var olmalıdır.

**e\_char**

E ya da Eolan bir üstel karakter.

**e\_sign**

Üstel için isteğe bağlı artı işareti (+) ya da eksi işareti (-).

**e\_basamakları**

Üs için bitişik basamaklar dizisi (0-9). Dizgi bir üstel karakter içeriyorsa en az bir basamak var olmalıdır.

Basamaklar sırasından sonra ya da bir üssü temsil eden isteğe bağlı karakterler, dizgi olmayan diğer karakterleri içerebilir, ancak bu karakterlere ilk girişe ulaşıldığında dönüştürme durakları durur. Dizginin, 10 'un gücü olan bir üstel bir ondalık kayan noktalı sayıyı temsil ettiği varsayılır.

Dizgi doğru biçimde değilse, IBM MQ classes for JMS bir NumberFormatkural dışı durumu kural dışı durumu yayınlıyor.

- When converting a numeric value (including a value with data type byte ) to a string, the value is converted to the string representation of the value as a decimal number, not the string containing the ASCII character for that value. For example, the integer 65 is converted to the string "65", not the string "A".

## Tek bir çağrıda birden çok özellik ayarlama

The JmsPropertyContext interface also contains the setBatchProperties() method, which an application can use to set more than one property in a single call. Yöntemin değiştirgesi, özellik ad-değer çiftleri kümesini sarmalayan bir Eşlem nesnesidir.

Örneğin, aşağıdaki kod, bir bağlantı üreticisinin beş özelliğini [“IBM MQ classes for JMS nesnelere ilişkin özelliklerinin ayarlanması” sayfa 188](#) içinde gösterildiği gibi ayarlamaya ilişkin setBatchProperties () yöntemini kullanır. Bu kod, Harita arabirimini gerçekleştiren HashMap sınıfının bir eşgörünümünü oluşturur.

```
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 bir veri tipine sahip bir özellik değeri, bir nesne içinde ya da örnekte gösterildiği gibi bir dizgi tarafından gösterilmelidir.

setBatchProperties () yöntemi her özelliği doğrular. setBatchProperties () yöntemi bir özelliği ayarlayamıyorsa, örneğin değeri geçerli değil, belirtilen özelliklerin hiçbiri ayarlanmamış.

## Özellik adları ve değerleri

Bir uygulama, IBM MQ classes for JMS nesnelere ilişkin özelliklerini ayarlamak ve almak için JmsPropertyContext arabiriminin yöntemlerini kullanıyorsa, uygulama aşağıdaki yöntemlerden herhangi birinde özelliklerin adlarını ve değerlerini belirtebilir. Each of the accompanying examples shows how to

set the PRIORITY property of the JmsQueue object q1 so that a message sent to the queue has the priority specified on the send() call.

### **com.ibm.msg.client.wmq.WMQConstants arabiriminde sabit deęerler olarak tanımlanan özellik adlarını ve deęerlerini kullanma**

Aşağıdaki deyim, bu şekilde özelliklerin adlarını ve deęerlerini nasıl belirleyeceğini gösteren bir örnektir:

```
q1.setIntProperty(WMQConstants.WMQ_PRIORITY, WMQConstants.WMQ_PRI_APP);
```

### **Kuyruk ve konu tekstili kaynak tanıtıcıları (URI ' ler) içinde kullanılacak özellik adlarını ve deęerlerini kullanma**

Aşağıdaki deyim, bu şekilde özelliklerin adlarını ve deęerlerini nasıl belirleyeceğini gösteren bir örnektir:

```
q1.setIntProperty("priority", -2);
```

Bu şekilde yalnızca hedeflerin özelliklerinin adları ve deęerleri belirtilebilir.

### **IBM MQ JMS yönetim aracı tarafından tanınan özellik adlarını ve deęerlerini kullanma**

Aşağıdaki deyim, bu şekilde özelliklerin adlarını ve deęerlerini nasıl belirleyeceğini gösteren bir örnektir:

```
q1.setStringProperty("PRIORITY", "APP");
```

Özellik adının kısa biçimi de, aşağıdaki deyimde gösterildiği gibi kabul edilebilir:

```
q1.setStringProperty("PRI", "APP");
```

Bir uygulama bir özellik aldığımda, döndürülen deęer, uygulamanın özelliğın adını belirtme yöntemine baęlıdır. For example, if an application specifies the constant WMQConstants.WMQ\_PRIORITY as the property name, the value returned is the integer -2:

```
int n1 = getIntProperty(WMQConstants.WMQ_PRIORITY);
```

Uygulama, özellik adı olarak "priority" dizesini belirtiyorsa, aynı deęer döndürülür:

```
int n2 = getIntProperty("priority");
```

Ancak, uygulama özellik adı olarak "PRIMARY" dizgisini ya da "PRI" dizgisini belirtiyorsa, döndürülen deęer "APP" dizgisidir:

```
String s1 = getStringProperty("PRI");
```

Dahili olarak IBM MQ classes for JMS , özellik adlarını ve deęerlerini com.ibm.msg.client.wmq.WMQConstants arabiriminde tanımlı hazır bilgi deęerleri olarak saklar. Bu, özellik adları ve deęerleri için tanımlanan kurallı biçimdir. Genel bir kural olarak, bir uygulama, özellik adlarını ve deęerlerini belirtmenin dięer iki yönteminden birini kullanarak özellikleri ayarlarsa, IBM MQ classes for JMS belirtilen giriş biçiminden adları ve deęerleri kurallı biçime dönüştürmesi gerekir. Benzer şekilde, bir uygulama özellik adlarını ve deęerlerini belirtmenin dięer iki yolundan birini kullanarak özellikler alıyorsa, IBM MQ classes for JMS belirtilen giriş biçiminden adları kurallı biçime dönüştürmeli ve deęerleri kurallı biçimden gerekli çıkış biçimine dönüştürmelidir. Bu dönüştürmeleri gerçekleştirmek için, bu dönüştürmelerin başarımı olumsuz sonuçlar doğurabilir.

Özellik adları ve istisnalar tarafından döndürülen deęerler, izleme dosyalarında ya da IBM MQ classes for JMS günlüğünde her zaman kurallı biçimde olur.



## Harita arabirimini kullanma

JmsPropertyBağlam arabirimi, java.util.Map arabirimini genişletir. Bu nedenle, bir uygulama, bir IBM MQ classes for JMS nesnesinin özelliklerine erişmek için Harita arabiriminin yöntemlerini kullanabilir.

Örneğin, aşağıdaki kodda bir bağlantı üreticisinin tüm özelliklerinin adları ve değerleri yazdırılıyor. Kod, özelliklerin adlarını ve değerlerini almak için yalnızca Harita arabirimi yöntemlerini kullanır.

```
// Get the names of all the properties
Set propNameNames = factory.keySet();

// Loop round all the property names and get the property values
Iterator iterator = propNameNames.iterator();
while (iterator.hasNext()) {
    String propName = (String)iterator.next();
    System.out.println(propName+"="+factory.get(propName));
}
```

Harita arabirimi yöntemlerinin kullanılması, özellik doğrulamalarını ya da dönüştürmeleri 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, çalışma 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 nesnelere erişmek yerine, çalışma zamanında bağlantı üreticilerini ve hedefleri dinamik olarak oluşturur
- 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 194.](#)

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.SVR adlı 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 biçimini kullanabilir. İleti sisteminin noktadan noktaya iletişim biçimini 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 özellikleri](#) başlıklı 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 değişikliği 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 tanımlanan 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 bir örnek 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. "Bir örnek kaynak tanıtıcıları (URI 'ler)" sayfa 199. Bir MQQueue nesnesinin özelliklerini ayarlamanın diğer bir yolu için bkz. "Hedeflerin özelliklerini ayarlama" sayfa 195.

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. [“Bir örnek kaynak tanıtıcıları \(URI 'ler\)” sayfa 199](#). Bir MQTopic nesnesinin özelliklerini ayarlamanın diğer bir yolu için bkz. [“Hedeflerin özelliklerini ayarlama” sayfa 195](#).

## 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 özellikleri](#) başlıklı konuya bakın.

## JMS uygulamasında bağlantı oluşturulması

Bir bağlantı oluşturmak için bir JMS uygulaması, Connection nesnesi yaratmak için ConnectionFactory nesnesini kullanır ve sonra bağlantıyı başlatır.

Bir uygulama, Connection nesnesi yaratmak için aşağıdaki örnekte gösterildiği gibi ConnectionFactory nesnesinin createConnection() yöntemini kullanır:

```
ConnectionFactory factory;
Connection connection;
.
.
.
connection = factory.createConnection();
```

Bir JMS bağlantısı oluşturulduğunda, IBM MQ classes for JMS bir bağlantı tanıtıcısı (Hconn) yaratır ve kuyruk yöneticisiyle bir etkileşim başlatır.

QueueConnectionFactory arabirimi ve TopicConnectionFactory arabirimi her biri 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 bir etki alanı bağımsız işlemi gerçekleştirebilir ya da yalnızca noktadan noktaya etki alanı için geçerli olan bir işlemdir. Ancak, uygulama yalnızca yayınlama/abone olma etki alanı için geçerli olan bir işlemi gerçekleştirmeye çalışırsa, şu iletiyle bir IllegalStateKural Dışı Durumu kural dışı durumu yayınladı:

```
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 bir etki alanından belirli bir bağlantı üreticisinden oluşturulduğundan kaynaklanır.

**Not:** Uygulama işlemi tanıtıcısının kuyruk yöneticisine geçirilecek varsayılan kullanıcı kimliği olarak kullanıldığını unutmayın. Uygulama istemci taşıma kipinde çalışıyorsa, bu işlem tanıtıcısı, ilgili yetkilerle sunucuda var olmalıdır. Farklı bir kimliğin kullanılmasını istiyorsanız, createConnection(kullanıcı adı, parola) yöntemini kullanın.

JMS belirtimi, stopped durumunda bir bağlantının yaratıldığını belirtir. Bağlantı başlatılıncaya kadar, bağlantıyla ilişkili bir ileti tüketicisi hiçbir ileti alamaz. Bir uygulama, bağlantı başlatmak için, aşağıdaki örnekte gösterildiği gibi, Connection nesnesinin start () yöntemini kullanır:

```
connection.start();
```

### **JMS uygulamasında oturum oluşturma**

Bir oturum oluşturmak için, bir JMS uygulaması Connection nesnesinin createSession() yöntemini kullanır.

createSession() yönteminde iki değiştirge vardır:

1. Oturumun hareket edip edilmediğini belirten bir değiştirge
2. Oturuma ilişkin alındı bildirim modunu belirten bir parametre

Örneğin, aşağıdaki kod, hareket eden bir oturum yaratır ve AUTO\_RELSE ' in bir alındı bildirim kipine sahiptir:

```
Session session;  
.  
boolean transacted = false;  
session = connection.createSession(transacted, Session.AUTO_ACKNOWLEDGE);
```

Bir JMS oturumu oluşturulduğunda, IBM MQ classes for JMS bir bağlantı tanıtıcısı (Hconn) yaratır ve kuyruk yöneticisiyle bir etkileşim başlatır.

Bir Oturum nesnesi ve ondan yaratılan herhangi bir MessageProducer ya da MessageConsumer nesnesi, çok iş parçacıklı bir uygulamanın farklı iş parçacıkları tarafından koşut zamanlı olarak kullanılamaz. Bu nesnelerin eş zamanlı olarak kullanılmamasını sağlamanın en basit yolu, her iş parçacığı için ayrı bir Oturum nesnesi yaratmamaktır.

### *JMS uygulamalarındaki hareket edilen oturumlar*

JMS uygulamaları, önce hareket edilen bir oturum oluşturarak yerel işlemleri çalıştırabilir. Bir uygulama bir işlemi kesinleştirebilir ya da geri alabilir.

JMS uygulamaları yerel işlemleri çalıştırabilir. Yerel hareket, değişikliklerin yalnızca, uygulamanın bağlı olduğu kuyruk yöneticisinin kaynaklarında yapılan değişiklikleri içeren bir işlemdir. To run local transactions, an application must first create a transacted session by calling the createSession() method of a Connection object, specifying as a parameter that the session is transacted. Daha sonra, oturum içinde gönderilen ve alınan tüm iletiler bir işlem sırasına göre gruplandırılır. Bir işlem, uygulama başlattığında ya da gönderdiği iletileri geri alırken, işlem başlatıldığından bu yana aldığı iletiler sona erer.

Bir işlem kesinleştirilmek için, bir uygulama Oturum nesnesinin commit () yöntemini çağırır. Bir işlem kesinleştirildiğinde, işlem içinde gönderilen tüm iletiler diğer uygulamalara teslim edilebilmesi için kullanılabilir duruma gelir ve işlem içinde alınan tüm iletiler, ileti alışverişi sunucusunun bunları yeniden uygulamaya teslim etme girişiminde bulunmayabilmesi için kabul edilir. Noktadan noktaya iletişim alanında, ileti alışverişi sunucusu alınan iletileri kuyruklarından da kaldırır.

Bir işlemi geri yüklemek için, bir uygulama Oturum nesnesinin rollback () yöntemini çağırır. Bir hareket geriye işlendiğinde, işlem içinde gönderilen tüm iletiler ileti alışverişi sunucusu tarafından atılır ve işlem içinde alınan tüm iletiler, teslim için yeniden kullanılabilir duruma gelir. Noktadan noktaya iletişim alanında, alınan iletiler kuyruklarına geri alınır ve diğer uygulamalar tarafından görülebilir duruma gelir.

Uygulama, hareket eden bir oturum yarattığında ya da commit () ya da rollback () yöntemini çağırıldığında otomatik olarak yeni bir işlem başlar. Bu nedenle, hareket eden bir oturumun her zaman etkin bir hareketi vardır.

Bir uygulama hareket edilen bir oturumu kapattığında, örtük bir geri alma gerçekleşir. Bir uygulama bağlantıyı kapattığında, tüm bağlantının hareket ettiği oturumlar için örtük bir geri alma gerçekleşir.

Bir uygulama bağlantıyı kapatmadan sona ererse, tüm bağlantının hareket ettiği oturumlar için örtük bir geri alma da gerçekleşir.

Hareket, hareket eden bir oturumda tamamen içerilidir. Bir hareket oturumlara yayılamaz. Diğer bir deyişle, bir uygulamanın iki ya da daha fazla sayıda dönüştürücü oturumuna ileti gönderip alması ve sonra tüm bu işlemleri tek bir işlem olarak kesinleştirmesi ya da geriye işlemleri mümkün değildir.

### *JMS oturumlarının alındı bildirimi kipleri*

İletilmeyen her oturumda, uygulamanın aldığı iletilerin nasıl kabul edildiğine karar veren bir alındı bildirimi kipi vardır. Üç alındı bildirimi kipi vardır ve alındı bildirimi kipinin seçimi, uygulamanın tasarımını etkiler.

Bir oturum hareket edilmezse, uygulamanın aldığı iletilerin kabul edilme şekli, oturumun alındı bildirimi kipiyle belirlenir. Üç alındı bildirimi kipleri aşağıdaki paragraflarda açıklanmaktadır:

#### **OTO\_BILDIR**

Oturum, uygulama tarafından alınan her iletiyi otomatik olarak kabul eder.

İletiler uygulamaya zamanuyumlu olarak teslim ediliyorsa, bir Receive çağrısı her başarıyla tamamlandığında, oturum bir iletinin alındığını kabul eder. İletilerin zamanuyumsuz olarak teslim edilmesi durumunda, bir ileti dinleyicisinin onMessage() yöntemine her çağrı başarıyla tamamlanırsa, oturum bir iletinin alındığını kabul eder.

Uygulama başarıyla bir ileti alırsa, ancak hata onayının oluşmasını önlüyorsa, ileti teslim için kullanılabilir duruma gelir. Bu nedenle, uygulamanın yeniden teslim edilen bir iletiyi işleyebilmesi gerekir.

#### **DUPS\_OK\_BILDIR**

Bu oturum, uygulama tarafından alınan iletilerin seçim süresinde kabul ettiği kabul eder.

Bu alındı bildirimi kipinin kullanılması, oturumun yapması gereken iş miktarını azaltır, ancak ileti onayını önleyen bir hata, birden çok iletinin teslim edilmesi için kullanılabilir duruma gelmesinden kaynaklanabilir. Bu nedenle, uygulamanın yeniden teslim edilen iletileri işleyebilmesi gerekir.

**Sınırlama:** AUTO\_TASK ve DUPS\_OK\_LASE kiplerinde, JMS bir ileti dinleyicisinde işlenemeyen bir kural dışı durum yayınlamak uygulamayı desteklemez. Bu, ileti dinleyici döndüğünde, başarıyla

işlenip işlenmediğine bakılmaksızın, ileti dinleyici döndüğünde iletilerin her zaman kabul ettiği anlamına gelir (hatalar onarılmaz değildir ve uygulamanın devam etmemesini engellemektedir). İleti onayının daha iyi bir şekilde denetlenmesini istiyorsanız, alındı bildirim işlevlerinin uygulamaya tam denetimini veren CLIENT\_RELIND ya da transated kiplerini kullanın.

## CLIENT\_ALIND

Uygulama, İleti sınıfının Acknowy yöntemini çağırarak aldığı iletileri kabul eder.

Uygulama, her iletinin alınmasını tek tek onaylayabilir ya da bir ileti kümesi alabilir ve Acknowy yöntemini yalnızca aldığı son ileti için çağırabilir. Yöntemin çağırıldığı son çağrıdan bu yana Acknowy yöntemi çağırıldığında alınan tüm iletiler onaylanır.

Bu onay kiplerinden herhangi biriyle birlikte, bir uygulama Oturum sınıfının Recon yöntemini çağırarak, bir oturumda iletilerin teslim edilmesini durdurabilir ve yeniden başlatabilir. Alınan ancak daha önce kabul edilmeyen 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 alanında, özgün iletilerin bazıları başka bir uygulama tarafından tüketilmiş olabilir.

Bir uygulama, iletinin JMSReverulmuş üstbilgi alanının içeriğini inceleyerek bir iletinin yeniden teslim edilip edilmediğini belirleyebilir. Uygulama bunu, Message sınıfının getJMSRedelivered() yöntemini çağırarak gerçekleştirir.

## JMS uygulamasında hedef oluşturma

Bir JMS uygulaması, Java Naming and Directory Interface (JNDI) ad alanından yönetilen nesnelere 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 adıyla 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, QM2 adlı 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 tanımlanan kuyruk yöneticisi yerel kuyruk yöneticisi ya da uzak kuyruk yöneticisi olabilir. Uzak kuyruk yöneticisiyse, IBM MQ yapılandırılmalıdır; böylece, uygulama bu hedefe bir ileti gönderdiğinde, WebSphere MQ iletiyi yerel kuyruk yöneticisinden QM2 kuyruk yöneticisine yönlendirebilir. URI ' ler hakkında daha fazla bilgi için bkz. [“Bir örnek kaynak tanıtıcıları \(URI ' ler\)” sayfa 199.](#)

createQueue() yöntemindeki parametrenin sağlayıcıya özgü bilgileri 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 199.](#)

`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 tanıtilan kuyruğun iyisi 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 nesnelere adlandırma kuralları](#) başlıklı konuya bakın.

#### **topicName**

Konunun adı.

Konu adları yaratmaya ilişkin kurallar için [IBM MQ nesnelere adlandırma kuralları](#) başlıklı konuya bakın. +, #, \* ve ? joker karakterlerini kullanmaktan kaçının Konu adlarında. Bu karakterleri içeren konu adları, bunlara abone olduğunuzda beklenmeyen sonuçlara neden olabilir. Bkz. [Konu dizgilerini kullanma](#).

#### **propertyName1, propertyName2, ...**

Uygulama tarafından yaratılan Kuyruk ya da Konu nesnesinin özelliklerinin adları. [Çizelge 34 sayfa 201](#) içinde, bir URI ' de kullanılacak 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 34 sayfa 201](#) içinde, bir URI ' de kullanılacak 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 34 sayfa 201](#) içinde, kuyrukta ve konu URI ' lerinde kullanılacak 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 34. 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	<ul style="list-style-type: none"><li>• IBM MQ tarafından desteklenen herhangi bir kodlanmış karakter takımı tanıtıcısı.</li></ul>
Kodlama	IBM MQ classes for JMS iletiyi hedefe ilettiğinde, iletinin gövdesindeki sayısal verilerin nasıl temsil edileceği	<ul style="list-style-type: none"><li>• IBM MQ ileti tanımlayıcısındaki <i>Kodlama</i> alanı için geçerli herhangi bir değer.</li></ul>
Son kullanma tarihi	Hedefe gönderilen iletiler için yaşam süresi	<ul style="list-style-type: none"><li>• -2-send () çağrısında belirtildiği gibi ya da send () çağrısında belirtilmezse, ileti üreticisinin varsayılan yaşam süresi.</li><li>• 0-Hedefe gönderilen bir iletinin süresi hiçbir zaman dolmaz.</li><li>• Milisaniye cinsinden yaşam süresini belirten pozitif bir tamsayı.</li></ul>
çoklu yayın	Bir aracıya gerçek zamanlı bağlantı kullanılırken bir konuya ilişkin çoklu yayın ayarı	<p>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. <a href="#">IBM MQ classes for JMS nesnelerinin özellikleri</a>.</p> <ul style="list-style-type: none"><li>• -1-ASCF</li><li>• 0-DEVRE Dışı</li><li>• 3-HAYIR</li><li>• 5-GÜVENİLİR</li><li>• 7-ETKİN</li></ul>
Kalıcılık	Hedefe gönderilen iletilerin kalıcılığı	<ul style="list-style-type: none"><li>• -2-send () çağrısında belirtildiği gibi ya da send () çağrısında belirtilmezse, ileti üreticisinin varsayılan kalıcılığını belirler.</li><li>• -1- IBM MQ kuyruğunun ya da konusunun <i>DefPersistence</i> (DefPersistence) özniteliği tarafından belirtildiği şekilde.</li><li>• 1-Kalıcı Değil.</li><li>• 2-Kalıcı.</li><li>• 3- IBM MQ JMS yönetim aracında kullanılan PERSISTENCE özelliği için HIGH değerinin eşdeğeri. Bu değer için açıklaması için bkz. "<a href="#">JMS kalıcı iletileri</a>" sayfa 224.</li></ul>

Çizelge 34. Kuyrukta ve konu URI ' lerinde kullanılmak üzere özellik adları ve geçerli değerler (devamı var)

Özellik adı	Açıklama	Geçerli değerler
öncelik	Hedefe gönderilen iletilerin önceliği	<ul style="list-style-type: none"><li>-2-send () çağrısında belirtildiği gibi ya da send () çağrısında belirtilmezse, ileti üreticisinin varsayılan önceliği.</li><li>-1- IBM MQ kuyruğunun ya da konusunun <i>DefPriority</i> özniteliği tarafından belirtildiği şekilde.</li><li>Hedefe gönderilen iletilerin önceliğini belirten 0-9 aralığında bir tamsayı.</li></ul>
targetClient	Hedefe gönderilen iletilerin MQRFH2 üstbilgisi içerip içermediğini belirler	<ul style="list-style-type: none"><li>0-İletiler bir MQRFH2 üstbilgisi içeriyor.</li><li>1-İletiler MQRFH2 üstbilgisi içermiyor.</li></ul>

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 uygulamasına ileti gönderme**

Bir JMS uygulaması bir hedefe ileti göndermeden önce, hedef için önce bir MessageProducer nesnesi yaratmalıdır. Hedefe bir ileti göndermek için, uygulama bir ileti nesnesi yaratır ve MessageProducer nesnesinin gönderme () yöntemini çağırır.

Bir uygulama, iletileri göndermek için bir MessageProducer nesnesini kullanır. Bir uygulama, genellikle belirli bir hedef için bir MessageProducer nesnesi yaratır; bu nesne, ileti üreticisi kullanılarak gönderilen tüm iletilerin aynı hedefe gönderileceği şekilde bir kuyruk ya da konu olabilir. Bu nedenle, bir uygulamanın bir MessageProducer nesnesi yaratabilmesi için önce bir Kuyruk ya da Konu nesnesi yaratması gerekir. Bir Kuyruk ya da Konu nesnesinin nasıl yaratılacağı hakkında bilgi için aşağıdaki konulara bakın:

- [“Bir JMS uygulamasındaki denetimli nesnelere almak için JNDI olanağını kullanma” sayfa 185](#)
- [“IBM JMS uzantılarını kullanma” sayfa 186](#)
- [“IBM MQ JMS uzantılarının kullanılması” sayfa 193](#)
- [“JMS uygulamasında hedef oluşturma” sayfa 198](#)

Bir uygulama, MessageProducer nesnesi yaratmak için, aşağıdaki örnekte gösterildiği gibi, bir Oturum nesnesinin createProducer() yöntemini kullanır:

```
MessageProducer producer = session.createProducer(destination);
```

destination parametresi, uygulamanın önceden oluşturduğu bir Kuyruk ya da Konu nesnesidir.

Bir uygulamanın ileti gönderebilmesi için bir ileti nesnesi yaratması gerekir. Bir iletinin gövdesinde uygulama verileri bulunur ve JMS beş tip ileti gövdeyi tanımlar:

- Bayt
- Eşlem
- Nesne
- Akış
- Metin

Her ileti gövdesi tipinin kendi JMS arabirimi vardır; bu arabirim, İleti arabiriminin bir alt arabirimidir ve Oturum arabirimindeki bir yöntemi, bu tip bir gövde ile birlikte yaratmak için kullanılan bir yöntemdir. Örneğin, bir metin iletisi için arabirim TextMessage olarak adlandırılır ve bir uygulama, aşağıdaki deyimde gösterildiği gibi, bir oturum nesnesinin createTextMessage () yöntemini kullanarak bir metin iletisi kullanır:

```
TextMessage outMessage = session.createTextMessage(outString);
```

İletiler ve ileti gövdeleriyle ilgili daha fazla bilgi için bkz. [“JMS ileti” sayfa 128](#).

Bir ileti göndermek için, aşağıdaki örnekte gösterildiği gibi, bir uygulama bir MessageProducer nesnesinin gönderme () yöntemini kullanır:

```
producer.send(outMessage);
```

Bir uygulama gönderme () yöntemini her iki ileti sistemi etki alanındaki iletileri göndermek için kullanabilir. Hedefin doğası, hangi ileti sistemi etki alanının kullanılacağını belirler. However, TopicPublisher, the sub-interface of MessageProducer that is specific to the publish/subscribe domain, also has a publish() method, which can be used instead of the send() method. İki yöntem işlevsel olarak aynıdır.

Bir uygulama, belirlenmiş hedefi olmayan bir MessageProducer nesnesi yaratabilir. Bu durumda, gönderme () yöntemini çağırırken uygulamanın hedefi belirtmesi gerekir.

Bir uygulama bir işlem içinde ileti gönderirse, hareket kesinleştirilinceye kadar bu ileti hedefine teslim edilmez. Başka bir deyişle, bir uygulama ileti gönderemez ve aynı hareket içinde iletiye yanıt alır.

Bir hedef, bir uygulama iletiye ileti gönderdiğinde, IBM MQ classes for JMS iletiyi iletir ve kuyruk yöneticisinin iletiyi güvenli bir şekilde alıp almadığını belirlemeden, denetimi yeniden uygulamaya geri döndürecek şekilde yapılandırılabilir. Bu, bazen *zamanuyumsuz koyma* olarak adlandırılır. Daha fazla bilgi için [“IBM MQ classes for JMS” da iletileri zamanuyumsuz olarak koyma](#) sayfa 297 başlıklı konuya bakın.

### ***JMS uygulamasında ileti alma***

Bir uygulama iletileri almak için bir ileti tüketicisi kullanır. Dayanıklı bir konu abonesi, tüketici etkinlik dışı olduğunda gönderilenler de dahil olmak üzere, bir hedefe gönderilen tüm iletileri alan bir ileti tüketicisi 'dir. Bir uygulama, bir ileti seçiciyi kullanarak almak istediği iletileri seçebilir ve ileti dinleyicisi kullanılarak zamanuyumsuz iletiler alabilir.

Bir uygulama, iletileri almak için bir MessageConsumer nesnesini kullanır. Bir uygulama, belirli bir hedef için bir MessageConsumer nesnesi yaratır; bu nesne, ileti tüketicisi kullanılarak alınan tüm iletilerin aynı hedeften alınması için bir kuyruk ya da konu olabilir. Bu nedenle, bir uygulamanın MessageConsumer nesnesi yaratabilmesi için önce bir Kuyruk ya da Konu nesnesi yaratması gerekir. Bir Kuyruk ya da Konu nesnesinin nasıl yaratılacağı hakkında bilgi için aşağıdaki konulara bakın:

- [“Bir JMS uygulamasındaki denetimli nesnelere almak için JNDI olanağını kullanma” sayfa 185](#)
- [“IBM JMS uzantılarını kullanma” sayfa 186](#)
- [“IBM MQ JMS uzantılarının kullanılması” sayfa 193](#)
- [“JMS uygulamasında hedef oluşturma” sayfa 198](#)

Bir uygulama, MessageConsumer nesnesi yaratmak için, aşağıdaki örnekte gösterildiği gibi, bir Oturum nesnesinin createConsumer() yöntemini kullanır:

```
MessageConsumer consumer = session.createConsumer(destination);
```

`destination` parametresi, uygulamanın önceden oluşturduğu bir Kuyruk ya da Konu nesnesidir.

Daha sonra, uygulama, aşağıdaki örnekte gösterildiği gibi, hedeften bir ileti almak için `MessageConsumer` nesnesinin `receive()` yöntemini kullanır:

```
Message inMessage = consumer.receive(1000);
```

`receive()` çağrısındaki parametre, hemen kullanılabilir bir ileti olmadığında, yöntemin uygun bir ileti için ne kadar süreyle bekleyeceğini belirtir. Bu parametreyi atlarsanız, uygun bir ileti gelene kadar arama blokları süresiz olarak ertelenmektedir. Uygulamanın bir ileti için beklemesini istemiyorsanız, bunun yerine `receiveNoWait()` yöntemini kullanın.

`receive()` yöntemi, belirli tipte bir ileti döndürür. Örneğin, bir uygulama bir metin iletisi aldığı anda, `receive()` çağrısının döndürdüğü nesne bir `TextMessage` nesnesi olur.

Ancak, bir `receive()` çağrısı tarafından döndürülen bildirilen nesne tipi bir ileti nesnesidir. Bu nedenle, yeni alınan bir iletinin gövdesinden verileri almak için, uygulamanın `Message` sınıfından daha belirli bir alt sınıfa (örneğin, `TextMessage`) dönüşümü gerekir. İletinin tipi bilinmiyorsa, uygulama tipini belirlemek için `instanceof` işlecini kullanabilir. Bir uygulamanın, dönüştürmeden önce bir iletinin tipini belirlemesi her zaman iyi bir uygulamadır; böylece hatalar düzgün bir şekilde işlenebilir.

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 işlem içinde ileti gönderirse, hareket kesinleştirilinceye kadar bu ileti hedefine teslim edilmez. Başka bir deyişle, bir uygulama ileti gönderemez ve aynı hareket içinde iletiye yanıt alır.

İleti tüketicisi, ileriye okuma için yapılandırılmış bir hedeften ileti alırsa, uygulama sona erdiğinde okuma öncesinde okuma arabelleğindeki kalıcı olmayan iletiler atılır.

Yayınlama/abone olma etki alanında, JMS iki tip ileti tüketicisi, kalıcı olmayan konu aboneleri ve dayanıklı konu aboneleri (aşağıdaki iki bölümde açıklanan) tanımlanır.

## Dayanıklı olmayan konu aboneleri

Kalıcı olmayan bir konu aboneleri yalnızca abone etkin durumdayken yayınlanan iletileri alır. Kalıcı olmayan abonelik, uygulama, dayanıklı olmayan bir konu aboneleri oluşturduğunda ve uygulama aboneleri kapattığında ya da abonelerin kapsamı dışlandığında sona erdiğinde başlar. IBM MQ classes for JMS' ta bir uzantı olarak, dayanıklı olmayan bir konu aboneleri de alıkonan yayınları alır.

Kalıcı olmayan bir konu aboneleri yaratmak için, uygulama etki alanı bağımsız `createConsumer()` yöntemini kullanabilir ve hedef olarak bir Konu nesnesi belirtebilirsiniz. Diğer bir seçenek olarak, bir uygulama etki alanını belirli bir `createSubscriber()` yöntemini kullanarak aşağıdaki örnekte gösterildiği gibi kullanabilir:

```
TopicSubscriber subscriber = session.createSubscriber(topic);
```

`topic` parametresi, uygulamanın önceden oluşturduğu bir Konu nesnesidir.

## Dayanıklı konu aboneleri

**Sınırlama:** Bir uygulama, bir aracıya gerçek zamanlı bağlantı kullanırken, dayanıklı konu aboneleri oluşturamaz.

Dayanıklı bir konu abonesi, kalıcı bir aboneliğin ömrü boyunca yayınlanan tüm iletileri alır. Bu iletiler, abone etkin olmamakla birlikte, yayınlananlar arasında yer alır. As an extension in IBM MQ classes for JMS, a durable topic subscriber also receives retained publications.

Bir uygulama, kalıcı bir konu abonesi oluşturmak için aşağıdaki örnekte gösterildiği gibi bir Oturum nesnesinin `createDurableSubscriber ()` yöntemini kullanır:

```
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001");
```

`createDurableSubscriber ()` çağrısında ilk parametre, uygulamanın daha önce yarattığı bir Konu nesnesidir ve ikinci parametre, sürekli aboneliği tanımlamak için kullanılan bir addır.

Kalıcı bir konu abonesi yaratmak için kullanılan oturumun ilişkili bir istemci tanıtıcısı olmalıdır. Bir oturumla ilişkili istemci tanıtıcısı, oturumu yaratmak için kullanılan bağlantıya ilişkin istemci tanıtıcısıyla aynıdır. İstemci tanıtıcısı, `ConnectionFactory` nesnesinin `CLIENTID` özelliği ayarlanarak belirlenebilir. Diğer bir seçenek olarak, bir uygulama, `ConnectionFactory` nesnesinin `setClientID ()` yöntemini çağırarak istemci tanıtıcısını belirtebilir.

Kalıcı bir aboneliği tanımlamak için kullanılan ad, yalnızca istemci tanıtıcısı içinde benzersiz olmalıdır; dolayısıyla, istemci tanıtıcısı, kalıcı bir aboneliğin tam, benzersiz tanıtıcısının bir parçası olur. Daha önce yaratılmış bir kalıcı aboneliği kullanmaya devam etmek için, bir uygulama, sürekli abonelikle ilişkilendirilmiş aynı istemci tanıtıcısına sahip bir oturum kullanarak ve aynı abonelik adını kullanarak, dayanıklı bir konu abonesi oluşturmalıdır.

Kalıcı abonelik, bir uygulama, şu anda hiçbir kalıcı aboneliği olmayan bir istemci tanıtıcısını ve abonelik adını kullanarak dayanıklı bir konu abonesi oluşturduğunda başlar. Ancak, kalıcı bir abonelik, uygulama dayanıklı konu abonesi kapatıldığında sona ermez. Bir uygulamanın, kalıcı aboneliği sona erdirmek için, kalıcı abonelikle ilişkilendirilmiş istemci tanıtıcısına aynı olan bir Oturum nesnesinin `unsubscribe ()` yöntemini çağırması gerekir. `unsubscribe ()` çağrısındaki parametre, aşağıdaki örnekte gösterildiği gibi abonelik adıdır:

```
session.unsubscribe("D_SUB_000001");
```

Dayanıklı bir aboneliğin kapsamı bir kuyruk yöneticidir. Bir kuyruk yöneticisinden kalıcı bir abonelik varsa ve başka bir kuyruk yöneticisine bağlı bir uygulama aynı istemci tanıtıcısı ve abonelik adına sahip kalıcı bir abonelik yarattıysa, iki kalıcı abonelik tamamen bağımsızdır.

## İleti seçicileri

Bir uygulama, `artArdaGelenAlma ()` çağrılarını tarafından yalnızca belirli ölçütlere uyan iletilerin döndürülmesini belirtebilir. Bir `MessageConsumer` nesnesi oluştururken, uygulama hangi iletilerin alınacağını belirleyen bir SQL (Yapılandırılmış Sorgu Dili) ifadesi belirleyebilir. Bu SQL ifadesine *ileti seçici* adı verilir. İleti seçici, JMS ileti üstbilgisi alanlarının adlarını ve ileti özelliklerini içerebilir. Bir ileti seçicinin nasıl oluşturulacağı hakkında bilgi için bkz. [“JMS’indeki ileti seçicileri” sayfa 128.](#)

Aşağıdaki örnekte, bir uygulamanın `myProp` adlı kullanıcı tanımlı bir özelliğe dayalı 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 seçicisiyle bir ileti tüketicisi yarattıktan sonra, ileti seçici o tüketicinin ömrü boyunca kalır. Bir uygulama birden çok ileti seçiciyi gerektiriyorsa, uygulamanın her ileti seçici için bir ileti tüketicisi yaratması gerekir.

Bir uygulama bir IBM WebSphere MQ 7 kuyruk yöneticisine bağlı olduğunda, bağlantı üreticisinin MSGSELECTION özelliğine ilişkin bir etki göstermemesine dikkat edin. Performansı en iyi duruma getirmek için, tüm ileti seçimi kuyruk yöneticisi tarafından gerçekleştirilir.

## Yerel yayınların engelleniyor

Bir uygulama, tüketicinin kendi bağlantısında yayınlanan yayınları yoksayan bir ileti tüketicisi yaratabilir. Uygulama, aşağıdaki örnekte gösterildiği gibi, bir createConsumer() çağrısında üçüncü parametreyi trueolarak ayarlanarak yapar:

```
MessageConsumer consumer = session.createConsumer(topic, null, true);
```

Bir createDurableSubscriber () çağrısında, aşağıdaki örnekte gösterildiği gibi, uygulama dördüncü parametreyi truedeğerine ayarlayarak bunu yapar.

```
String selector = "company = 'IBM'";
TopicSubscriber subscriber = session.createDurableSubscriber(topic, "D_SUB_000001",
    selector, true);
```

## İletilerin zamanuyumsuz teslim edilmesi

Bir uygulama, ileti tüketicisi bir ileti dinleyicisini kaydettirerek, iletileri zamanuyumsuz olarak alabilir. İleti dinleyicisinin onMessageadlı bir yöntemi vardır. Bu yöntem, uygun bir ileti varsa ve amacı iletiyi işlemek olan, zamanuyumsuz olarak çağrılır. Aşağıdaki kod mekanizmayı gösterir:

```
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, alma () çağrılarını zamanuyumlu olarak almak ya da ileti dinleyicilerini kullanarak zamanuyumsuz iletiler almak için bir oturumu kullanabilir, ancak her ikisi için de zamanuyumsuz bir oturum kullanabilir. Bir uygulamanın iletileri zamanuyumlu olarak ve zamanuyumsuz olarak alması gerekiyorsa, ayrı oturumlar yaratmalıdır.

Bir oturum, iletileri zamanuyumsuz olarak almak için ayarlandıktan sonra, o oturumda ya da o oturumdan yaratılan nesnelere ilgili olarak aşağıdaki yöntemler çağrılmaz:

- MessageConsumer.alma ()
- MessageConsumer.alma (uzun)
- MessageConsumer.receiveNoBekle ()

- Session.acknowledge()
- MessageProducer.gönderme (Hedef, İleti)
- MessageProducer.gönderme (Hedef, İleti, int, tamsayı, uzun)
- MessageProducer.gönderme (İleti)
- MessageProducer.gönderme (Message, int, int, long)
- MessageProducer.gönderme (Hedef, İleti, CompletionListener)
- MessageProducer.gönderme (Destination, Message, int, int, long, CompletionListener)
- MessageProducer.gönder (İleti, CompletionListener)
- MessageProducer.gönderme (İleti, int, int, uzun, 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(Serializable)
- 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 biri çağrılırsa, iletiyi içeren bir JMS kural dışı durumu (JMSException):

```
JMSCC0033: Bir oturum zamanuyumsuz olarak kullanıldığında zamanuyumlu bir yöntem çağrısına izin verilmez: 'method name'
```

atılır.

## Zehirli iletiler alınıyor

Bir uygulama işlenemeyen bir iletiyi alabilir. İletinin işlenememesinin birkaç nedeni olabilir; örneğin, iletinin biçimi yanlış olabilir. Bu tür iletiler, etkili bir şekilde işlenmesini önlemek için, zehirli iletiler olarak tanımlanır ve özel işleme gerektirir.

Zehirli iletilerin nasıl işleneceği konusunda ayrıntılı bilgi için bkz. [“IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 209.](#)

### **Abonelik kullanıcı verilerinin alınması**

IBM MQ classes for JMS uygulamasının bir kuyruktan tüketmekte olduğu iletiler, yönetimsel olarak tanımlanmış bir kalıcı abonelik tarafından konulursa, uygulamanın, abonelik ile ilişkili kullanıcı verileri bilgilerine erişmesi gerekir. Bu bilgi, iletiye özellik olarak eklenir.

Bir ileti, MQPS klasörüyle RFH2 üstbilgisi içeren bir kuyruktan tüketildiğinde, Sud tuşuyla ilişkili değer (varsa), IBM MQ classes for JMS uygulamasına döndürülen JMS ileti nesnesine bir String özelliği olarak eklenir. To enable the retrieval of this property from the message, the constant JMS\_IBM\_SUBSCRIPTION\_USER\_DATA in the JmsConstants interface can be used with the method javax.jms.Message.getStringProperty(java.lang.String) to get the subscription user data.

In the following example, an administrative durable subscription is defined by using the MQSC command **DEFINE SUB:**

```
DEFINE SUB('MY.SUBSCRIPTION') 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 ( MY . SUBSCRIPTION . Q ) yerleştirilir. The user data that is associated with the durable subscription is then added as a property to the message, which is stored in the MQPS folder of the RFH2 header with the key Sud.

IBM MQ classes for JMS uygulaması şunları arayabilir:

```
javax.jms.Message.getStringProperty(JmsConstants.JMS_IBM_SUBSCRIPTION_USER_DATA);
```

Daha sonra aşağıdaki Dizgi döndürülür:

```
Administrative durable subscription to put message to the queue MY.SUBSCRIPTION.Q
```

### **İlgili kavramlar**

[“MQRFH2 üstbilgisi ve JMS” sayfa 133](#)

### **İlgili görevler**

[Yönetimle ilgili abonelik tanımlanması](#)

### **İlgili başvurular**

[ALT](#)

[Arabirim JmsConstants](#)

### **IBM MQ classes for JMS uygulamasını kapatma**

Bir IBM MQ classes for JMS uygulamasının, belirli JMS nesnelerini durdurmadan önce belirtik olarak kapatması önemlidir. Finalizerler çağrılmayabilir, bu nedenle kaynaklara ücretsiz olarak güvenmeyin. Bir uygulamanın sıkıştırılmış izleme etkin ile sonlandırmasına izin verme.

Yalnızca bir uygulama, oturum düzeyinde ya da daha düşük düzeyde kısa ömürlü birçok JMS nesnesi yarattıysa, yalnızca çöp toplama işlemi tüm IBM MQ classes for JMS ve IBM MQ kaynaklarını zamanında yayınlamaz. Bu nedenle, bir uygulamanın bir Bağlantı, Oturum, MessageConsumerya da MessageProducer nesnesini kapatması artık gerekli olmadığında çok önemlidir.

Bir uygulama Connection 'ı kapatmadan sona ererse, tüm bağlantının hareket ettiği oturumlar için örtük bir geri alma gerçekleşir. Uygulama tarafından yapılan değişikliklerin kesinleştirildiğinden emin olmak için, uygulamayı kapatmadan önce bağlantıyı açık bir şekilde kapatın.

Do not use finalizers in an application to close JMS objects. Finaller çağrılmayacağından, kaynaklar serbest bırakılmayabilir. Bir Bağlantı kapatıldığında, bu bağlantı, içinden oluşturulan tüm Oturumları kapatır. Benzer şekilde, Oturum kapatıldığında bir oturumdan oluşturulan MessageConsumers ve MessageProducers kapatılır. Ancak, kaynakların zamanında serbest bırakılmasını sağlamak için, Oturum, MessageConsumersve MessageProducers ' u belirtik olarak kapatmayı düşünün.



İzleme sıkıştırması etkinleştirildiyse, System.Halt() kapanları ve olağandışı olağan dışı, denetimsiz JVM sonlandırmalarının bozuk bir izleme dosyasıyla sonuçlanabilir olması gerekir. Olanaklı olduğu durumlarda, gereksinim duyduğunuz izleme bilgilerini topladığınızda izleme olanağını kapatın. Bir uygulamayı olağan dışı bir uca izliyorsanız, sıkıştırılmamış izleme çıkışı kullanın.

**Not:** Bir kuyruk yöneticisinden bağlantıyı kesmek için, bir JMS uygulaması bağlantı nesnesindeki 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 işlemek 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 sorgulayın.
- 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 classes for JMS for IBM MQ 9.1.0 Fix Pack 1 ve öncesi için, ya da IBM MQ 9.1.1, yerel kuyruk ya da bir diğer ad kuyruğuna ilişkin hedef kuyruk bir küme kuyruğuyorsa, IBM MQ classes for JMS kuyruğu MQ00\_INPUT\_AS\_Q\_DEF, MQ00\_INQUIRE ve MQ00\_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 , MQ00\_INQUIRE ve MQ00\_FAIL\_IF QUIESCING açma 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.

- **V9.1.2** > **V9.1.0.2** IBM MQ classes for JMS for IBM MQ 9.1.0 Fix Pack 2 ve sonraki yayın düzeylerini ya da IBM MQ 9.1.2 ve sonraki yayın düzeylerini kullanırken, alıcı uygulamayı çalıştıran kullanıcının, kuyruğun tipine bakılmaksızın, yerel kuyrukta erişimi sorgulaması gerekir.

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 işlemleri 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 taşımaya 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 ileteler 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 işlemleri 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 iletelerin işlenmesi

Bir uygulama MessageListener aracılığıyla zamanuyumsuz olarak ileti alıyorsa, IBM MQ classes for JMS , ileti teslimini etkilemeden zehirli ileteleri 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 taşımaya 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 ileteler kuyruğuna koyma girişimi başarısız olursa, IBM MQ classes for JMS iletiyi giriş kuyruğuna döndürür.

Etkinleştirme belirtilerinin ve ConnectionConsumers özelliğinin zehirli ileteleri nasıl işlediğine ilişkin bilgi için [ASF ' de kuyruktan ileti kaldırılması](#) başlıklı 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 yansıtabilecek 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 yansıtabacak 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 311

Application Server Facilitis (Uygulama Sunucusu Tesisi) içinde, zehirli ileti işleme IBM MQ classes for JMSiçindeki başka bir yerde biraz farklı şekilde işlenir.

### **IBM MQ classes for JMSiçinde kural dışı durumlar**

Bir IBM MQ classes for JMS uygulaması, bir JMS API çağruları tarafından yayınlanan ya da bir kural dışı durum işleyicisine teslim edilen kural dışı durumları işleyebilmelidir.

IBM MQ classes for JMS , kural dışı durumlar yayınlayarak çalıştırma zamanı sorunlarını bildirir. JMSEException, JMS yöntemleri tarafından yayınlanan kural dışı durumlar için kök sınıftır ve JMSEException kural dışı durumlarını yakalamak, JMS ile ilgili tüm özel durumları işlemek için sosyal bir yol sağlar.

Her JMSEException kural dışı durumu aşağıdaki bilgileri sarsalıyor:

- Sağlayıcıya özgü bir kural dışı durum iletisi; bir uygulama, Throwable.getMessage() yöntemini çağırarak bu kural dışı durum iletisini alır.
- Sağlayıcıya özgü bir hata kodu; bir uygulama, JMSEException.getErrorCode() yöntemini çağırarak bu kodu alır.
- Bağlantılı bir kural dışı durum. Bir JMS API çağrısı tarafından yayınlanan bir kural dışı durum, genellikle bu kural dışı duruma bağlı başka bir kural dışı durum tarafından bildirilen daha düşük düzeyli bir sorunun sonucudur. Bir uygulama, JMSEException.getLinkedException() ögesini ya da Throwable.getCause() yöntemini çağırarak, bağlantılı bir kural dışı durumu alır.

IBM MQ classes for JMS tarafından verilen çoğu kural dışı durum, JMSEException alt sınıflarının örnekleridir. Bu alt sınıflar, aşağıdaki ek bilgileri sağlayan com.ibm.msg.client.jms.JmsExceptionDetail arabirimini uygular.

- Bir uygulamanın JmsExceptionDetail.getExplanation() yöntemini çağırarak ilgili olduğu kural dışı durum iletisine ilişkin açıklama.
- Bir uygulamanın JmsExceptionDetail.getUserAction() yöntemini çağırarak ilgili olduğu kural dışı duruma yanıt veren bir kullanıcı yanıtı.
- Kural dışı durum iletisinde ileti eklerinin anahtarları. Bir uygulama, JmsExceptionDetail.getKeys() yöntemini çağırarak, tüm tuşlar için bir yineleyici alır.
- İleti, kural dışı durum iletisine eklenir. Örneğin, bir ileti ekleme işlemi, kural dışı duruma neden olan kuyruğun adı olabilir ve bir uygulamanın bu ada erişebilmesinin yararlı olabileceği bir uygulama olabilir. Bir uygulama, JmsExceptionDetail.getValue() yöntemini çağırarak, belirtilen bir anahtara karşılık gelen ileti eklenmesini alır.

Herhangi bir ayrıntı yoksa, JmsExceptionAyrıntı arabirimindeki tüm yöntemler boş değer döndürebilir.

Örneğin, bir uygulama var olmayan bir IBM MQ kuyruğu için ileti üreticisi yaratmayı denerse, aşağıdaki bilgilerle bir kural dışı durum oluşur:

```
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.

Kural dışı durum yayınlandı, com.ibm.msg.client.jms.DetailedInvalidDestinationException, javax.jms.InvalidDestinationException sınıfının alt sınıfıdır ve com.ibm.msg.client.jms.JmsExceptionDetail arabirimini gerçekleştirir.

## Bağlantılı özel durumlar

Bağlantılı kural dışı durum, bir yürütme ortamı sorunuyla ilgili ek bilgi sağlar. Bu nedenle, yayınlanan her JMSEException kural dışı durumu için, bir uygulamanın bağlantılı kural dışı durumu denetmesi gerekir. Bağlantılı kural dışı durumun kendisi başka bir bağlantılı kural dışı duruma sahip olabilir ve bu nedenle bağlantılı kural dışı durumlar, özgün temel soruna geri giden bir zincir oluşturur. java.lang.Throwable sınıfının zincirleme kural dışı durum mekanizması kullanılarak bağlantılı bir kural dışı durum uygulandı ve bir uygulama, Throwable.getCause() yöntemini çağırarak bağlantılı bir kural dışı durumu alır. Bir JMSEException kural dışı durumu için, getLinkedException () yöntemi aslında Throwable.getCause() yönteminde yetki aktarır.

Örneğin, bir uygulama kuyruk yöneticisine bağlanırken yanlış bir kapı numarası belirtiyorsa, kural dışı durumlar aşağıdaki zinciri oluşturur:

```
com.ibm.msg.client.jms.DetailIllegalStateException
|
+- -->
com.ibm.mq.MQException
|
+- -->
com.ibm.mq.jmqi.JmqiException
|
+- -->
java.net.ConnectionException
```

Genellikle, bir zincirdeki her kural dışı durum, koddaki farklı bir katmandan atılır. Örneğin, yukarıdaki zincirdeki kural dışı durumlar aşağıdaki katmanlar tarafından atılır:

- JMSEException 'in alt sınıfının bir eşgörünümü olan ilk kural dışı durum, IBM MQ classes for JMS içindeki ortak katman tarafından atılır.
- Sonraki kural dışı durum, IBM MQ ileti sistemi sağlayıcısı tarafından com.ibm.mq.MQException örneği tarafından verilir.
- Bir sonraki kural dışı durum olan com.ibm.mq.jmqi.JmqiException eşgörünümü, ortak Java arabirimi tarafından MQI ' ye atılır.
- Son kural dışı durum, Java sınıf kitaplığı tarafından bir java.net.ConnectionException eşgörünümü yayınlanır.

For more information about the layered architecture of IBM MQ classes for JMS, see [JMS mimarisi için IBM MQ sınıfları](#).

Aşağıdaki kodla benzer kod kullanılarak, bir uygulama tüm uygun bilgileri ayıklamak için bu zincirden geçerek yinelenabilir:

```
import com.ibm.msg.client.jms.JmsExceptionDetail;
import com.ibm.mq.MQException;
import com.ibm.mq.jmqi.JmqiException;
import javax.jms.JMSEException;
.
.
.
catch (JMSEException je) {
System.err.println("Caught JMSEException");

// Check for linked exceptions in JMSEException
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 JMSEException) {
    JMSEException je1 = (JMSEException) 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 jmque = (JmqiException)t;
    System.err.println("WMQ Log Message: " + jmque.getWmqLogMessage());
    System.err.println("WMQ Explanation: " + jmque.getWmqMsgExplanation());
    System.err.println("WMQ Msg Summary: " + jmque.getWmqMsgSummary());
    System.err.println("WMQ Msg User Response: "
    + jmque.getWmqMsgUserResponse());
    System.err.println("WMQ Msg Severity: " + jmque.getWmqMsgSeverity());
    }

// Get the next cause
t = t.getCause();
}
}
}

```

Bir uygulamanın bir zincirdeki her bir kural dışı durumun tipini her zaman denetleyeceğini unutmayın; kural dışı durum tipi farklı tipteki bilgileri sarmalayan farklı tiplerdeki kural dışı durumlar ve kural dışı durumlar olabilir.

## IBM MQ ile ilgili bir sorunla ilgili özel bilgilerin alınması

Instances of `com.ibm.mq.MQException` and `com.ibm.mq.jmqi.JmqiException` encapsulate IBM MQ specific information about a problem.

Bir `MQException` kural dışı durumu, aşağıdaki bilgileri sarmalıyor:

- Bir uygulama, `getCompCode()` yöntemini çağırarak bir uygulama tarafından elde edilen bir kod
- Bir uygulama, bir uygulamanın `getReason()` yöntemini çağırarak edindiği bir neden kodu

Bir `JmqiException` kural dışı durumu, bir tamamlanma kodunu ve bir neden kodunu da sarsalıyor. Ancak ek olarak, bir `JmqiException` kural dışı durumu, bir AMQ *nnnn* ya da CSQ *nnnn* iletişimde, kural dışı durumla ilişkiliyse, bilgileri sarsalıyor. Bir uygulama, kural dışı durumun uygun yöntemlerini çağırarak, bu iletinin çeşitli bileşenlerini (önem derecesi, açıklama ve kullanıcı yanıtı gibi) alabilir.

Bu kısımda sözü edilen yöntemlerin kullanılmasına ilişkin örnekler için, [“Bağlantılı özel durumlar” sayfa 213](#) içindeki örnek kodlara bakın.

## Önceki IBM MQ classes for JMS sürümlerinden yükseltme

Compared to previous versions of IBM MQ classes for JMS, most error codes and exception messages changed in IBM WebSphere MQ 7.0. Bu değişikliklerin nedeni, IBM WebSphere MQ 7.0'ten IBM MQ classes for JMS ' in katmanlı bir mimariye sahip olması ve koddaki farklı katmanlardan kural dışı durumlar atılmasına neden olur.

For example, if an application tries to connect to a queue manager that does not exist, a previous version of IBM MQ classes for JMS threw a `JMSEException` exception with the following information:

```
MQJMS2005: Failed to create MQQueueManager for 'localhost:QM_test'.
```

Bu kural dışı durum, aşağıdaki bilgilerle bağlantılı bir `MQException` kural dışı durumu içeriyordu:

```
MQJE001: Completion Code 2, Reason 2058
```

Aynı koşullarda karşılaştırmaya göre, IBM MQ classes for JMS sürümü 7.0 , aşağıdaki bilgilerle JMSEException kural dışı durumunu yayınlıyor:

```
Message : JMSWMQ0018: Failed to connect to queue manager 'QM_test' with
connection mode 'Client' and host name 'localhost'.
Class : class com.ibm.msg.client.jms.DetailedJMSEException
Error Code : JMSWMQ0018
Explanation : null
User Action : Check the queue manager is started and if running in client mode,
check there is a listener running. Please see the linked exception
for more information.
```

Bu kural dışı durum, aşağıdaki bilgilerle bağlantılı bir MQException kural dışı durumu içeriyor:

```
Message : JMSMQ0001: IBM MQ call failed with compcode '2' ('MQCC_FAILED')
reason '2058' ('MQRC_Q_MGR_NAME_ERROR').
Class : class com.ibm.mq.MQException
Completion Code : 2
Reason Code : 2058
```

If your application parses or tests exception messages returned by the Throwable.getMessage() method, or error codes returned by the JMSEException.getErrorCode() method, and you are upgrading from a release before IBM WebSphere MQ 7.0, your application probably needs to be modified in order to use version 7.0 or later of IBM MQ classes for JMS.

## Kural dışı durum

Bir uygulama, bir kural dışı durum dinleyicisini bir Bağlantı nesnesiyle kaydettirebilir. Subsequently, if a problem occurs that makes the connection unusable, IBM MQ classes for JMS delivers an exception to the exception listener by calling its onException() method. Uygulama, daha sonra bağlantıyı yeniden oluşturma olanağına sahiptir. IBM MQ classes for JMS , zamanuyumsuz bir iletiyi teslim etme girişimi sırasında bir sorun ortaya çıkarsa, kural dışı durum dinleyicisine de bir özel durum sağlayabilir.

Bir JMS MessageListener ve bir JMS ExceptionListener yapılandırılan geçerli JMS uygulamaları için ve IBM MQ classes for JMS 'in JMS belirtimiyle tutarlı olduğundan emin olmak için IBM MQ 8.0.0 Fix Pack 2' dan, IBM MQ classes for JMS için ASYNC\_EXCEPTIONS JMS ConnectionFactory özelliğinin varsayılan değeri ASYNC\_EXCEPELS\_CONNECTIONRENMIR olarak değiştirilir. Sonuç olarak, varsayılan olarak, bir uygulamanın JMS ExceptionListener' a yalnızca bozuk bağlantı hata kodlarına karşılık gelen istisnalar verilir.

IBM MQ 9.0.0 Fix Pack 1 içinde yer alan [APAR IT14820](#), şunları güncelleştirir: IBM MQ classes for JMS .

- Bir uygulama tarafından kaydedilen bir ExceptionListener , uygulamanın zamanuyumlu ya da zamanuyumsuz ileti tüketicileri kullanıp kullanmadığı dikkate alınmaksızın, herhangi bir bağlantı bozuk kural dışı durumu için çağrılır.
- Bir uygulama tarafından kaydedilen bir ExceptionListener , JMS oturumu tarafından kullanılan bir TCP/IP yuvası bozursa çağrılır.
- Uygulama zamanuyumsuz ileti tüketicilerini kullanırken ve uygulama tarafından kullanılan JMS ConnectionFactory 'un ASYNC\_EXCEPTIONS özelliği ASYNC\_EXCEPTIONS\_ALL değerine ayarlandığı zaman, ileti teslimi sırasında ortaya çıkan bağlantısız kural dışı durumlar (örneğin, MQRC\_GET\_INHIBITED) bir uygulamanın ExceptionListener ' e teslim edilir.

**Not:** An ExceptionListener is only invoked once for a connection broken exception, even if two TCP/IP connections (one used by a JMS Connection and one used by a JMS Session) are broken.

Diğer herhangi bir sorun için, geçerli JMS API çağrısı tarafından JMSEException kural dışı durumu yayınlanır.

If an application does not register an exception listener with a Connection object, any exceptions that would have been delivered to the exception listener are written to the IBM MQ classes for JMS for JMS log.

## İlgili başvurular

JMS için IBM MQ sınıfları

ASYNCEXCEPTION

### **Accessing IBM MQ features from an IBM MQ classes for JMS application**

IBM MQ classes for JMS provides facilities to exploit a number of features of IBM MQ.



**Uyarı:** Bu özellikler JMS belirtiminin dışındadır ya da bazı durumlarda JMS belirtimini ihlal eder. Bunları kullanırsanız, uygulamanızın diğer JMS sağlayıcılarıyla uyumlu olma olasılığının düşük olduğunu da sağlar. JMS belirtimine uygun olmayan özellikler, bir Attention (Dikkat) bildirimini ile etiketlenir.

#### *Reading and writing the message descriptor from an IBM MQ classes for JMS application*

Bir Hedef ve İleti üzerinde özellikleri ayarlayarak, ileti tanımlayıcısına (MQMD) erişme yeteneğini kontrol edin.

Bazı IBM MQ uygulamalarının, kendilerine gönderilen iletilerin MQMD ' de ayarlanması için belirli değerler olması gerekir. IBM MQ classes for JMS provides message attributes that allow JMS applications to set MQMD fields and so enable JMS applications to "drive" IBM MQ applications.

MQMD özelliklerinin herhangi bir etkisine sahip 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 Sürüm dışında tüm MQMD alanları gösterilir; BackoutCount okunabilir ancak yazılamaz.

Bu örnek, MQMD.UserIdentifier , "JoeBloggs" olarak ayarlanır.

```
// 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\_UserIdentifier' ı ayarlamadan önce WMQ\_MQMD\_MESSAGE\_CONTEXT ayarını ayarlamamız gerekir. WMQ\_MQMD\_MESSAGE\_CONTEXT kullanımı hakkında daha fazla bilgi için bkz. [“JMS ileti nesnesi özellikleri” sayfa 219](#).

Benzer şekilde, bir ileti almadan önce WMQ\_MQMD\_READ\_ENABLED ' i true değerine ayarlayarak ve daha sonra iletinin alma yöntemlerini ( getStringÖzelliği gibi) kullanarak MQMD alanlarının içeriğini alabilirsiniz. Alınan özellikler salt okunurdur.

Bu örnek, bir iletinin MQMD.ApplIdentityData alanının değerini, bir kuyruktan ya da bir konudan elde edilen *değer* alanı ile 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 = icvMsg.getStringProperty("JMS_IBM_MQMD_AppIdentityData");
```

### JMS hedef nesne özellikleri

Hedef nesnesinin iki özelliği, JMS'tan MQMD' ye erişimi ve bir üçüncü denetim iletisi bağlamını denetler.

Çizelge 35. Özellik adları ve açıklamaları

Özellik	Kısa Biçim	Tanım
WMQ_MQMD_WRITE_ENABLED	MDC	Bir JMS uygulamasının MQMD alanlarının değerlerini ayarlayıp ayarlayamayacağı
WMQ_MQMD_READ_ENABLED	MDR	Bir JMS uygulamasının MQMD alanlarının değerlerini ayıklayıp çıkaramayacağı
WMQ_MQMD_MESSAGE_BAĞLAMı	MDCTX	What level of message context is to be set by the JMS application. Uygulama, bu özelliğin yürürlüğe girmesi için uygun bağlam yetkilisiyle çalışıyor olmalıdır.

Çizelge 36. Özellik adları, değerler ve küme yöntemleri

Özellik	Denetim aracındaki geçerli değerler (koyu olarak varsayılan değerler)	Programlar daki geçerli değerler	Set yöntemi
WMQ_MQMD_WRITE_ENABLED	<ul style="list-style-type: none"><li>• <b>NO</b> Tüm JMS_IBM_MQMD* özellikleri yoksayılr ve değerleri, temeldeki MQMD yapısıyla kopyalanmaz.</li><li>• <b>YES</b> JMS_IBM_MQMD* özellikleri işlendi. Değerleri, temeldeki MQMD yapısına kopyalanır.</li></ul>	<ul style="list-style-type: none"><li>• <b>Yanlış</b></li><li>• <b>Doğru</b></li></ul>	setMQMDWriteEtkinleştirildi

Çizelge 36. Özellik adları, değerler ve küme yöntemleri (devamı var)

Özellik	Denetim aracındaki geçerli değerler (koyu olarak varsayılan değerler)	Programlar daki geçerli değerler	Set yöntemi
WMQ_MQMD_READ_ENABLED	<ul style="list-style-type: none"> <li>• <b>NO</b> İletileri gönderirken, gönderilen bir iletteki JMS_IBM_MQMD* özellikleri, MQMD ' deki güncellenen alan değerlerini yansıtacak şekilde güncellenmez. İleti alınırken, gönderenin bir kısmını ya da tümünü ayarlasa bile, alınan bir iletide JMS_IBM_MQMD* özelliklerinin hiçbiri kullanılabilir değil.</li> <li>• <b>YES</b> İletileri gönderirken, gönderilen bir iletteki tüm JMS_IBM_MQMD* özellikleri, gönderenin belirtik olarak ayarlanmamış olması da içinde olmak üzere, MQMD ' deki güncellenmiş alan değerlerini yansıtacak şekilde güncellenir. İleti alınırken, tüm JMS_IBM_MQMD* özellikleri, gönderenin belirtik olarak ayarlamayanlar da içinde olmak üzere, alınan bir iletlerde kullanılabilir.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Yanlış</b></li> <li>• Doğru</li> </ul>	setMQMDReadEtkinleştirildi
WMQ_MQMD_MESSAGE_CONTEXT	<ul style="list-style-type: none"> <li>• <b>VARSAYILAN</b> MQOPER API çağrısı ve MQPMO yapısı belirtik ileti bağlamı seçeneklerini belirtmiyor</li> <li>• <b>SET_IDENTITY_CONTEXT</b> MQOPEN API çağrısı, MQOO_SET_IDENTITY_CONTEXT ileti bağlamı seçeneğini belirtir ve MQPMO yapısı MQPMO_SET_IDENTITY_CONTEXT belirtisini belirtir</li> <li>• <b>SET_ALL_CONTEXT</b> MQOPEP API çağrısı, MQOO_SET_ALL_CONTEXT ileti bağlamı seçeneğini belirtir ve MQPMO yapısı MQPMO_SET_ALL_CONTEXT ' yi belirtir</li> </ul>	<ul style="list-style-type: none"> <li>• <b>WMQ_MD CTX_VAR SAYILAN</b></li> <li>• WMQ_MD CTX_SET_IDENTITY_CONTEXT</li> <li>• WMQ_MD CTX_SET_ALL_CONTEXT</li> </ul>	setMQMDMessageBağlam

## JMS ileti nesnesi özellikleri

İleti nesnesi özellikleri önekli JMS\_IBM\_MQMD, ilgili MQMD alanını ayarlamanıza ya da okumanıza olanak sağlar.

## İletilerin gönderilmesi

StrucId ve Version dışındaki tüm MQMD alanları temsil edilir. Bu özellikler yalnızca MQMD alanlarına gönderme yapar; hem MQMD ' de hem de MQRFH2 üstbilgisinde bir özellik oluşur, MQRFH2 içindeki sürüm belirlenmez ya da çıkarılmaz.

JMS\_IBM\_MQMD\_BackoutCount'da, bu özelliklerden herhangi biri ayarlanabilir.

JMS\_IBM\_MQMD\_BackoutCount için ayarlanan herhangi bir değer yok sayılır.

Bir özelliğin uzunluk üst sınırı varsa ve siz çok uzun bir değer sağlıyorsa, değer kesilir.

Bazı özellikler için, Hedef nesnede WMQ\_MQMD\_MESSAGE\_CONTEXT özelliğini de ayarlamanız gerekir.

Uygulamanın, bu özelliğin geçerli olması için uygun bağlam yetkisi ile çalışıyor olması gerekir.

WMQ\_MQMD\_MESSAGE\_CONTEXT değerini uygun bir değere ayarlamadıysanız, özellik değeri yoksayıdır.

WMQ\_MQMD\_MESSAGE\_CONTEXT uygun bir değere ayarlıysa, ancak kuyruk yöneticisi için yeterli bağlam yetkisine sahip değilseniz, JMSEException yayınlanır. WMQ\_MQMD\_MESSAGE\_CONTEXT ile ilgili belirli değerleri gerektiren özellikler aşağıdaki gibidir.

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

## İletileri alma

WMQ\_MQMD\_READ\_ENABLED özelliği true değerine ayarlıysa, üreten uygulamanın ayarlı olduğu gerçek özelliklerden bağımsız olarak tüm bu özellikler alındı iletisi üzerinde kullanılabilir. Bir uygulama, JMS belirtimine göre önce tüm özellikler temizlenmedikçe, alınan bir iletinin özelliklerini değiştiremez. Alınan ileti, özellikler değiştirilmeden iletilebilir.



**Uyarı:** Uygulamanız WMQ\_MQMD\_READ\_ENABLED özelliği true değerine ayarlanmış bir hedeften bir ileti alırsa ve onu WMQ\_MQMD\_WRITE\_ENABLED değerine ayarlanmış bir hedefe iletirse, alınan iletinin tüm MQMD alanı değerleri, iletilen iletiye kopyalanmakta olan tüm MQMD alan değerlerinde sonuçlanır.

## Özellikler tablosu





Bu çizelge, MQMD alanlarını temsil eden ileti nesnesinin özelliklerini listeler. Alanların ve izin verilen değerlerinin tam açıklamaları için bağlantılara bakın.

Çizelge 37. Özellik adları, açıklamalar ve tipler			
Özellik	Tanım	Java Tip	Tam açıklamayla bağlantı
JMS_IBM_MQMD_Report	Rapor iletileri için seçenekler	Tamsayı	<a href="#">Rapor</a>

Çizelge 37. Özellik adları, açıklamalar ve tipler (devamı var)			
Özellik	Tanım	Java Tip	Tam açıklamayla bağlantı
JMS_IBM_MQMD_MsgType	İleti tipi	Tamsayı	<a href="#">MsgType</a>
JMS_IBM_MQMD_Expiry	İleti kullanım süresi	Tamsayı	Son kullanma tarihi
JMS_IBM_MQMD_Feedback	Geribildirim ya da neden kodu	Tamsayı	<a href="#">Geribildirim</a>
JMS_IBM_MQMD_Kodlaması	İleti verilerinin sayısal kodlaması	Tamsayı	<a href="#">Kodlama</a>
JMS_IBM_MQMD_CodedCharSetId	İleti verilerinin karakter kümesi tanıttıcısı	Tamsayı	<a href="#">CodedCharSetId</a>
JMS_IBM_MQMD_Format	İleti verilerinin adını biçimle	Dizgi	<a href="#">Biçim</a>
JMS_IBM_MQMD_Priority <sup>1</sup>	İleti önceliği	Tamsayı	<a href="#">Öncelik</a>
JMS_IBM_MQMD_Persistence	İleti kalıcılığı	Tamsayı	<a href="#">Kalıcılık</a>
JMS_IBM_MQMD_MsgId <sup>2</sup>	İleti Tanıtıcısı	Nesne (byte []) <sup>4</sup>	<a href="#">MsgId</a>
JMS_IBM_MQMD_CorrelId <sup>3</sup>	İlinti tanıtıcısı	Nesne (byte []) <sup>4</sup>	<a href="#">CorrelId</a>
JMS_IBM_MQMD_BackoutCount	Geriletme sayacı	Tamsayı	<a href="#">BackoutCount</a>
JMS_IBM_MQMD_ReplyToQ	Yanıt kuyruğunun adı	Dizgi	<a href="#">ReplyToQ</a>
JMS_IBM_MQMD_ReplyToQMgr	Yanıt kuyruğu yöneticisinin adı	Dizgi	<a href="#">ReplyToQMgr</a>
JMS_IBM_MQMD_UserIdentifier	Kullanıcı kimliği	Dizgi	<a href="#">UserIdentifier</a>
JMS_IBM_MQMD_AccountingToken	Hesap simgesi	Nesne (byte []) <sup>4</sup>	<a href="#">AccountingToken</a>
JMS_IBM_MQMD_ApplIdentityVerileri	Kimlikle ilgili uygulama verileri	Dizgi	<a href="#">ApplIdentityVerileri</a>
JMS_IBM_MQMD_PutApplTipi	İletiyi koyan uygulamanın tipi	Tamsayı	<a href="#">PutApplTipi</a>
JMS_IBM_MQMD_PutApplAdı	İletiyi koyan uygulamanın adı	Dizgi	<a href="#">PutApplAdı</a>
JMS_IBM_MQMD_PutDate	İletin konulduğu tarih	Dizgi	<a href="#">PutDate</a>
JMS_IBM_MQMD_PutTime	İletin konulduğu saat	Dizgi	<a href="#">PutTime</a>
JMS_IBM_MQMD_ApplOriginVerileri	Köken ile ilgili uygulama verileri	Dizgi	<a href="#">ApplOriginVerileri</a>
JMS_IBM_MQMD_GroupId	Grup tanıtıcısı	Nesne (byte []) <sup>4</sup>	<a href="#">GroupId</a>
JMS_IBM_MQMD_MsgSeqNumarası	Grup içindeki mantıksal iletinin sıra numarası	Tamsayı	<a href="#">MsgSeqNumarası</a>

Çizelge 37. Özellik adları, açıklamalar ve tipler (devamı var)

Özellik	Tanım	Java Tip	Tam açıklamayla bağlantı
JMS_IBM_MQMD_Görelİ Konum	Mantıksal iletinin başlangıcındaki fiziksel iletilerde verilerin görelİ konumu	Tamsayı	<a href="#">Görelİ Konum</a>
JMS_IBM_MQMD_MsgFlags	İleti İşaretleri	Tamsayı	<a href="#">MsgFlags</a>
JMS_IBM_MQMD_OriginalLength	Özgün iletinin uzunluğu	Tamsayı	<a href="#">OriginalLength</a>

-  **Uyarı:** JMS\_IBM\_MQMD\_Priority değerine 0-9 aralığında olmayan bir değer atarsanız, bu, JMS belirtimini ihlal eder.
-  **Uyarı:** JMS belirtimi, ileti tanıtıcısının JMS sağlayıcısı tarafından ayarlanması gerektiğini ve bunun benzersiz ya da boş değer olması gerektiğini belirtir. JMS\_IBM\_MQMD\_MsgId değerine bir değer atarsanız, bu değer JMSMessageID'ye kopyalanır. Bu nedenle, JMS sağlayıcısı tarafından ayarlanmamış ve benzersiz olmayabilir: bu, JMS belirtimini ihlal eder.
-  **Uyarı:** JMS\_IBM\_MQMD\_CorrelId değerine, 'ID:' dizisiyle başlayan bir değer atarsanız, bu, JMS belirtimini ihlal eder.
-  **Uyarı:** Bir iletideki bayt dizisi özelliklerinin kullanımı, JMS belirtimini ihlal eder.

#### IBM MQ classes for JMS kullanarak bir uygulamadan IBM MQ ileti verilerine erişilmesi

Tam IBM MQ ileti verilerine IBM MQ classes for JMS kullanarak bir uygulama içinde erişebilirsiniz. Tüm verilere erişmek için, iletinin bir JMSBytesMessage olması gerekir. JMSBytesMessage gövdesinde herhangi bir MQRFH2 üstbilgisi, diğer IBM MQ üstbilgileri ve aşağıdaki ileti verileri bulunur.

Set the WMQ\_MESSAGE\_BODY property of the destination to WMQ\_MESSAGE\_BODY\_MQ, to receive all the message body data in the JMSBytesMessage.

WMQ\_MESSAGE\_BODY, WMQ\_MESSAGE\_BODY\_JMS ya da WMQ\_MESSAGE\_BODY\_UNSPECIFIED olarak ayarlanırsa, ileti gövdesi JMS MQRFH2 üstbilgisi olmadan döndürülür ve JMSBytesMessage'ın özellikleri, RFH2 içindeki özellikleri yansıtır.

Bazı uygulamalar bu konuda açıklanan işlevleri kullanamıyor. If an application is connected to an IBM MQ V6 queue manager, or if it has set SAĞLAMA Sü to 6, the functions are not available.

### İleti gönderilmesi

İletiler gönderilirken hedef özellik, WMQ\_MESSAGE\_BODY, WMQ\_TARGET\_CLIENT' a göre öncelik kazanır.

WMQ\_MESSAGE\_BODY, WMQ\_MESSAGE\_BODY\_JMS olarak ayarlanırsa, IBM MQ classes for JMS otomatik olarak JMSMessage özellikleri ve üstbilgi alanları ayarlarına dayalı olarak bir MQRFH2 üstbilgisi oluşturur.

WMQ\_MESSAGE\_BODY seçeneği WMQ\_MESSAGE\_BODY\_MQ olarak ayarlandıysa, ileti gövdesine ek üstbilgi eklenmez.

WMQ\_MESSAGE\_BODY, WMQ\_MESSAGE\_BODY\_UNSPECIFIED olarak ayarlandıysa, IBM MQ classes for JMS bir MQRFH2 üstbilgisi gönderir ( WMQ\_TARGET\_CLIENT, WMQ\_TARGET\_DEST\_MQ olarak ayarlandıysa). On receive, setting WMQ\_TARGET\_CLIENT to WMQ\_TARGET\_DEST\_MQ results in any MQRFH2 being removed from the message body.

**Not:** JMSBytesMessage ve JMSTextMessage, JMSStreamMessage, JMSMapMessage, andve JMSObjectMessage gibi bir MQRFH2 gerektirmez.

WMQ\_MESSAGE\_BODY\_UNSPECIFIED, WMQ\_MESSAGE\_BODY için varsayılan ayardır ve WMQ\_TARGET\_DEST\_JMS, WMQ\_TARGET\_CLIENT için varsayılan ayardır.

Bir JMSBytesMessage gönderdiğinizde, 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, önceki bir WebSphere MQ üstbilgisi yoksa, JMS ileti gövdeyi başlatan IBM MQ üstbilgisinin ya da uygulama bilgi yükünün biçimini belirtir.
- JMS\_IBM\_Character\_Set ya da JMS\_IBM\_MQMD\_CodedCharSetId: Bu özellik, önceki bir WebSphere MQ üstbilgisi yoksa, JMS ileti gövdeyi başlatan IBM MQ üstbilgisinin ya da uygulama bilgi yükünün CCSID değerini belirtir.
- JMS\_IBM\_Encoding ya da JMS\_IBM\_MQMD\_Encoding: Bu özellik, önceki bir WebSphere MQ üstbilgisi yoksa, JMS ileti gövdesine başlayan IBM MQ üstbilgi ya da uygulama bilgi yükünün kodlamasını belirtir.

If both types of property are specified, the JMS\_IBM\_MQMD\_\* properties override the corresponding JMS\_IBM\_\* properties, as long as the destination property WMQ\_MQMD\_WRITE\_ENABLED is set to true.

JMS\_IBM\_MQMD\_\* ve JMS\_IBM\_\* kullanılarak ileti özellikleri ayarı arasında yürürlükte olan farklar önemlidir:

1. JMS\_IBM\_MQMD\_\* özellikleri, IBM MQ JMS sağlayıcısına özeldir.
2. JMS\_IBM\_MQMD\_\* özellikleri yalnızca MQMD içinde ayarlanır. JMS\_IBM\_\* properties are set in the MQMD only if the message does not have an MQRFH2 JMS header. Aksi takdirde bunlar JMS RFH2 üstbilgisinde ayarlanır.
3. JMS\_IBM\_MQMD\_\* özelliklerinin, bir JMSMessage içine yazılan metin ve sayıların kodlamasını etkilemez.

Alma uygulaması büyük olasılıkla, MQMD.Encoding ve MQMD.CodedCharSetId değerlerinin, ileti gövdesindeki sayı ve metin kümesi kodlamasına ve karakter kümesine karşılık geldiğini varsayar. JMS\_IBM\_MQMD\_\* özellikleri kullanılırsa, bunu yapmak için gönderme uygulamasının sorumluluğunda olur. İleti gövdesindeki sayı ve metin kodlaması ve karakter kümesi, JMS\_IBM\_\* özellikleri tarafından ayarlanır.

The badly coded snippet in [Şekil 45 sayfa 222](#) sends a message encoded in character set 1208, with MQMD.CodedCharSetId set to 37.

#### a. Yanlış kodlanmış iletiyi 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. Receiving the message, relying on the value of JMS\_IBM\_XX\_ENCODE\_CASE\_ONE charter\_set set by the value of MQMD.CodedCharSetId:

```
TextMessage tmi = (TextMessage) cons.receive();
System.out.println("Message is \"" + tmi.getText() + "\"");
```

#### c. Sonuç çıkışı:

```
Message is "éÈÈ'>...??>?"
```

*Şekil 45. Sürekli olarak kodlanan MQMD ve ileti verileri*

Şekil 46 sayfa 223 içindeki kodun parçacıklarından biri, otomatik olarak oluşturulan bir MQRFH2 üstbilgisi eklenmeden, uygulamanın bilgi yükünü içeren gövdesi ile bir kuyruğa ya da konuya eklenmekte olan bir iletiyle sonuçlanıyor.

---

## 1. Setting WMQ\_MESSAGE\_BODY\_MQ:

```
((MQDestination) destination).setMessageBodyStyle  
(WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

## 2. Setting WMQ\_TARGET\_DEST\_MQ:

```
((MQDestination) destination).setMessageBodyStyle  
(WMQConstants.WMQ_MESSAGE_BODY_UNSPECIFIED);  
((MQDestination) destination).  
setTargetClient(WMQConstants.WMQ_TARGET_DEST_MQ);
```

Şekil 46. MQ ileti gövdesiyle bir ileti gönderin.

---

## İleti alma

WMQ\_MESSAGE\_BODY seçeneği WMQ\_MESSAGE\_BODY\_JMSolarak ayarlandıysa, gelen JMS ileti tipi ve gövdesi, alınan WebSphere MQ iletinin içeriği tarafından belirlenir. The message type and body are determined by fields in the MQRFH2 header, or in the MQMD, if there is no MQRFH2.

WMQ\_MESSAGE\_BODY seçeneği WMQ\_MESSAGE\_BODY\_MQolarak ayarlandıysa, gelen JMS ileti tipi JMSBytesMessageolur. JMS ileti gövdesi, temeldeki MQGET API çağrısı tarafından döndürülen ileti verileridir. İleti gövdesinin uzunluğu, MQGET çağrısı tarafından döndürülen uzunluktur. İleti gövdesindeki verilerin karakter takımı ve kodlaması, MQMD' un CodedCharSetId ve Kodlama alanları tarafından belirlenir. İleti gövdesindeki verilerin biçimi, MQMD' un Biçim alanı tarafından belirlenir.

WMQ\_MESSAGE\_BODY seçeneği WMQ\_MESSAGE\_BODY\_UNSPECIFIEDolarak ayarlandıysa, varsayılan değer olan IBM MQ classes for JMS , bunu WMQ\_MESSAGE\_BODY\_JMSolarak ayarlar.

Bir JMSBytesMessagealdığınızda, aşağıdaki özelliklere başvurarak kodu çözebilirsiniz:

- JMS\_IBM\_Format ya da JMS\_IBM\_MQMD\_Format: Bu özellik, önceki bir WebSphere MQ üstbilgisi yoksa, JMS ileti gövdeyi başlatan IBM MQ üstbilgisinin ya da uygulama bilgi yükünün biçimini belirtir.
- JMS\_IBM\_Character\_Set ya da JMS\_IBM\_MQMD\_CodedCharSetId: Bu özellik, önceki bir WebSphere MQ üstbilgisi yoksa, JMS ileti gövdeyi başlatan IBM MQ üstbilgisinin ya da uygulama bilgi yükünün CCSID değerini belirtir.
- JMS\_IBM\_Encoding ya da JMS\_IBM\_MQMD\_Encoding: Bu özellik, önceki bir WebSphere MQ üstbilgisi yoksa, JMS ileti gövdesine başlayan IBM MQ üstbilgi ya da uygulama bilgi yükünün kodlamasını belirtir.

The following code snippet results in a received message that is a JMSBytesMessage. Alınan iletinin içeriğinden ve alınan MQMD' ın biçim alanının içeriğinden bağımsız olarak, ileti bir JMSBytesMessageilettir.

```
((MQDestination)destination).setMessageBodyStyle  
(WMQConstants.WMQ_MESSAGE_BODY_MQ);
```

### Hedef özelliği WMQ\_MESSAGE\_BODY

WMQ\_MESSAGE\_BODY determines whether a JMS application processes the MQRFH2 of an IBM MQ message as part of the message payload (that is, as part of the JMS message body).

Çizelge 38. Özellik adları ve açıklamaları

Özellik	Kısa Biçim	Tanım
WMQ_MESSAGE_BODY	GÖVDE	Whether a JMS application processes the MQRFH2 of an IBM MQ message as part of the message payload (that is, as part of the JMS message body).

Çizelge 39. Özellik adları, değerler ve küme yöntemleri

Özellik	Denetim aracındaki geçerli değerler (koyu olarak varsayılan değerler)	Programlardaki geçerli değerler	Set yöntemi
WMQ_MESSAGE_BODY	<ul style="list-style-type: none"> <li>• <b>Belirtilmedi</b> When sending, IBM MQ classes for JMS does or does not generate and include an MQRFH2 header, depending on the value of WMQ_TARGET_CLIENT. Alma sırasında, JMSdeğeri olarak hareket eder.</li> <li>• <b>JMS</b> Gönderirken, IBM MQ classes for JMS otomatik olarak bir MQRFH2 üstbilgisi oluşturur ve bunu IBM MQ iletilerinde içerir. When receiving, IBM MQ classes for JMS set the JMS message properties according to values in the MQRFH2 (if present); it does not present the MQRFH2 as part of the JMS message body.</li> <li>• <b>MQ</b> When sending, IBM MQ classes for JMS does not generate an MQRFH2. receivingalınırken, IBM MQ classes for JMS , JMS ileti gövdesinin bir parçası olarak MQRFH2 ' yi gösterir.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>WMQ_MESSAGE_BODY_BELIRTIEMEYEN</b></li> <li>• WMQ_MESSAGE_BODY_JMS</li> <li>• WMQ_MESSAGE_BODY_MQ</li> </ul>	setMessageBodyStyle

#### JMS kalıcı iletileri

IBM MQ classes for JMS uygulamaları, bazı güvenilirlik pahasına, JMS kalıcı iletileri için daha iyi performans sağlamak üzere **NonPersistentMessageClass** kuyruk özniteliğini kullanabilir.

Bir IBM MQ kuyruğu, **NonPersistentMessageClass**adlı bir özniteliğe sahiptir. Bu özniteliğin değeri, kuyruk yöneticisi yeniden başlatıldığında kuyruktaki kalıcı olmayan iletilerin atılıp atılmayacağını belirler.



You can set the attribute for a local queue by using the IBM MQ Script (MQSC) command, DEFINE QLOCAL, with either of the following parameters:

#### **NPMSİNİFİ (NORMAL)**

Kuyruk yöneticisi yeniden başlatıldığında, kuyruklardaki kalıcı olmayan iletiler atılır. Bu varsayılan değerdir.

#### **NPMCLASS (YÜKSEK)**

Kuyruk yöneticisi susturulmuş ya da hemen sona erdirildiğinde kuyruk yöneticisi yeniden başlatıldığında, kuyruklardaki kalıcı olmayan iletiler atılmaz. Ancak, önleyici olmayan iletiler atılabilir; ancak, önleyici bir sona erdirme ya da hata nedeniyle atılabilir.

Bu konuda, IBM MQ classes for JMS uygulamalarının JMS kalıcı iletileri için daha iyi performans sağlamak amacıyla bu kuyruk özneliğini nasıl kullanabilecekleri ele alınmıştır.

Bir Kuyruk ya da Konu nesnesinin PERSISTENCE özelliği HIGH değerine sahip olabilir. Bu değeri ayarlamak için IBM MQ JMS yönetim aracını kullanabilir ya da bir uygulama değıştirme olarak WMQConstants.WMQ\_PER\_NPHIGH değerini geçen Destination.setPersistence() yöntemini çağırabilir.

Bir uygulama, PERSISTENCE özelliğinin HIGH değeri (HIGH) değerine sahip olduğu bir hedefe JMS kalıcı iletisi ya da JMS kalıcı olmayan bir ileti gönderirse, temeldeki IBM MQ kuyruğu NPMCLASS (HIGH) olarak ayarlandıysa, ileti kuyruğa IBM MQ kalıcı olmayan bir ileti olarak konmuştur. If the PERSISTENCE property of the destination does not have the value HIGH, or if the underlying queue is set to NPMCLASS(NORMAL), a JMS persistent message is put on the queue as an IBM MQ persistent message, and a JMS nonpersistent message is put on the queue as an IBM MQ nonpersistent message.

JMS kalıcı iletisi, IBM MQ kalıcı olmayan bir ileti olarak bir kuyruğa yerleştirilirse ve bir kuyruk yöneticisinin susturulmuş ya da hemen sona erdirilmesi sonrasında iletinin atılmamasını sağlamak istiyorsanız, iletinin yönetileceği tüm kuyruklar NPMCLASS (HIGH) olarak ayarlanmalıdır. Yayınlama/abone olma etki alanında bu kuyruklar, abone kuyruklarını içerir. Bu yapılandırmayı zorlamak için bir yardım olarak IBM MQ classes for JMS bir InvalidDestinationKural Dışı Durumu atar; bir uygulama, PERSISTENCE özelliğinin HIGH (yüksek) değerine sahip olduğu ve temeldeki IBM MQ kuyruğunun npmclass (normal) olarak ayarlandığı bir hedef için ileti tüketicisi yaratmayı denerse.

Bir hedefin HIGH ' a PERSISTENCE özelliğinin ayarlanması, o hedeften alınan bir iletinin nasıl alınmadığını etkilemez. A message sent as a JMS persistent message is received as a JMS persistent message, and a message sent as a JMS nonpersistent message is received as a JMS nonpersistent message.

Bir uygulama, PERSISTENCE özelliğinin HIGH değerine sahip olduğu bir hedefe ilk iletiyi gönderdiğinde ya da bir uygulama, PERSISTENCE özelliğinin HIGH VALUE değerine sahip olduğu bir hedef için ilk ileti tüketicisi yarattığında, IBM MQ classes for JMS , temeldeki IBM MQ kuyruğunda NPMCLASS (YÜKSEK) olup olmadığını belirlemek için bir MQINQ çağrısı yayınlar. Bu nedenle, uygulamanın kuyruğa sorgulama yetkisi olmalıdır. Buna ek olarak, IBM MQ classes for JMS , hedef silinceye kadar MQINQ çağrısının sonucunu korur ve daha fazla MQINQ çağrıları yayınlamaz. Bu nedenle, uygulama yine de hedef kullanılırken, temeldeki kuyruğun NPMCLASS ayarını değıştirirseniz, IBM MQ classes for JMS yeni ayarı fark etmez.

JMS kalıcı iletilerinin IBM MQ kuyruklarına IBM MQ kalıcı olmayan iletiler olarak yerleştirilmesine izin vererek, bazı güvenilirlik giderlerinde performans kazanıyorsunuz. JMS kalıcı iletileri için maksimum güvenilirlik gereksiniminiz varsa, iletileri PERSISTENCE özelliğinin HIGH değerine sahip olduğu bir hedefe göndermeyin.

JMS Katmanı, SYSTEM.DEFAULT.MODEL.QUEUEyerine SYSTEM.JMS.TEMPQ.MODEL' yi kullanabilir. SYSTEM.JMS.TEMPQ.MODEL , kalıcı iletileri kabul edemeyen kalıcı dinamik kuyruklar oluşturur; SYSTEM.DEFAULT.MODEL.QUEUE kalıcı iletileri kabul edemez. Kalıcı iletileri kabul etmek üzere geçici kuyruklar kullanmak için, bu nedenle SYSTEM.JMS.TEMPQ.MODEL(MODEL) ya da model kuyruğunu, seçtiğiniz alternatif bir kuyruğa çevirin.

#### *Using TLS with IBM MQ classes for JMS*

IBM MQ classes for JMS uygulamaları, Transport Layer Security (TLS) şifrelemesini kullanabilir. Bunu yapmak için bir JSSE sağlayıcısına gereksinim duyarlar.

IBM MQ classes for JMS connections using TRANSPORT(CLIENT) support TLS encryption. TLS, iletişim şifreleme, kimlik doğrulama ve ileti bütünlüğü sağlar. Bu, genellikle İnternet üzerinde ya da bir intranet içinde herhangi iki eş arasında iletişim sağlamak için kullanılır.

IBM MQ classes for JMS , TLS şifrelemesini işlemek için Java Secure Socket Extension (JSSE) olanağını kullanır ve bu nedenle bir JSSE sağlayıcısını gerektirir. JSE v1.4 JVM ' ler yerleşik bir JSSE sağlayıcısına sahiptir. Sertifikaların nasıl yönetileceği ve saklanabileceği ile ilgili ayrıntılar sağlayıcıdan sağlayıcıya göre değişebilir. Bu konuda bilgi almak için JSSE sağlayıcısının belgelerine bakın.

Bu bölümde, JSSE sağlayıcınızın doğru bir şekilde kurulmuş ve yapılandırılmış olduğu ve JSSE sağlayıcınız için uygun sertifikaların kurulmuş ve kullanılabilir durumda olduğu varsayılır. Artık bir dizi 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, [“Using a client channel definition table with IBM MQ classes for JMS” sayfa 262](#) başlıklı konuya bakın.

### *SSLCIPHERSUIT nesne özelliği*

Bir ConnectionFactory nesnesi üzerinde TLS şifrelemesini etkinleştirmek için SSLCIPHERSUIT ögesini ayarlayın.

TLS şifrelemesini bir ConnectionFactory nesnesi üzerinde etkinleştirmek için, SSLCIPHERSUIT özelliğini JSSE sağlayıcınız tarafından desteklenen bir CipherSuite olarak ayarlamak için JMSAdmin 'i kullanın. Bu, hedef kanaldaki CipherSpec ayarına uygun olmalıdır. Ancak, CipherSuites , CipherSpecs ' ten farklıdır ve bu nedenle farklı adlara sahiptir. [“TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS” sayfa 229](#) contains a table mapping the CipherSpecs supported by IBM MQ to their equivalent CipherSuites as known to JSSE. IBM MQ ile CipherSpecs ve CipherSuites hakkında daha fazla bilgi için bkz. [Securing IBM MQ](#).

For example, to set up a ConnectionFactory object that can be used to create a connection over an TLS enabled MQI channel with a CipherSpec of TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA, issue the following command to JMSAdmin:

```
ALTER CF(my.cf) SSLCIPHERSUITE(SSL_RSA_WITH_AES_128_CBC_SHA)
```

This can also be set from an application, using the setSSLCipherSuite() method on an MQConnectionFactory object.

Kolaylık sağlamak amacıyla, SSLCIPHERSUITE özelliğinde bir CipherSpec belirtilirse, JMSAdmin CipherSpec ' yi uygun bir CipherSuite ile eşleştirmeyi dener ve bir uyarı yayınlar. Özellik bir uygulama tarafından belirtilmişse, bu eşleme girişimi yapılmaz.

CCDT (Client Channel Definition Table; İstemci Kanal Tanımlama Çizelgesi) olanağını kullanın. Daha fazla bilgi için [“Using a client channel definition table with IBM MQ classes for JMS” sayfa 262](#) başlıklı konuya bakın.

### *SSLFIPSREQUIRD nesne özelliği*

IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) tarafından desteklenen bir CipherSuite kullanmak için bağlantı gerekiyorsa, bağlantı üreticisinin SSLFIPSREQUIRE özelliğini YES değerine ayarlayın.

Bu özelliğin varsayılan değeri NO ise, bir bağlantı IBM MQ tarafından desteklenen herhangi bir CipherSuite ' i kullanabileceği anlamına gelir.

Bir uygulama birden çok bağlantı kullanıyorsa, uygulama ilk bağlantıyı yarattığında kullanılan SSLFIPSIN değeri, uygulama birbirini izleyen herhangi bir bağlantı yarattığında kullanılan değeri belirler. Bu, sonraki bir bağlantı yaratmak için kullanılan bağlantı üreticisinin SSLFIPSREQUIRD özelliğinin değerinin dikkate alınmamasını gösterir. Farklı bir SSLFIPSREQUIRET değeri kullanmak istiyorsanız uygulamayı yeniden başlatmalısınız.

Bir uygulama, ConnectionFactory nesnesinin setSSLFipsRequired () yöntemini çağırarak bu özelliği ayarlayabilir. CipherSuite ayarı belirlenmezse, özellik yoksayılr.

## İlgili görevler

MQI istemcisinde çalıştırma sırasında yalnızca FIPS onaylı CipherSpecs ' in kullanıldığını belirtme

## İlgili başvurular

Federal Information Processing Standards (FIPS) for UNIX, Linux, and Windows

### *SSLPEERNAME nesne özelliği*

Use SSLPEERNAME to specify a distinguished name pattern, to ensure that your JMS application connects to the correct queue manager.

Bir JMS uygulaması, ayırt edici ad (DN) kalıbı belirterek doğru kuyruk yöneticisine bağlanmasını sağlar. Bağlantı yalnızca, kuyruk yöneticisi örüntüyle eşleşen bir DN (DN) sunarsa başarılı olur. Bu örüntüm biçimiyle ilgili ayrıntılar için ilgili konulara bakın.

DN, bir ConnectionFactory nesnesine ilişkin SSLPEERNAME özelliği kullanılarak ayarlanır. Örneğin, aşağıdaki JMSAdmin komutu bir ConnectionFactory nesnesini, kuyruk yöneticisinin kendisini QMGR . karakterleriyle başlayan bir Ortak Ad ile ve en az iki Kuruluş Birimi adıyla tanıtmayı beklemesini sağlar; bunun ilki IBM ve ikinci WEBSPPHERE olması gerekir:

```
ALTER CF(my.cf) SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPPHERE)
```

Denetleme, büyük/küçük harfe duyarlı değildir ve virgül yerine noktalı virgüller yerleştirilebilir. SSLPEERNAME, bir MQConnectionFactory nesnesindeki setSSLPeerName () yöntemini kullanarak bir uygulamadan da ayarlanabilir. Bu özellik belirlenmezse, kuyruk yöneticisi tarafından sağlanan Ayırt Edici Ad üzerinde herhangi bir denetleme gerçekleştirilmez. CipherSuite ayarı belirlenmezse bu özellik yok sayılır.

### *SSLCERTSTORS nesne özelliği*

Sertifika iptal listesi (CRL) denetimi için kullanılacak LDAP sunucularının bir listesini belirlemek için SSLCertstors ögesini kullanın.

Artık güvenilir olmayan sertifikaları tanımlamak için sertifika iptal listesi (CRL) kullanılması yaygındır. CRL ' ler genellikle LDAP sunucularında barındırılır. JMS allows an LDAP server to be specified for CRL checking under Java 2 v1.4 or later. Aşağıdaki JMSAdmin örneği, JMS 'un crl1.ibm.com:adlı bir LDAP sunucusunda barındırılan bir CRL ' yi kullanmasını sağlar.

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com)
```

**Not:** Bir LDAP sunucusunda barındırılan bir CRL ile başarılı bir şekilde CertStore kullanmak için, Java Software Development Kit (SDK) ürününüzün CRL ile uyumlu olduğundan emin olun. Bazı SDK 'lar, CRL ' nin, LDAP v2 için bir şema tanımlayan RFC 2587 'ye uymasını gerektirir. Çoğu LDAP v3 sunucusu, bunun yerine RFC 2256 'yı kullanır.

LDAP sunucunuz varsayılan 389 kapısında çalışmıyorsa, kapıyı iki nokta üst üste (: ) ve kapı numarasını anasistem adına ekleyerek belirleyebilirsiniz. Kuyruk yöneticisi tarafından sunulan sertifika crl1.ibm.com'da bulunan CRL ' de varsa, bağlantı tamamlanmamaktadır. Tek hata noktasını önlemek için, JMS , birden çok LDAP sunucusunun, boşluk karakteriyle sınırlanmış LDAP sunucularının bir listesini sağlayarak sağlanmasını sağlar. Örnek:

```
ALTER CF(my.cf) SSLCRL(ldap://crl1.ibm.com ldap://crl2.ibm.com)
```

Birden çok LDAP sunucusu belirtildiğinde, JMS , kuyruk yöneticisinin sertifikasını başarıyla doğrulayabilecek bir sunucu buluncaya kadar her birini sırayla dener. Her bir sunucunun aynı bilgileri içermesi gerekir.

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 oluşturabilir, bunları uygun bir Toplama nesnesine yerleştirebilir ve bu Veri Toplama nesnesini setSSLCertStores () yöntemine sağlayabilir. Bu şekilde, uygulama CRL denetimini özelleştirebilir. CertStore nesnelerini oluşturmaya ve kullanmaya ilişkin ayrıntılar için JSSE belgelerinize bakın.

Bir bağlantı ayarlandığında, kuyruk yöneticisi tarafından sunulan sertifika aşağıdaki gibi doğrulanır:

1. Kaynak grubundaki ilk CertStore nesnesi, bir CRL sunucusunu tanımlamak için kullanılır. sslCertdepolar tarafından tanımlanır.
2. CRL sunucusuyla bağlantı kurma girişiminde bulunmanız gerekir.
3. Girişim başarılı olursa, sunucu, sertifikana için bir eşleşme için arama yapılır.
  - a. Sertifikana geri alınacak bir sertifika bulunursa, arama işlemi sona erer ve bağlantı isteği, MQRC\_SSL\_CERTIFICATE\_FESHRIVE neden kodlarıyla başarısız olur.
  - b. Sertifika bulunamazsa, arama işlemi sona ermiş ve bağlantının devam etmesine izin verilir.
4. Sunucuyla iletişim kurma girişimi başarısız olursa, bir sonraki CertStore nesnesi bir CRL sunucusunu tanımlamak için kullanılır ve süreç 2. adımdan yinelenir.

Bu, derlemdeki son CertStore ise ya da Kaynak Grubu hiçbir CertStore nesnesi içermiyorsa, arama işlemi başarısız olur ve bağlantı isteği MQRC\_SSL\_CERT\_STORE\_ERROR neden koduyla başarısız olur.

Veri Toplama nesnesi, CertStores ' un kullanıldığı sırayı belirler.

Uygulamanız bir CertStore nesnelere derlemi ayarlamak için setSSLCertStores () kullanıyorsa, MQConnectionFactory artık bir JNDI ad alanına bağlanmaz. Bunu gerçekleştirme girişimi bir kural dışı duruma neden olur. sslCertStores özelliği ayarlanmazsa, kuyruk yöneticisi tarafından sağlanan sertifikada herhangi bir iptal denetimi gerçekleştirilmez. CipherSuite ayarı belirlenmezse bu özellik yok sayılır.

#### *SSSRESETCOUNT nesne özelliği*

Bu özellik, şifreleme için kullanılan gizli anahtardan önce bir bağlantı tarafından gönderilen ve alınan toplam bayt sayısını temsil eder.

Gönderilen bayt sayısı, şifrelemeden önceki sayıdır ve alınan bayt sayısı, şifre çözme işleminden sonra gelen sayıdır. Bayt sayısı, IBM MQ classes for JMS tarafından gönderilen ve alınan denetim bilgilerini de içerir.

Örneğin, 4 MB ' lik veri akıldıktan sonra yeniden anlaşma sağlanan bir TLS etkin MQI kanalı üzerinden bir bağlantı yaratmak üzere kullanılacak bir ConnectionFactory nesnesini yapılandırmak için, JMSAdmin komutunu aşağıdaki komutu verin:

```
ALTER CF(my.c#) SSLRESETCOUNT(4194304)
```

Bir uygulama, bir ConnectionFactory nesnesinin setSSLResetCount () yöntemini çağırarak bu özelliği ayarlayabilir.

Bu özelliğin değeri sıfır ise, varsayılan değer olan gizli anahtar hiçbir zaman yeniden anlaşmamalıdır. CipherSuite ayarı belirlenmezse, özellik yok sayılır.

#### *SSLSocketFactory nesne özelliği*

Bir uygulamaya ilişkin TLS bağlantısının diğer yönlerini uyarlamak için bir SSLSocketFactory yaratın ve bunu kullanmak için JMS ' i yapılandırın.

Bir uygulama için TLS bağlantısının diğer yönlerini özelleştirmek isteyebilirsiniz. Örneğin, şifreleme donanımını başlatmak ya da anahtar deposunu ve güvenli deponun kullanımını değiştirmek isteyebilirsiniz. Bunu yapmak için, uygulamanın öncelikle uygun şekilde özelleştirilen bir javax.net.ssl.SSLSocketFactory nesnesi yaratması gerekir. Özelleştirilebilir özellikler sağlayıcıdan sağlayıcıya farklılık gösterdiğinden, bunu nasıl yapacağına ilişkin bilgi için JSSE belgelerimize bakın. Uygun bir SSLSocketFactory nesnesi elde edildikten sonra, uyarlanmış SSLSocketFactory nesnesini kullanmak üzere JMS ' u yapılandırmak için MQConnectionFactory.setSSLSocketFactory () yöntemini kullanın.

Uygulamanız, özelleştirilmiş bir SSLSocketFactory nesnesi ayarlamak için setSSLSocketFactory () yöntemini kullanıyorsa, MQConnectionFactory nesnesi artık bir JNDI ad alanına bağlanmaz. Bunu gerçekleştirme girişimi bir kural dışı duruma neden olur. Bu özellik belirlenmezse, varsayılan SSLSocketFactory nesnesi kullanılır. Varsayılan SSLSocketFactory nesnesinin davranışına ilişkin ayrıntılar için JSSE belgelerimize bakın. CipherSuite ayarı belirlenmezse bu özellik yok sayılır.

**Önemli:** Güvenli olmayan bir JNDI ad alanından bir ConnectionFactory nesnesi alındığında SSL özelliklerinin kullanılmasının güvenlik güvenceye alınmasını güvenceye aldığından emin olun. Özellikle, JNDI 'nin standart LDAP somutlaması güvenli değildir. Bir saldırgan LDAP sunucusunu taklit edebilir, bir JMS uygulamasını fark etmeden yanlış sunucuya bağlanmaya yönlendirebilir. Uygun güvenlik düzenlemeleriyle, diğer JNDI somutlamaları (fscontext somutlaması gibi) güvenlidir.

*JSSSE anahtar deposunda ya da güvenilir depoda değişiklik yapılması*

Anahtar deposunda ya da güvenilir depoda değişiklik yaparsanız, değişikliklerin alınabilmesine ilişkin bazı işlemleri yapmanız gerekir.

JSSSE anahtar deposu ya da güvenilirlik deposunun içeriğini değiştirirseniz ya da anahtar deposu ya da güvenilirlik deposu dosyasının konumunu değiştirdiğinizde, o sırada çalışan IBM MQ classes for JMS uygulamaları otomatik olarak değişiklikleri almaz. Değişikliklerin yürürlüğe girmesi için aşağıdaki işlemler gerçekleştirilmelidir:

- Uygulamalar tüm bağlantılarını kapatmalı ve bağlantı havuzlarındaki kullanılmayan bağlantıları yok etmelidir.
- JSSSE sağlayıcınız, bilgileri anahtar deposundan ve güvenilir depodan önbelleğe aldıysa, bu bilgiler yenilenmelidir.

Bu işlemler gerçekleştirildikten sonra, uygulamalar bağlantılarını yeniden yaratabilirler.

Uygulamalarınızı nasıl tasarladığınıza ve JSSSE sağlayıcınız tarafından sağlanan işlemlere bağlı olarak, uygulamalarınızı durdurup yeniden başlatmaksızın bu işlemleri gerçekleştirmeniz de mümkün olabilir. Ancak, uygulamaların durdurulması ve yeniden başlatılması en basit çözüm olabilir.

*TLS CipherSpecs and CipherSuites in IBM MQ classes for JMS*

IBM MQ classes for JMS uygulamalarının bir kuyruk yöneticisiyle bağlantı kurabilme yeteneği, MQI kanalının sunucu ucunda belirtilen CipherSpec değerine ve istemci ucunda belirtilen CipherSuite 'e bağlıdır.

Aşağıdaki çizelgede IBM MQ tarafından desteklenen CipherSpecs ve eşdeğeri olan CipherSuiteslistesi listelenmektedir.

Aşağıdaki çizelgede listelenen CipherSpecs'in IBM MQ tarafından kullanımdan kaldırılmış olup olmadığını ve bu durumda CipherSpec 'in kullanımdan kaldırılıp kaldırılmadığını görmek için [Kullanımdan Kaldırılan CipherSpecs](#) konusunu gözden geçirmelisiniz.

**Önemli:** Listelenen CipherSuites , IBM MQ ile sağlanan IBM Java Runtime Environment (JRE) tarafından desteklenen sürümlerdir. Listelenen CipherSuites , Oracle Java JRE tarafından desteklenenler içerir. Uygulamanızın bir Oracle Java JRE kullanacak şekilde yapılandırılmasına ilişkin ek bilgi edinmek için [Uygulamanızın IBM Java ya da Oracle Java CipherSuite eşlemelerini kullanacak şekilde yapılandırılması](#) konusuna bakın.

Tablo ayrıca, iletişim için kullanılan protokolü ve CipherSuite 'in FIPS 140-2 standardına uygun olup olmadığını da belirtir.

Uygulama, FIPS 140-2 uyumluluğunu zorunlu kılacak şekilde yapılandırılmamışsa, FIPS 140-2 uyumluluğu yapılandırılmamışsa, ancak uygulama için FIPS 140-2 uyumluluğu yapılandırıldıysa (yapılandırma için aşağıdaki notlara bakın), yalnızca FIPS 140-2 uyumlu olarak işaretlenmiş olan CipherSuites ' lar yapılandırılabilir; başka CipherSuites sonuçlarını bir hatayla kullanma girişiminde bulunmanız gerekir.

**Not:** Her JRE birden çok şifreleme güvenlik sağlayıcısına sahip olabilir; bunların her biri aynı CipherSuite ürününün uygulanmasına katkıda bulunabilir. Ancak, tüm güvenlik sağlayıcıları FIPS 140-2 sertifikalı değildir. If FIPS 140-2 compliance is not enforced for an application then it is possible that an uncertified implementation of the CipherSuite might be used. Onaylanmamış uygulamalar FIPS 140-2 ile uyumlu çalışmayabilir; CipherSuite teorik olarak, standart için gerekli olan en düşük güvenlik düzeyini karşılayabilir. FIPS 140-2 uygulamasını IBM MQ JMS uygulamalarında yapılandırılmasıyla ilgili daha fazla bilgi için aşağıdaki notlara bakın.

CipherSpecs ve CipherSuitesi için FIPS 140-2 ve Suite-B uyumluluğu ile ilgili daha fazla bilgi için bkz. [Specifying CipherSpecs](#). Ayrıca, [US Federal Information Processing Standards](#) (Federal Bilgi İşleme Standartları) ile ilgili bilgilerden haberdar olmanız da gerekebilir.

To use the full set of CipherSuites and to operate with certified FIPS 140-2 and/or Suite-B compliance, a suitable JRE is required. IBM Java 7 Service Refresh 4 Düzeltme Paketi 2 ya da daha yüksek bir IBM JRE düzeyi, [Çizelge 40 sayfa 231](#) içinde listelenen TLS 1.2 CipherSuites için uygun desteği sağlar.

**V 9.1.5** TLS v1.3 Ciphers olanağını kullanabilmek için uygulamanızın çalışan JRE 'nin TLS v1.3' ü desteklemesi gerekir.

**Not:** Bazı CipherSuites'i kullanmak için, JRE' de 'kısıtlanmamış' ilke dosyalarının yapılandırılması gerekir. İlke dosyalarının bir SDK ya da JRE ' de nasıl ayarlarıyla ilgili ayrıntılı bilgi için, kullandığınız sürüm için *Security Reference for IBM SDK, Java Technology Edition* içindeki *IBM SDK Policy files* başlıklı konuya bakın.

Çizelge 40. IBM MQ ve eşdeğer CIPHER Suite tarafından desteklenen Cipher Specs

CipherSpec	Eşdeğer Cipher Suite (IBM JRE)	Eşdeğer Cipher Suite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CIPHERSPEC	Eşdeğer CIPHERSUITE (IBM JRE)	Eşdeğer CIPHERSUITE (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 40. IBM MQ ve eşdeğer CIPHER Suite tarafından desteklenen Cipher Specs (devamı var)

CipherSpec	Eşdeğer Cipher Suite (IBM JRE)	Eşdeğer Cipher Suite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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

Çizelge 40. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLSECDHE_ECDSA_WITH_NULL_SHA	TLS 1.2	hayır

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CIPHERSPEC	Eşdeğer CIPHERSUITE (IBM JRE)	Eşdeğer CIPHERSUITE (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS_ECDHE_ECDSA_WITH_RC4_128_SHA	TLS 1.2	hayır

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_RSA_3DES_EDE_CBC_SHA 256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	evet

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	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 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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

Çizelge 40. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLS 1.2	hayır

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLSECDHE_RSA_WITH_RC4_128_SHA	TLS 1.2	hayır

Çizelge 40. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
TLS_RSA_WITH_3DES_EDE_CBC_SHA <a href="#">"1" sayfa 257</a>	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TL S_ R S A - W I T H _ 3 D E S _ E D E _ C B C _ S H A	TLS 1.0	evet

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TL S_ R S A - W I T H - A E S _1 2 8_ C B C_ S H A	TLS 1.0	evet

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TL S_ R S A - W I T H - A E S _1 2 8_ C B C_ S H A 2 5 6	TLS 1.2	evet

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_128_GCM_SHA256	SSL_RSA_WITH_AES_128_GCM_SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	evet



Çizelge 40. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0	evet

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CIPHERSPEC	Eşdeğer CIPHERSUITE (IBM JRE)	Eşdeğer CIPHERSUITE (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_256_CBC_SHA256	SSL_RSA_WITH_AES_256_CBC_SHA256	TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	evet

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TL S_ R S A - W I T H - A E S _2 5 6_ G C M _S H A 3 8 4	TLS 1.2	evet

Çizelge 40. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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

Çizelge 40. IBM MQ ve eşdeğer CipherSuites tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA	TLS 1.2	hayır
ANY_TLS12	*TLS12	*TLS12	TLS 1.2	evet

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CIPHERSPEC	Eşdeğer CIPHERSUITE (IBM JRE)	Eşdeğer CIPHERSUITE (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
<b>V9.1.5</b> TLS_AES_128_GCM_SHA256 <a href="#">"2" sayfa 257</a>	TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLS V1.3	hayır
<b>V9.1.5</b> TLS_AES_256_GCM_SHA384 <a href="#">"2" sayfa 257</a>	TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLS V1.3	hayır

Çizelge 40. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
<p>► <b>V9.1.5</b></p> <p>TLS_CHACHA20_POLY1305_SHA256 6 "2" sayfa 257</p>	TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLS V1.3	hayır

Çizelge 40. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
<p><b>V9.1.5</b>                      TLS_AES_128_CCM_SHA256                      "2" sayfa 257</p>	TLS_AES_128_CCM_SHA256	TLS_AES_128_CCM_SHA256	TLS V1.3	hayır
<p><b>V9.1.5</b>                      TLS_AES_128_CCM_8_SHA256                      "2" sayfa 257</p>	TLS_AES_128_CCM_8_SHA256	TLS_AES_128_CCM_8_SHA256	TLS V1.3	hayır
<p><b>V9.1.5</b> ANY "2" sayfa 257</p>	*ANA	*ANA	Çoklu	hayır



Çizelge 40. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
> V9.1.5 ANY_TLS13 "2" sayfa 257	*TLS13	*TLS13	TLS V13	hayır
> V9.1.5 ANY_TLS12_OR_HIGHER "2" sayfa 257	*TLS12ORHIGHER	*TLS12ORHIGHER	TLS V1.2 ve üstü	hayır
> V9.1.5 ANY_TLS13_OR_HIGHER "2" sayfa 257	*TLS13ORHIGHER	*TLS13ORHIGHER	TLS V1.3 ve üstü	hayır

**Notlar:**

1. CipherSpec TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA kullanımdan kaldırıldı. However, it can still be used to transfer up to 32 GB of data before the connection is terminated with error AMQ9288. Bu hatayı önlemek için, üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpec komutunu kullanırken gizli anahtar sıfırlamayı etkinleştirmeniz gerekir.
2. > V9.1.5 TLS v1.3 Ciphers olanağını kullanabilmek için uygulamanızın çalışan JRE 'nin TLS v1.3' ü desteklemesi gerekir.

## IBM MQ classes for JMS uygulamasında Ciphersuites ve FIPS uyumluluğunu yapılandırma

- IBM MQ classes for JMS 'u kullanan bir uygulama, bir bağlantı için CipherSuite ' i ayarlamak için iki yöntemden birini kullanabilir:
  - ConnectionFactory nesnesine ilişkin setSSLCipherSuite yöntemini çağırın.
  - Bir ConnectionFactory nesnesine ilişkin SSLCIPHERSUIT özelliğini ayarlamak için IBM MQ JMS yönetim aracını kullanın.
- IBM MQ classes for JMS ' u kullanan bir uygulama, FIPS 140-2 uyumluluğunu zorunlu kılacak iki yöntemden birini kullanabilir:
  - Call the setSSLFipsRequired method of a ConnectionFactory object.
  - Bir ConnectionFactory nesnesinin SSLFIPSREQUIRET özelliğini ayarlamak için IBM MQ JMS yönetim aracını kullanın.

## Uygulamanızın IBM Java ya da Oracle Java CipherSuite eşlemelerini kullanacak şekilde yapılandırılması

Uygulamanızın varsayılan IBM Java CipherSuite ögesini IBM MQ CipherSpec eşlemelerine ya da Oracle CipherSuite ' i IBM MQ CipherSpec eşlemelerine kullanıp kullanmayacağını yapılandırabilirsiniz. Bu nedenle, uygulamanızın bir IBM JRE ya da bir Oracle JRE kullanıyorsa TLS CipherSuites ' i kullanabilirsiniz. The Java System Property `com.ibm.mq.cfg.useIBMCipherMappings` controls which mappings are used. Özellik, aşağıdaki değerlerden biri olabilir:

### doğru

IBM Java CipherSuite ' i IBM MQ CipherSpec eşlemelerine kullanın.

Bu değer, varsayılan değerdir.

### yanlış

Oracle CipherSuite ' i IBM MQ CipherSpec eşlemelerine kullanın.

For more information about using IBM MQ Java and TLS Ciphers, see the MQdev blog post [MQ Java, TLS Ciphers, Non-IBM JREs & APAR 'lar IT06775, IV66840, IT09423, IT10837.](#)

## Birlikte çalışabilirlik

Belirli CipherSuites , kullanılan protokole bağlı olarak birden çok IBM MQ CipherSpec ile uyumlu olabilir. Ancak , yalnızca Tablo 1 'de belirtilen TLS sürümünü kullanan CipherSuite/CipherSpec bileşimi desteklenir. Desteklenmeyen CipherSuites ve CipherSpecs birleşimlerini kullanmaya çalışmak uygun bir kural dışı durum ile başarısız olur. Bu CipherSuite/CipherSpec birleşimlerinden herhangi birini kullanan kuruluşlar, desteklenen bir birleşmeye geçmelidir.

Aşağıdaki tabloda, bu sınırlamanın geçerli olduğu CipherSuites gösterilmektedir.

CipherSuite	Desteklenen TLS CipherSpec	Desteklenmeyen SSL CipherSpec
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA A "1" sayfa 258	TRIPLE_DES_SHA_US
SSL_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA	DESTE_SHA_EXPORT
SSL_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_RC4_128_SHA256	RC4_SHA_US

### Not:

1. Bu CipherSpec TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA kullanımdan kaldırılmıştır. However, it can still be used to transfer up to 32 GB of data before the connection is terminated with error AMQ9288. Bu

hatayı önlemek için, üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckomutunu kullanırken gizli anahtar sıfırlamayı etkinleştirmeniz gerekir.

### Writing channel exits in Java for IBM MQ classes for JMS

Belirtilen arabirimleri gerçekleştiren Java sınıflarını tanımlayarak kanal çıkışları oluşturursunuz.

com.ibm.mq.exits paketinde üç arabirim tanımlanır:

- Bir gönderme çıkışı için WMQSendExit
- Bir alma çıkışı için WMQReceiveExit
- WMQSecurityExit, for a security exit

Aşağıdaki örnek kod, üç arabirimi gerçekleştiren bir sınıfı tanımlıyor:

```
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 bir çıkış bir MQCXP nesnesi ve bir MQCD nesnesi değiştirgeleri olarak alır. Bu nesnelere, yordamsal arabirimde tanımlanan 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 gerekli değildir. Çıkış gönderme işlevi, değeri sunucu kuyruk yöneticisine gönderilecek olan değeri olarak döndürülür. Ancak, gönderme çıkışı bir gönderme çıkışındaki son çıkış değilse, döndürülen veriler sırayla bir sonraki gönderme çıkışına aktarılır. Bir gönderme çıkışı, agentBuffer değiştirgesinde aldığı verilerin değiştirilmiş bir sürümünü döndürebilir ya da verileri değiştirmeden geri döndürebilir. Bu nedenle, olabilecek en basit çıkış gövdesi:

```
{ return agentBuffer; }
```

Bir alma çıkışı çağrıldığında, agentBuffer parametresi, sunucu kuyruk yöneticisinden alınan verileri içerir. Alma çıkışı, IBM MQ classes for JMS tarafından uygulamaya geçirilecek verilerin değeri olarak geri döner. Ancak, alma çıkışı bir alma işlemi sırasında son alma çıkışı değilse, döndürülen veriler sırayla bir sonraki alma çıkışına geçirilir.

Bir güvenlik çıkışı çağrıldığında, agentBuffer parametresi, bağlantının sunucu ucundaki güvenlik çıkışından bir güvenlik akımında alınan verileri içerir. Güvenlik çıkışı, bir güvenlik akışında sunucu güvenlik çıkışına gönderileceği veri değeri olarak döndürülür.

Kanal çıkışları, arka diziyeye sahip bir arabelleğe çağrılır. En iyi başarıyı elde etmek için, çıkışta yedek diziyeye sahip bir arabellek döndürülmelidir.

Çağrıldığında, en çok 32 karakterlik kullanıcı verileri kanal çıkışa 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 verilerde yapılan değişiklikler kaybedilir. Ancak, çıkış, MQCXP nesnesinin çıkış kullanıcı alanını kullanarak bir çağrıdan diğerine veri geçirebilir. The exit accesses the exit user area by reference by calling the `getExitUserArea()` method.

Her çıkış sınıfının bir oluşturucusu olmalıdır. Oluşturucu, önceki örnekte gösterilen varsayılan oluşturucu olabilir ya da dizgi değiştirgesi olan bir oluşturucu olabilir. Oluşturucu, sınıftaki her çıkışa ilişkin çıkış sınıfının bir eşgörünümünü yaratmak için çağrılır. Bu nedenle, önceki örnekte, gönderme çıkışı için `MyMQExits` sınıfının bir örneği yaratılır; alma çıkışı için başka bir yönetim ortamı yaratılır ve güvenlik çıkışı için üçüncü bir yönetim ortamı yaratılır. Bir dizgi değiştirgesi içeren bir oluşturucu çağrıldığında, değiştirge, yönetim ortamının yaratılmakta olduğu kanal çıkışa geçirilen kullanıcı verilerini içerir. Bir çıkış sınıfının hem varsayılan oluşturucusu hem de tek bir değiştirge oluşturucusu varsa, tek parametre oluşturucusu öncelik kazanır.

Bağlantıyı kanal çıkışı içinden kapatmayın.

Veriler bir bağlantının sunucu ucuna gönderildiğinde TLS şifrelemesi, kanal çıkışlarından *sonra* çağrılır. Benzer şekilde, bir bağlantının sunucu ucundan veri alındığında TLS şifre çözme işlemi, kanal çıkışlarının çağrılmadan *önce* gerçekleştirilmesini sağlar.

7.0 sürümünden önceki IBM MQ classes for JMS sürümlerinde kanal çıkışları, `MQSendExit`, `MQReceiveExit` ve `MQSecurityExit` arabirimlerini kullanılarak uygulanmış. Bu arabirimleri kullanmaya devam edebilirsiniz, ancak yeni arabirimler geliştirilmiş işlev ve başarımlar için tercih edilir.

#### *Configuring IBM MQ classes for JMS to use channel exits*

Bir IBM MQ classes for JMS uygulaması, uygulama bir kuyruk yöneticisine bağlandığında başlatılacak olan MQI kanalından kanal güvenliğini, gönderme ve alma çıkışlarını kullanabilir. Uygulama, Java, C ya da C++ içinde yazılan çıkışları kullanabilir. Uygulama, art arda çalıştırılan gönderme ya da alma çıkışlarını da kullanabilir.

Aşağıdaki özellikler, bir gönderme çıkışı ya da bir JMS bağlantısı tarafından kullanılan bir gönderme çıkışı sırası belirtmektedir:

- Bir `MQConnectionFactory` nesnesine ilişkin **SENDEXIT** özelliği.
- Gelen iletişim için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan etkinleştirme belirtimindeki **sendexit** özelliği,
- Çıkış iletişimi için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir `ConnectionFactory` nesnesindeki **sendexit** özelliği.

Özelliğin değeri, virgüllerle ayrılmış bir ya da daha çok öğeyi oluşturan bir dizedir. Her bir öğe, bir gönderme çıkışını aşağıdaki yollardan biriyle tanımlar:

- The name of a class that implements the `WMQSendExit` interface for a send exit written in Java.
- C ya da C++ içinde yazılmış bir gönderme çıkışı için *libraryName (entryPointAd)* biçiminde bir dizgi.

Benzer bir şekilde, aşağıdaki özellikler bağlantı tarafından kullanılan alma çıkışını ya da alma çıkışlarını belirtir:

- Bir `MQConnectionFactory` nesnesine ilişkin **RECEXIT** özelliği.
- Gelen iletişim için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan etkinleştirme belirtimindeki **receiveexit** özelliği,
- Çıkış iletişimi için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir `ConnectionFactory` nesnesindeki **receiveexit** özelliği.

Aşağıdaki özellikler, bağlantı tarafından kullanılan güvenlik çıkışını belirtir:

- Bir `MQConnectionFactory` nesnesine ilişkin **SECEXIT** özelliği.

- Gelen iletişim için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan etkinleştirme belirtimindeki **securityexit** özelliği,
- Çıkış iletişimi için IBM MQ kaynak bağdaştırıcısı tarafından kullanılan bir ConnectionFactory nesnesindeki **securityexit** özelliği.

MQConnectionFactory için, IBM MQ JMS yönetim aracını ya da IBM MQ Explorer olanağını kullanarak **SENDEXIT**, **RECEXIT** ve **SECEXIT** özelliklerini ayarlayabilirsiniz. Alternatively, an application can set the properties by calling the `setSendExit()`, `setReceiveExit()`, and `setSecurityExit()` methods.

Kanal çıkışları, kendi sınıf yükleyicilerine göre yüklenir. Bir kanal çıkışı bulmak için sınıf yükleyici belirtilen siparişte aşağıdaki konumları arar.

1. **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** özelliği tarafından belirtilen sınıf yolu ya da IBM MQ istemcisi yapılandırma dosyasının Channels kısmında **JavaExitsClassPath** özneliği tarafından.
2. Java sistem özelliği **com.ibm.mq.exitClasspath** tarafından belirtilen sınıf yolu. Bu özelliğin artık kullanımdan kaldırıldığını unutmayın.
3. The IBM MQ exits directory, as shown in [Çizelge 42 sayfa 261](#). Sınıf yükleyici ilk olarak, Java arşiv (JAR) dosyalarında paketlenmeyen sınıf dosyaları için dizini arar. Kanal çıkışı bulunamazsa, sınıf yükleyici dizinde JAR dosyalarıyla arama yapar.

Çizelge 42. IBM MQ çıkış dizini	
Platform	Dizin
Linux UNIX Linux	/var/mqm/exits (32 bit kanal çıkışları) /var/mqm/exits64 (64 bit kanal çıkışları)
Windows Windows	kurulus_data_dzn\çıkışlar Burada <i>kurulus_veri_dzn</i> , kuruluş sırasında IBM MQ veri kütükleri için seçtiğiniz dizindir. Varsayılan dizin C:\ProgramData\IBM\MQdizidir.

**Not:** Bir kanal çıkışı birden çok yerde varsa, IBM MQ classes for JMS bulunduğu ilk eşgörünümü yükler.

Sınıf yükleyicisinin üst ögesi, IBM MQ classes for JMS' yi yüklemek için kullanılan sınıf yükleyicidir. Bu nedenle, üst sınıf yükleyicisinin önceki konumların hiçbirinde bulunamaması durumunda kanal çıkışı yüklemesi mümkündür. However, when you are using the IBM MQ classes for JMS in an environment such as a JEE application server, you are not likely to be able to influence the choice of the parent class loader and so the class loader should be configured by setting the Java system property **com.ibm.mq.cfg.ClientExitPath.JavaExitsClasspath** on the application server.

Uygulamanız Java security manager etkinleştirilmiş olarak çalıştırılıyorsa, uygulamanın çalışmakta olduğu Java yürütme ortamı tarafından kullanılan ilke yapılandırma dosyası, kanal çıkış sınıfını yüklemek için gereken izinlere sahip olmalıdır. Bunun nasıl yapacağına ilişkin bilgi için [Java Security Manager altındaki JMS uygulamaları için IBM MQ sınıflarının çalıştırılması](#) başlıklı konuya bakın.

The MQSendExit, MQReceiveExit, and MQSecurityExit interfaces supplied with versions of IBM WebSphere MQ earlier than IBM WebSphere MQ 7.0 are still supported. If you use channel exits that implement these interfaces, `com.ibm.mq.jar` must be present in the class path.

Kanal çıkışlarının C ' de nasıl yazılacağı ile ilgili bilgi için bkz. "[İleti alışverişi kanallarına ilişkin kanal çıkışı programları](#)" sayfa 993. Kanal çıkış programlarını, [Çizelge 42 sayfa 261](#) içinde gösterilen dizinde C ya da C++ dilinde yazılmış olarak saklamanız gerekir.

Uygulamanız bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, "[Using a client channel definition table with IBM MQ classes for JMS](#)" sayfa 262 başlıklı konuya bakın.

*IBM MQ classes for JMS kullanılırken kanal çıkışlarına geçirilecek kullanıcı verilerinin belirtilmesi* Çağrıldığında, en çok 32 karakterlik kullanıcı verileri kanal çıkışa geçirilebilir.

Bir MQConnectionFactory nesnesinin SENDEXITINIT özelliđi, çağrıldıđında her bir gönderme çıkışa geçirilecek kullanıcı verilerini belirtir. Özelliđin deđeri, kullanıcı verilerinin virgülle ayrılmış bir ya da daha fazla öđesini içeren bir dizgidir. Dizgi içindeki kullanıcı verilerinin her bir öđesinin konumu, hangi gönderme çıkıştan çıkıldığında, çıkış gönderilerinde kullanıcı verilerinin aktarılacağı saptar. Örneđin, dizgideki kullanıcı verilerinin ilk öđesi, bir gönderme çıkışları sırasındaki ilk gönderme çıkışa geçirilir.

You can set the SENDEXITINIT property by using the IBM MQ JMS administration tool or IBM MQ Explorer. Diđer bir seçenek olarak, bir uygulama özelliđi setSendExitInit() yöntemini çağırarak ayarlayabilir.

Benzer bir şekilde, bir ConnectionFactory nesnesinin RESEXITINIT özelliđi, her alma çıkışa geçirilen kullanıcı verilerini belirtir ve SESEXITINIT özelliđi, güvenlik çıkışa geçirilen kullanıcı verilerini belirtir. Bu özellikleri, IBM MQ JMS yönetim aracını ya da IBM MQ Explorer' i kullanarak ayarlayabilirsiniz. Alternatively, an application can set the properties by calling the setReceiveExitInit() and setSecurityExitInit() methods.

Kanal çıkışlarına geçirilen kullanıcı verilerini belirlerken aşağıdaki kurallara dikkat edin:

- Bir dizgideki kullanıcı verilerinin sayısı, bir sıradaki çıkışların sayısından fazlaysa, kullanıcı verilerinin fazla öđeleri yoksayılır.
- Bir dizgideki kullanıcı verilerinin sayısı, bir sıradaki çıkış sayısından azsa, kullanıcı verilerinin her belirlenmemiş öđesi boş bir dizgi olarak ayarlanır. Bir dizgi içinde art arda iki virgöl ya da bir dizginin başlangıcındaki bir virgöl, kullanıcı verilerinin belirlenmemiş bir öđesini de gösterir.

Bir uygulama, bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, istemci bağlantı kanalı tanımlamasında belirtilen tüm kullanıcı verileri, çağrıldıđında kanal çıkışlarına geçirilir. İstemci kanal tanımlama çizelgesini kullanma hakkında daha fazla bilgi için bkz. [“Using a client channel definition table with IBM MQ classes for JMS” sayfa 262.](#)

#### *Using a client channel definition table with IBM MQ classes for JMS*

Bir IBM MQ classes for JMS uygulaması, istemci kanal tanımlama çizelgesinde (CCDT) saklanan istemci bağlantısı kanal tanımlarını kullanabilir. CCDT ' i kullanmak için bir ConnectionFactory nesnesi yapılandırırıyorsunuz. Kullanımıyla ilgili bazı kısıtlamalar vardır.

Bir ConnectionFactory nesnesinin belirli özelliklerini ayarlayarak bir istemci bağlantı kanalı tanımlamak için bir alternatif olarak, bir IBM MQ classes for JMS uygulaması, istemci kanal tanımlama çizelgesinde saklanan istemci bağlantısı kanal tanımlamalarını kullanabilir. Bu tanımlar, IBM MQ Script (MQSC) komutları ya da IBM MQ Programmable Command Format (PCF) komutları tarafından yaratılır. Uygulama bir Connection nesnesi yarattığında, IBM MQ classes for JMS , uygun bir istemci bağlantısı kanal tanımlaması için istemci kanal tanımlama çizelgesini arar ve bir MQI kanalı başlatmak için kanal tanımlamasını kullanır. İstemci kanal tanımlama tabloları ve nasıl oluşturulacağı hakkında daha fazla bilgi için bkz. [İstemci kanal tanımlama tablosu.](#)

Bir istemci kanal tanımlama çizelgesi kullanmak için, bir ConnectionFactory nesnesinin CCDTURL özelliđinin bir URL nesnesi olarak ayarlanması gerekir. IBM MQ classes for JMS do not read the information about the CCDT from the IBM MQ MQI client configuration file, although some other values are used from there (see [“IBM MQ classes for JMS yapılandırma dosyası” sayfa 88](#) for which value apply). URL nesnesi, istemci kanal tanımlama çizelgesini içeren dosyanın adını ve yerini tanımlayan ve dosyanın nasıl erişilebileceđini belirten bir URL adresi (uniform resource locator; URL) yerleřtirir. CCDTURL özelliđini IBM MQ JMS yönetim aracını kullanarak ayarlayabilir ya da bir uygulama, bir URL nesnesi yaratarak ve ConnectionFactory nesnesinin setCCDTURL() yöntemini çağırarak özelliđi ayarlayabilirler.

Örneđin, ccdt1.tab dosyası bir istemci kanal tanımlama çizelgesi içeriyorsa ve uygulamanın çalıştığı sistemde saklandıysa, uygulama CCDTURL özelliđini 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 içerdiđini ve uygulamanın çalışmakta olduđu sistemden farklı bir sistemde saklandıđı varsayıldıđını varsayalım.

Dosyaya FTP iletişim kuralı kullanılarak erişilebildiyse, 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 ayarlamaya ek olarak, aynı nesnenin QVALER özelliği aşağıdaki değerlerden birine ayarlanmalıdır:

- Kuyruk yöneticisinin adı
- Bir yıldız işareti (\*) ve ardından bir kuyruk yöneticisi grubu adı gelir.

Bu değerler, Message Queue Interface (MQI) kullanan bir istemci uygulaması tarafından yayınlanan bir MQCONN çağrısındaki **QMgrName** parametresi için kullanılacak değerlerdir. Bu değerlerin anlamı hakkında daha fazla bilgi için bkz. [MQCONN](#). You can set the QMANAGER property by using the IBM MQ JMS administration tool or IBM MQ Explorer. Alternatively, an application can set the property by calling the setQueueManager() method of the ConnectionFactory object.

Bir uygulama daha sonra ConnectionFactory nesnesinden bir bağlantı nesnesi yaratırsa, IBM MQ classes for JMS , CCDTURL özelliğiyle tanımlanan istemci kanalı tanımlama çizelgesine erişir, uygun bir istemci bağlantısı kanalı tanımlaması için çizelgeyi aramak için QMANAGER özelliğini kullanır ve sonra bir MQI kanalını kuyruk yöneticisine başlatmak için kanal tanımlamasını kullanır.

Uygulama createConnection() yöntemini çağırdığında, bir ConnectionFactory nesnesinin CCDTURL ve KANAL özelliklerinin her ikisi de belirlenemez. Her iki özellik de ayarlandıysa, yöntem bir kural dışı durum yayınlar. Değeri boş değerli, boş bir dizgi ya da tüm boş karakterleri içeren bir dizgi ise CCDTURL ya da CHANNEL özelliği ayarlanacak şekilde kabul edilir.

IBM MQ classes for JMS , istemci kanal tanımlama çizelgesinde uygun bir istemci bağlantısı kanalı tanımlaması bulunduğunda, bir MQI kanalını başlatmak için çizelgeden alınan bilgileri kullanır. ConnectionFactory nesnesinin kanalla ilgili özellikleri yok sayılır.

TLS kullanıyorsanız, özellikle aşağıdaki noktaları göz önünde bulundurun:

- Bir MQI kanalı TLS 'yi yalnızca istemci kanal tanımlama çizelgesinden alınan kanal tanımlaması, IBM MQ classes for JMS tarafından desteklenen bir CipherSpec ' in adını belirtiyorsa kullanır.
- İstemci kanalı tanımlama çizelgesi, sertifika iptal listelerini (CRL ' ler) bulduran LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) sunucularının konularıyla ilgili bilgileri de içerir. IBM MQ classes for JMS , CRL ' leri tutan LDAP sunucularına erişmek için yalnızca bu bilgileri kullanır.
- İstemci kanalı tanımlama çizelgesi bir OCSP yanıtlayıcıya ilişkin yeri de içerebilir. IBM MQ classes for JMS , bir istemci kanal tanımlama çizelgesi dosyasında OCSP bilgilerini kullanamaz. Ancak, OCSP ' yi [Online Certificate Status Protocol \(OCSP\) in Java and JMS client applications](#) bölümünde açıklandığı gibi yapılandırabilirsiniz.

İstemci kanal tanımlama çizelgesiyle TLS ' nin kullanılmasına ilişkin ek bilgi için [Genişletilmiş işlemsel istemcinin TLS kanallarıyla kullanılması](#) başlıklı konuya bakın.

Kanal çıkışlarını kullanıyorsanız aşağıdaki noktalara da dikkat edin:

- Bir MQI kanalı, istemci kanalı tanımlama çizelgesinden alınan kanal tanımlaması tarafından belirlenen kanal çıkışlarını ve ilişkili kullanıcı verilerini kullanır.
- İstemci kanal tanımlama çizelgesinden çıkarılan bir kanal tanımlaması, Java içinde yazılan kanal çıkışlarını belirleyebilir. Örneğin, bir istemci bağlantı kanalı tanımlaması yaratmak için DEFINE CHANNEL komutundaki SCYEXIT parametresinin, WMQSecurityExit arabirimini gerçekleştiren bir sınıfın adını belirtebileceği anlamına gelir. Benzer şekilde, SENDEXIT parametresi, WMQSendExit arabirimini gerçekleştiren sınıfın adını belirleyebilir ve RCVEEXIT parametresi, WMQReceiveExit arabirimini gerçekleştiren sınıfın adını belirtebilir. For more information about how to write a channel exit in Java, see ["Writing channel exits in Java for IBM MQ classes for JMS"](#) sayfa 259.

The use of channel exits written in a language other than Java is also supported. Başka bir dilde yazılmış kanal çıkışları için DEFINE CHANNEL komutunda SCYEXIT, SENDEXIT ve RCVEXIT parametrelerinin nasıl belirtileceği hakkında bilgi için [DEFINE CHANNEL](#) başlıklı konuya bakın.

#### *Otomatik JMS istemcisi yeniden bağlantısı*

Bir ağ, kuyruk yöneticisi ya da sunucu hatasının ardından otomatik olarak yeniden bağlantı kurmak için JMS istemcinizi yapılandırın.

Olağan durumda, istemci iletimi kullanılarak bir kuyruk yöneticisine bağımsız bir IBM MQ classes for JMS uygulaması bağlıysa ve bir nedenden ötürü kuyruk yöneticisi kullanılamaz duruma gelirse (örneğin, bir ağ kesintisinden, bir kuyruk yöneticisi hatasından ya da kuyruk yöneticisinin durdurulmuş olması nedeniyle), uygulama kuyruk yöneticisiyle iletişim kurmayı denediğinde IBM MQ classes for JMS bir JMSEnception yayınlayacaktır. Uygulamanın JMSEnception '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 otomatik olarak kuyruk yöneticisine yeniden bağlanmayı dener. Bu, uygulamanın yeniden bağlanmak için mantık içermesine gerek olmadığı anlamına gelir.

Bu otomatik istemci yeniden bağlantısının bu somutlamasının kullanımı, Java Platform, Enterprise Edition uygulama sunucuları içinde desteklenmez. Alternatif bir somutlama için bkz. [“Java EE ortamlarında otomatik istemci yeniden bağlantısını kullanma” sayfa 270](#) .

#### *Otomatik JMS istemcisi yeniden bağlantısının kullanılması*

Bağımsız bir IBM MQ classes for JMS uygulaması, CONNECTIONNAMELIST ya da CCDTURL özellik kümesine sahip bir Bağlantı Üreticisi kullanıyorsa, uygulama otomatik istemci yeniden bağlantı işlevini kullanmaya hak kazanır.

Yüksek kullanılabilirlik (HA) yapılandırmasının bir parçası olanlar da dahil olmak üzere kuyruk yöneticilerine yeniden bağlanmak için otomatik istemci yeniden bağlantısı kullanılabilir. HA yapılandırmaları, bir IBM MQ aygıtında çok eşgörünümlü kuyruk yöneticileri, RDQM kuyruk yöneticileri ya da HA kuyruk yöneticileri içerir.

The behavior of the automatic client reconnection functionality that is provided by the IBM MQ classes for JMS depends on the properties that follow:

#### **JMS Connection Factory özelliği TRANSPORT (Kısa ad TRAN)**

TRANSPORT, bağlantı üreticisini kullanan uygulamaların bir kuyruk yöneticisine nasıl bağlanacağını belirtir. Bu özelliğin, kullanılacak otomatik istemci yeniden bağlantısı için CLIENT değeri olarak ayarlanması gerekir. TRANSPORT özelliği için BIND, DIRECT ya da DIRECTTHHTTP değerine ayarlanmış bir Bağlantı Üreticisi kullanan bir kuyruk yöneticisine bağlanan uygulamalar için otomatik istemci yeniden bağlantısı sağlanmaz.

#### **JMS Connection Factory özelliği QMANYER (Kısa adı QMGR)**

QMANYER özelliği, Bağlantı Üreticisi 'nin bağlanacağı kuyruk yöneticisinin adını belirtir.

#### **JMS Connection Factory özelliği CONNECTIONNAMELIST (Kısa ad CRHOSTS)**

CONNECTIONNAMELIST özelliği, virgülle ayrılmış bir listedir; burada her bir giriş, CLIENT iletimi kullanılırken QASYER özelliği tarafından belirlenen kuyruk yöneticisine bağlanmak için kullanılacak anasistem adı ve kapıyla ilgili bilgiler içerir. Listede şu biçim bulunur: anasistem adı (kapı), anasistem adı (kapı).

#### **JMS Connection Factory özelliği CCDTURL (Kısa ad CCDT)**

The CCDTURL property points to the client channel definition table that the IBM MQ classes for JMS uses when it connects to a queue manager using a CCDT.

#### **JMS Connection Factory özelliği CLIENTRECONNECTOPTIONS (Kısa ad CRPT)**

CLIENTRECONNECTOPTIONS, kuyruk yöneticisi kullanılabilir duruma gelirse, IBM MQ classes for JMS ' in bir uygulama adına otomatik olarak bir kuyruk yöneticisine bağlanmayı deneyip denemeyeceğini denetler.



## **İstemci yapılandırma dosyasının Kanallarındaki DefRecon özneliđi**

DefRecon özneliđi, tüm uygulamaların otomatik olarak yeniden bađlanmasını sađlamak ya da otomatik olarak yeniden bađlanmak için yazılan uygulamalar için otomatik yeniden bađlantı özneliđini devre dıřı bırakmak üzere bir yönetim seęeneđi sađlar.

Otomatik istemci yeniden bađlantısı yalnızca, bir uygulama kuyruk yöneticisine başarıyla bađlandığında kullanılabilir.

Bir uygulama, CLIENT iletmesini kullanan bir kuyruk yöneticisine bađlandığında, uygulamanın bađlı olduđu kuyruk yöneticisi kullanılamaz duruma gelirse, otomatik istemci yeniden bađlanma özneliđinin kullanılmasının gerekip gerekmediđini saptamak için, IBM MQ classes for JMS , CLIENTRECONNECTORTIONS bađlantı üreticisi özneliđinin deđerini kullanır. Çizelge 1, CLIENTRECONNECTORTIONS özneliđi için olası deđerleri ve bu deđerlerin her biri için IBM MQ classes for JMS ' in davranışını gösterir:

<i>Çizelge 43. Olası CLIENTRECCETOPTIONS özellik deđerleri.</i>	
<b>İSTEMCİLERİYÜZLER</b>	<b>IBM MQ classes for JMSdavranışı</b>
ANY	<p>CONNECTIONNAMELIST deđeri belirlendiyse, bir anasistem adı ve kapı bileřimiyle bađlantı açmak ve kuyruk yöneticisine bađlanmak için CONNECTIONNAMELIST özneliđinin deđerini kullanın. Bu otomatik istemci yeniden bađlanma seęeneđini kullanabilmek için, QMANYER özneliđinin varsayılan deđer ya da "*" olarak ayarlanması gerekir.</p> <p>CCDTURL ayarlandıysa, CCDTURL özneliđi tarafından belirlenen istemci kanal tanımlama çizelgesini açın, çizelgede bir giriş seęin ve sonra bu giriři, bir kuyruk yöneticisine istemci bađlantısı kanalı başlatmak için kullanın. Bu otomatik istemci yeniden bađlanma seęeneđini kullanmak için, QMANYER özneliđi ařađıdaki gibi ayarlanmalıdır:</p> <ul style="list-style-type: none"><li>• Yıldız işareti (*)</li><li>• Bir yıldız işareti (*) ve ardından bir kuyruk yöneticisi grubu adı gelir.</li><li>• Boř bir dizgi ya da tüm boř karakterleri içeren bir dize</li></ul>
ASDEF	<p>Otomatik istemci yeniden bađlantısının olup olmadıđını belirlemek için DefRecon deđerini kullanın.</p>
DEVRE DIřI	<p>Herhangi bir otomatik istemci yeniden bađlantısı geręekleřtirmeyin ve uygulamaya JMSException (JMSException) döndürülmesini geręekleřtirmeyin.</p>

Çizelge 43. Olası CLIENTRECETOPTIONS özellik değerleri. (devamı var)

İSTEMCİLERİYÜZLER	IBM MQ classes for JMSdavranışı
MMGR	<p>İstemcinin aynı kuyruk yöneticisine yeniden bağlanması gerektiğini belirtir. Bu seçenek, aynı kuyruk yöneticisinin başka bir örneğine yeniden bağlanması gerektiğinde, yüksek kullanılabilirlik çözümleri için kullanılmalıdır.</p> <p>CONNECTIONNAMELIST ayarı belirlenirse, bir anasistem adı ve kapı bileşimine bağlantı açmak için CONNECTIONNAMELIST özelliğinin değerini kullanın ve QMANMICER özelliği tarafından belirlenen kuyruk yöneticisine bağlanın.</p> <p>CCDTURL ayarlandıysa, CCDTURL özelliği tarafından belirtilen istemci kanal tanımlama çizelgesini açın, QMANMSER özelliği tarafından belirtilen kuyruk yöneticisi adıyla eşleşen girişleri bulun ve o kuyruk yöneticisine bir istemci bağlantı kanalı başlatmak için bu girişleri kullanın.</p>

CONNECTIONNAMELIST ayarlanırsa, otomatik istemci yeniden bağlantısını gerçekleştirdiğinizde IBM MQ classes for JMS , hangi sistemin yeniden bağlanabileceğini belirlemek için Connection Factory özelliği CONNECTIONNAMELIST adlı bağlantı üreticisi bilgilerini kullanır.

IBM MQ classes for JMS başlangıçta, CONNECTIONNAMELIST içindeki ilk girdide belirtilen anasistem adı ve kapı kullanılarak yeniden bağlanmayı dener. Bir bağlantı kurulacaksa, IBM MQ classes for JMS daha sonra QASER özelliğinde belirtilen ada sahip kuyruk yöneticisine bağlanmayı dener. Kuyruk yöneticisine yönelik bir bağlantı kurulabiliyorsa, IBM MQ classes for JMS , otomatik istemci yeniden bağlanmadan önce uygulamanın açık olduğu tüm IBM MQ nesnelere yeniden açar ve daha önce olduğu gibi çalışmaya devam eder.

CONNECTIONNAMELIST içindeki ilk girişi kullanarak, gerekli kuyruk yöneticisine 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 yeniden denemeden önce bir süre beklemektedir. Yeni yeniden bağlanma girişimi gerçekleştirmek için, IBM MQ classes for JMS , CONNECTIONNAMELIST içindeki ilk girişle başlar. Daha sonra, CONNECTIONNAMELIST içindeki her girişi, yeniden bağlantı gerçekleşinceye ya da CONNECTIONNAMELIST değerine ulaşıncaya kadar, IBM MQ classes for JMS ' un yeniden denemeden önce bir süre beklemesi durumunda, her girişi yeniden denemeye çalışırlar.

CCDTURL ayarlandıysa, otomatik istemci yeniden bağlantısını gerçekleştirirken IBM MQ classes for JMS , hangi sistemin yeniden bağlanacağını belirlemek için CCDTURL özelliğinde belirtilen istemci kanalı tanımlama çizelgesini kullanır.

IBM MQ classes for JMS , başlangıçta istemci kanal tanımlama çizelgesini ayrıştırır ve QMANYER 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 yöneticisine yönelik bir bağlantı kurulabiliyorsa, IBM MQ classes for JMS , otomatik istemci yeniden bağlanmadan önce uygulamanın açık olduğu tüm IBM MQ nesnelere yeniden açar ve daha önce olduğu gibi çalışmaya devam eder.

Gerekli kuyruk yöneticisine bir bağlantı kurulamazsa, IBM MQ classes for JMS , istemci kanal tanımlama çizelgesinde uygun başka bir girişi arar ve bunu kullanmayı dener.

IBM MQ classes for JMS , istemci kanal tanımlama çizelgesindeki uygun girişlerin tümünü denediğinde, yeniden yeniden bağlanmayı denemeden önce bir süre beklemektedir. Yeni bir yeniden bağlanma girişimi gerçekleştirmek için, IBM MQ classes for JMS istemci kanal tanımlama çizelgesini yeniden ayrıştırır ve ilk uygun girişi dener. Daha sonra, istemci kanal tanımlama çizelgesinde, yeniden bağlantı gerçekleşinceye ya da istemci kanal tanımlama çizelgesinde son uygun giriş deneneceye kadar, IBM MQ classes for JMS

' in yeniden denemeden önce bir süre bekleyeceği şekilde, istemci kanal tanımlama çizelgesinde uygun girişleri deneyeceklerdir.

CONNECTIONNAMELISTConnectionnamelist ya da CCDURL kullanılarak, otomatik istemci yeniden bağlanma işlemi, IBM MQ classes for JMS başarıyla QManaged Manager özelliği tarafından belirtilen kuyruk yöneticisine yeniden bağlanıncaya kadar devam eder.

Varsayılan olarak, yeniden bağlanma girişimleri şu aralıklarda gerçekleşir:

- İlk deneme, 1 saniyelik ilk gecikmeden sonra, artı 250 milisaniyeye kadar rasgele bir öğeden sonra yapılır.
- İkinci deneme, ilk deneme başarısız olduktan sonra 2 saniye, artı 500 milisaniyeye kadar rasgele bir aralıkla yapılır.
- Üçüncü girişim 4 saniye, ikincisi ise ikinci girişim başarısız olduktan sonra 1 saniyeye kadar olan rasgele bir aralıktan oluşur.
- Dördüncü girişim, üçüncü girişim başarısız olduktan sonra, 8 saniye, artı 2 saniyeye kadar rasgele bir aralıkla yapılır.
- Beşinci girişim 16 saniye, artı dördüncü girişim başarısız olduktan sonra 4 saniyeye kadar rastgele bir aralık haline getirilmektedir.
- Altıncı girişim ve sonraki tüm denemeler 25 saniye, bir önceki denemeden sonra 6 saniye ve 250 milisaniyeye kadar rasgele bir aralıkla yapılır.

Yeniden bağlanma denemeleri, kısmen sabit ve kısmen rasgele olan aralıklar tarafından geciktirilir. Bu, bir kuyruk yöneticisine bağlı olan tüm IBM MQ classes for JMS uygulamalarının, artık yeniden bağlanmaktan başka bir şekilde kullanılamamasını engellemesidir.

Varsayılan değerleri artırmanız gerekiyorsa, bir kuyruk yöneticisinin kurtarılması için gereken süreyi daha doğru bir şekilde yansıtmak için ya da bir beklemedeki kuyruk yöneticisini etkin duruma getirmeniz gerekirse, istemci yapılandırma dosyasının Kanal kısmında ReconDelay özneliğini değiştirin; daha fazla bilgi için bkz. [İstemci yapılandırma dosyasının STANA kısmı](#).

Bir IBM MQ classes for JMS uygulamasının otomatik olarak yeniden bağlandıktan sonra doğru şekilde çalışmaya devam edip etmeyeceğini, tasarımına bağlıdır. Uygulamaların, otomatik yeniden bağlanma işlevini nasıl kullanabileceğini anlamak için ilgili konuları okuyun.

*Kuyruk yöneticisinin artık kullanılmadığını gösteren neden kodları*

Neden kodları, bir kuyruk yöneticisinin artık kullanılmadığını ya da otomatik IBM MQ classes for JMS yeniden bağlantı girişiminde bulunulduğunda ulaşamadığını gösterir.

[“Otomatik JMS istemcisi yeniden bağlantısı” sayfa 264](#) , JMSEExceptions 'a genel bir bakış sunar ve uygulamalarınızın otomatik olarak nasıl yeniden başlatılabileceğini ve [“Otomatik JMS istemcisi yeniden bağlantısının kullanılması” sayfa 264](#) 'daki bilgilerin otomatik istemci yeniden bağlantı için gerekleri ayrıntılarıyla ayrıntılarını verir.

Aşağıdaki bilgiler, uygulamanızın hangi IBM MQ neden kodlarını denetlediğinden emin olarak listelenir:

**RC2009**

MQRC\_CONNECTION\_BROKEN

**RC2059**

MQRC\_Q\_MGR\_NOT\_VAR

**RC2161**

MQRC\_Q\_MGR QUIESCING

**RC2162**

MQRC\_Q\_MGR\_STOPPING

**RC2202**

MQRC\_CONNECTION QUIESCING

**RC2203**

MQRC\_CONNECTION\_DURATION

## RC2223

MQRC\_Q\_MGR\_NOT\_ETKİN

## RC2279

MQRC\_CHANNEL\_STOPPED\_BY\_USER

## RC2537

MQRC\_CHANNEL\_NOT\_VAR

## RC2538

MQRC\_HOST\_NOT\_AVAM

Kurumsal uygulamalara geri verilen çoğu JMSEExceptions, neden kodunu içeren bağlantılı bir MQException içeriyor. Önceki listedeki neden kodlarına ilişkin yeniden deneme mantığını uygulamak için, kurumsal uygulamalarınız aşağıdaki örneğe benzer bir kod kullanarak bu bağlantılı kural dışı durumu kontrol etmelidir:

```
} catch (JMSEException 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 başvurular

[JMS için IBM MQ sınıfları](#)

*Java SE ve Java EE ortamlarında otomatik istemci yeniden bağlantı kullanma*

Bir Java SE ve Java EE ortamında çeşitli yüksek kullanılabilirlik (HA) ve olağanüstü durumdan kurtarma (DR) çözümlerini kolaylaştırmak için IBM MQ otomatik istemci yeniden bağlantısının kullanılmasını yapabilirsiniz.

Çeşitli HA ve DR çözümleri farklı platformlarda mevcuttur:

- **Multi** Çok eşgörümlü kuyruk yöneticileri, farklı sunucularda yapılandırılan aynı kuyruk yöneticisinin eşgörümleridir ( Çok eşgörümlü kuyruk yöneticileri konusuna bakın). Kuyruk yöneticisinin bir eşgörümlü etkin yönetim ortamı olarak tanımlanır ve başka bir yönetim ortamı yedek yönetim ortamı olarak tanımlanır. Etkin yönetim ortamı başarısız olursa, çoklu yönetim ortamı kuyruk yöneticisi otomatik olarak yedek sunucuda yeniden başlatılır.  
Hem etkin hem de beklemedeki kuyruk yöneticisinde aynı kuyruk yöneticisi tanıtıcısı (QMID) vardır. Çok eşgörümlü bir kuyruk yöneticisine bağlanan IBM MQ istemci uygulamaları, otomatik istemci yeniden bağlantısı kullanılarak kuyruk yöneticisinin yedek yönetim ortamına otomatik olarak bağlanarak otomatik olarak yeniden bağlanabilirler.
- **Linux** RDQM (eşlenmiş veri kuyruğu yöneticisi), Linux platformlarında bulunan yüksek kullanılabilirlikli bir çözümdür ( RDQM yüksek kullanılabilirlikli başlıklı konuya bakın). Bir RDQM yapılandırması, her biri kuyruk yöneticisinin bir eşgörümlü olan yüksek kullanılabilirlikli (HA) bir grupta yapılandırılan üç sunucudan oluşur. Çalışmakta olan kuyruk yöneticisi, verileri zaman uyumlu olarak diğer iki eşgörümlü eşleyen bir kuyruk yöneticidir. Bu kuyruk yöneticisini çalıştıran sunucu başarısız olursa, kuyruk yöneticisinin başka bir eşgörümlü başlatılır ve üzerinde çalışmak için geçerli veriler bulunur. Kuyruk yöneticisinin üç eşgörümlü bir kayan IP adresini paylaşır; bu nedenle, istemciler yalnızca tek bir IP adresiyle yapılandırılmalıdır. RDQM kuyruk yöneticisine bağlanan istemci uygulamaları, otomatik istemci yeniden bağlantısı kullanılarak kuyruk yöneticisinin yedek yönetim ortamına otomatik olarak yeniden bağlanarak yapılandırılabilir.
- **MQ Appliance** Bir HA çözümü, IBM MQ Araçları çifti tarafından da sağlanabilir ( IBM MQ Appliance belgelerinde [High Availability](#) (Yüksek Kullanılabilirlik) ve [Disaster Recovery](#) (Olağanüstü Durum Kurtarma) başlıklı konuya bakın). Bir HA kuyruk yöneticisi, verileri diğer aygıttaki kuyruk yöneticisinin yedek örneğine zaman uyumlu olarak eşlerken, aygıtlardan birinde çalışır. Birincil aygıt başarısız olursa, kuyruk yöneticisi otomatik olarak başlatılır ve diğer araçta çalışır. Kuyruk yöneticisinin iki eşgörümlü

bir kayan IP adresini paylayacak şekilde yapılandırılabilir; bu nedenle, istemcilerin yalnızca tek bir IP adresiyle yapılandırılması gerekir. Bir IBM MQ Appliance üzerinde bir HA kuyruk yöneticisine bağlanan istemci uygulamaları, otomatik istemci yeniden bağlantısı kullanılarak kuyruk yöneticisinin yedek yönetim ortamına otomatik olarak yeniden bağlanarak yapılandırılabilir.

### İlgili kavramlar

[Çok eşgörünümlü kuyruk yöneticileri](#)

[Otomatik istemci yeniden bağlantısı](#)

[rdqm yüksek kullanılabilirlik](#)

*Java SE ortamlarında otomatik istemci yeniden bağlantısını kullanma*

Java SE ortamlarında çalışan IBM MQ classes for JMS ' ı kullanan uygulamalar,

**CLIENTRECONNECTOPTIONS** bağlantı üreticisi özelliği aracılığıyla otomatik istemci yeniden bağlantı işlevini kullanabilir.

The connection factory property **CLIENTRECONNECTOPTIONS** uses two additional connection factory properties, **CONNECTIONNAMELIST** and **CCDTURL**, to determine how to connect to the server on which the queue manager is running.

### 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 birlikte kullanılır. Bir uygulama istemci bağlantısı yaratmak için **CONNECTIONNAMELIST** özelliğini kullandığında, IBM MQ classes for JMS , listedeki her anasisteme bağlanmayı dener. If the first queue manager host is unavailable, the IBM MQ classes for JMS attempt to connect to the next host on the list. If the end of the connection name list is reached without creating a connection, the IBM MQ classes for JMS throw the MQRC\_QMGR\_NOT\_AVAILABLE IBM MQ reason code.

Uygulamanın bağlı olduğu kuyruk yöneticisi başarısız olursa, o kuyruk yöneticisine bağlanmak için **CONNECTIONNAMELIST** kullanan uygulamalar, kuyruk yöneticisinin kullanılmadığını gösteren bir kural dışı durum alır. Uygulama, kural dışı durumu yakalayıp kullanmakta olduğu kaynakları temizlememelidir. Bağlantı yaratmak için, uygulamanın bağlantı üreticisini kullanması gerekir. Bağlantı üreticisi, liste sırasıyla her bir anasisteme bağlanmayı dener; başarısız olan kuyruk yöneticisi artık kullanılamaz. Bağlantı üreticisi, listedeki başka bir anasisteme bağlanmayı dener.

### CCDTURL özellik

**CCDTURL** özelliği, bir Client Channel Definition Table (CCDT) özelliğini gösteren bir Uniform Resource Locator (URL) içerir; bu özellik **QMANAGER** özelliğiyle kullanılır. CCDT, bir IBM MQ sisteminde tanımlı bir kuyruk yöneticisine bağlanmak için kullanılan istemci kanallarının bir listesini içerir. CCDT ' lerin IBM MQ classes for JMS tarafından nasıl kullanılırsa ilgili bilgi için bkz. [“Using a client channel definition table with IBM MQ classes for JMS” sayfa 262.](#)

## Using the CLIENTRECONNECTOPTIONS property to enable automatic client reconnection within the IBM MQ classes for JMS

**CLIENTRECONNECTOPTIONS** özelliği, IBM MQ classes for JMS içinde otomatik istemci yeniden bağlantısını etkinleştirmek için kullanılır. Bu özelliğe ilişkin olası değerler aşağıdaki gibidir:

#### ASDEF

Otomatik istemci yeniden bağlantı davranışı, IBM MQ istemci yapılandırma dosyasının (mqclient.ini) kanal kısmında belirtilen varsayılan değer tarafından tanımlanır.

#### DEVRE DIŞI

Otomatik istemci yeniden bağlantısı geçersiz kılındı.

#### QMGR

IBM MQ classes for JMS , aşağıdaki seçeneklerden birini kullanarak, bağlı olduğu kuyruk yöneticisiyle aynı kuyruk yöneticisi tanıtıcısına sahip bir kuyruk yöneticisine bağlanmayı dener:

- **CONNECTIONNAMELIST** özelliği ve **CHANNEL** özelliğinde tanımlanan kanal.
- **CCDTURL** özelliğinde tanımlanan CCDT.

#### ANY

IBM MQ classes for JMS , **CONNECTIONNAMELIST** özelliğinin ya da **CCDTURL** özelliğinin kullanılmasıyla aynı adı taşıyan bir kuyruk yöneticisine yeniden bağlanma girişiminde bulunmayı dener.

#### İlgili bilgiler

İstemci yapılandırma dosyasının STANA kısmı

#### *Java EE ortamlarında otomatik istemci yeniden bağlantısını kullanma*

Java EE (Java Platform, Enterprise Edition) ortamlarında devreye alınabilen IBM MQ kaynak bağdaştırıcısı ve WebSphere Application Server IBM MQ ileti sistemi sağlayıcısı, IBM MQ kuyruk yöneticileriyle iletişim kurmak için IBM MQ classes for JMS ' i kullanır. IBM MQ kaynak bağdaştırıcısı ve WebSphere Application Server IBM MQ ileti sistemi sağlayıcısı, otomatik istemci yeniden bağlantısı için destek sağlar.

Bir Java EE ortamında otomatik istemci yeniden bağlantısı sağlamak için kullanılacak seçenekler şunlardır:

- Etkinleştirme belirtimi
- WebSphere Application Server dinleyici kapıları
- Kurumsal JavaBeans ve web tabanlı uygulamalar
- İstemci kapsayıcıları içinde çalışan uygulamalar

**Not:** IBM MQ classes for JMS tarafından sağlanan işlevselliği kullanarak etkinleştirme belirtileriyle otomatik istemci yeniden bağlantısı desteklenmiyor. The IBM MQ resource adapter provides its own mechanism for reconnecting activation specifications if the queue manager that the activation specification was connecting to becomes unavailable.

Bu mekanizma aşağıdaki şekilde 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 yapılandırması” sayfa 427.

İletiyi yönlendirilen bir bean uygulamasının `onMessage()` yönteminde ya da Java Platform, Enterprise Edition ortamında çalışan başka bir uygulama içinde otomatik istemci yeniden bağlantısı kullanılması desteklenmez. Bağlandığı kuyruk yöneticisi kullanılamaz duruma gelirse, uygulamanın kendi yeniden bağlanma mantığını gerçekleştirmesi gerekir.

#### *Java EE ortamlarında otomatik istemci yeniden bağlantısı desteği*

WebSphere Application Server gibi Java EE ortamlarında, IBM MQ kaynak bağdaştırıcısı ve WebSphere Application Server IBM MQ ileti sistemi sağlayıcısı, otomatik istemci yeniden bağlantısı için destek sağlar. Ancak bazı durumlarda, kısıtlamalar bu destek için geçerli olur.

Java EE ortamlarında ve WebSphere Application Server IBM MQ ileti alışverişi sağlayıcısında devreye alınabilen IBM MQ kaynak bağdaştırıcısı, IBM MQ kuyruk yöneticileriyle iletişim kurmak için IBM MQ classes for JMS ' i kullanır.

Aşağıdaki çizelge, IBM MQ kaynak bağdaştırıcısı ve WebSphere Application Server IBM MQ ileti alışverişi sağlayıcısının otomatik istemci yeniden bağlantısı için destek sağladığı desteği özetler.

Çizelge 44. Java EE ortamlarındaki otomatik istemci yeniden bağlantı seçeneklerine ilişkin desteğin özeti

Otomatik yeniden bağlantı için seçenekler	CONNECTIONNAMELIST özelliği	CCDTURL özelliği	CLIENTRECONNECTPTIONS özelliği	Otomatik istemci yeniden bağlantılarına alternatif yaklaşım
Etkinleştirme belirtileri	Kısıtlamalarla desteklenir	Kısıtlamalarla desteklenir	Desteklenmiyor	Java EE ortamı ve etkinleştirme belirtileri kendi yeniden bağlantı mekanizmasını 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	Uygulamanın kendi yeniden bağlanma mantığını gerçekleştirmesi gerekir
İstemci kapsayıcıları içinde çalışan uygulamalar	Destekleniyor	Destekleniyor	Destekleniyor	Burada geçerli değil

Message-driven bean applications that are installed in a Java EE environment, such as IBM MQ classes for JMS, can use activation specifications to process messages on an IBM MQ system. Etkinleştirme belirtileri, bir IBM MQ sistemine gelen iletileri saptamak ve bunları işlemek üzere ileti odaklı Bean 'lere teslim etmek için kullanılır. Message-driven beans can also make more connections to IBM MQ systems from inside their **onMessage()** method. Bu bağlantıların otomatik istemci yeniden bağlantısını nasıl kullanabileceğiyle ilgili daha fazla bilgi için bakınız: [Enterprise JavaBeans ve Web tabanlı uygulamalar](#).

#### *Etkinleştirme belirtileri*

For activation specifications, the **CONNECTIONNAMELIST** and **CCDTURL** properties are supported with restrictions and the **CLIENTRECONNECTPTIONS** property is not supported.

Message-driven bean (MDB) applications that are installed in a Java EE environment, such as WebSphere Application Server, can use activation specifications to process messages on an IBM MQ system.

Etkinleştirme belirtileri, bir IBM MQ sistemine gelen iletileri saptamak için kullanılır ve bunları işlemek üzere MDBS 'lere teslim etmek için kullanılır. Bu bölümde, etkinleştirme belirtiminin IBM MQ sistemini nasıl izlediğini ele alan bölüm ele alır.

MDBs can also make additional connections to IBM MQ systems from inside their **onMessage()** method.

Bu bağlantıların otomatik istemci yeniden bağlantısını nasıl kullanabilmesiyle ilgili ayrıntılar [“Kurumsal JavaBeans ve web tabanlı uygulamalar”](#) sayfa 275’inde bulunabilir.

### **CONNECTIONNAMELIST özellik**

Başlatma sırasında, etkinleştirme belirtimi kuyruk yöneticisine bağlanmayı dener:

- **QMANAGER** özelliğinde belirtilen
- Channel mentioned in the **CHANNEL** property

- **CONNECTIONNAMELIST** içindeki ilk girişten ana makine adı ve kapı bilgileri

Etkinleştirme belirtimi listedeki ilk girişi kullanarak kuyruk yöneticisine bağlanamazsa, etkinleştirme belirtimi ikinci girişe geçer ve bu nedenle, kuyruk yöneticisine bir bağlantı yapılıncaya ya da listenin sonuna ulaşıncaya kadar bu giriş üzerinde devam eder.

Etkinleştirme belirtimi belirtilen kuyruk yöneticisine bağlanamazsa, **CONNECTIONNAMELIST** içindeki girişlerden herhangi birini kullanarak etkinleştirme belirtimi durdurulur ve yeniden başlatılmalıdır.

Etkinleştirme belirtimi çalıştırıldıktan sonra, etkinleştirme belirtimi IBM MQ sisteminden iletileri alır ve iletileri işlenmek üzere bir MDB ' ye gönderir.

Kuyruk yöneticisi bir ileti işlenirken başarısız olursa, Java EE ortamı hatayı algılar ve etkinleştirme belirtimini yeniden bağlamaya çalışır.

The activation specification uses the information in the **CONNECTIONNAMELIST** property as before, when the activation specification performs the reconnection attempts.

Etkinleştirme belirtimi **CONNECTIONNAMELIST** içindeki tüm girişleri denerse ve hala kuyruk yöneticisine bağlanamazsa, 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 art arda yeniden bağlantı girişimi sayısını tanımlar ve el ile yeniden başlatma gerektirir.

Etkinleştirme belirtimi bir IBM MQ sistemine yeniden bağlandıktan sonra, Java EE ortamı gereken işlem temizliği gerçekleştirir ve işlenmek üzere iletileri MDBS ' lere teslim etmeye devam eder.

İşlem temizliğinin doğru şekilde çalışması için, Java EE ortamının, başarısız olan kuyruk yöneticisine ilişkin günlüklere erişebilmeleri gerekir.

Etkinleştirme belirtimleri XA hareketlerine katılan işlemsel MDB'lerle birlikte kullanılıyorsa ve çok eşgözümlü bir kuyruk yöneticisine bağlanıyorsa, **CONNECTIONNAMELIST** hem etkin hem de beklemedeki kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

This means that the Java EE environment can access the queue manager logs if the environment needs to perform transaction recovery, regardless of which queue manager the environment reconnects to following a failure.

İşlemsel Maximo İş DBM ' leri bağımsız kuyruk yöneticileriyle kullanılıyorsa, etkinleştirme belirtiminin, başarısızlığın ardından aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlandığından emin olmak 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 bu nedenle, kuyruk yöneticisine bir bağlantı yapılıncaya ya da çizelgenin sonuna ulaşıncaya kadar bu giriş için devam eder.

Etkinleştirme belirtimi belirtilen kuyruk yöneticisine bağlanamazsa, CCDT ' deki girişlerden herhangi birini kullanarak etkinleştirme belirtimi durdurulur ve yeniden başlatılmalıdır.

Etkinleştirme belirtimi çalıştırıldıktan sonra, etkinleştirme belirtimi IBM MQ sisteminden iletileri alır ve iletileri işlenmek üzere bir MDB ' ye gönderir.

Kuyruk yöneticisi bir ileti işlenirken başarısız olursa, Java EE ortamı hatayı algılar ve etkinleştirme belirtimini yeniden bağlamaya çalışır.

Etkinleştirme belirtimi, daha önce olduğu gibi CCDT özelinde yer alan bilgileri kullanır; etkinleştirme belirtimi yeniden bağlanma denemelerini gerçekleştirir.



Etkinleştirme belirtimi CCDT ' deki tüm girişleri denerse ve hala kuyruk yöneticisine bağlanamazsa, etkinleştirme belirtimi yeniden denemeden önce IBM MQ kaynak bağdaştırıcısı özelliği **reconnectionRetryInterval** tarafından belirtilen süreye kadar bekler.

IBM MQ kaynak bağdaştırıcısı özelliği **reconnectionRetryCount** , bir etkinleştirme belirtimi durdurulmadan önce yapılacak art arda yeniden bağlantı girişimi sayısını tanımlar ve el ile yeniden başlatma gerektirir.

Etkinleştirme belirtimi bir IBM MQ sistemine yeniden bağlandıktan sonra, Java EE ortamı gereken işlem temizliği gerçekleştirir ve işlenmek üzere iletileri MDBS ' lere teslim etmeye devam eder.

İşlem temizliğinin doğru şekilde çalışması için, Java EE ortamının, başarısız olan kuyruk yöneticisine ilişkin günlüklere erişebilmeleri gerekir.

Etkinleştirme belirtileri XA hareketlerine katılan işlemsel MDB'lerle kullanılıyorsa ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, CCDT hem etkin hem de beklemedeki kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

This means that the Java EE environment can access the queue manager logs if the environment needs to perform transaction recovery, regardless of which queue manager the environment reconnects to following a failure.

İşlemsel Maximo İş BDL 'leri bağımsız kuyruk yöneticileriyle kullanılıyorsa, etkinleştirme belirtiminin bir hata sonrasında aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlanmasını sağlamak için CCDT ' nin tek bir giriş içermesi gerekir.

Bağlantılarla aynı etkin kuyruk yöneticisiyle bağlantı kurmak için, etkinleştirme belirtileriyle birlikte kullanılan CCDT ' lerdeki **AFFINITY** özelliği için varsayılan *TERCIH edilen* değerini ayarladığınızdan emin olun.

## **CLIENTRECONNECTOPTIONS özellik**

Etkinleştirme belirtileri kendi yeniden bağlantı işlevselliğini sağlar. Sağlanan işlevsellik, bağlı oldukları kuyruk yöneticisi başarısız olursa, belirtilerin bir IBM MQ sistemine otomatik olarak yeniden bağlanmasını sağlar.

Bu yüzden, IBM MQ classes for JMS tarafından sağlanan otomatik istemci yeniden bağlanma işlevselliği desteklenmez.

Java EE' de kullanılan tüm etkinleştirme belirtileri için **CLIENTRECONNECTOPTIONS** özelliğini *DISABLE* olarak ayarlamalısınız.

### *WebSphere Application Server dinleyici kapıları*

WebSphere Application Server ' ta kurulu olan ileti odaklı Bean (MDB) uygulamaları, bir IBM MQ sistemindeki iletileri işlemek için dinleyici kapılarını da kullanabilir.

Dinleyici kapıları, bir IBM MQ sistemine gelen iletileri saptamak için kullanılır ve bunları işlenmek üzere MDBS ' lere teslim eder. Bu konu, dinleyici kapısının IBM MQ sistemini nasıl izlediğini açıklar.

MDBs can also make additional connections to IBM MQ systems from inside their onMessage () method.

Bu bağlantıların otomatik istemci yeniden bağlantısını nasıl kullanabileceğiyle ilgili daha fazla bilgi için bkz. [“Kurumsal JavaBeans ve web tabanlı uygulamalar” sayfa 275](#)

WebSphere Application Server dinleyici kapıları için:

- **CONNECTIONNAMELIST** and **CCDTURL** are supported with restrictions
- **CLIENTRECONNECTOPTIONS** desteklenmiyor

## **CONNECTIONNAMELIST özellik**

Dinleyici kapıları, IBM MQ ile bağlantı kurulurken JMS bağlantı havuzlarından kullanılmasını sağlar; bu nedenle, bağlantı havuzlarının kullanılmasının etkisine tabidir. Ek bilgi için [“Etkinleştirme belirtileri” sayfa 271](#) ' e bakın.

Serbest bağlantı yoksa ve bağlantı sayısı üst sınırı bu bağlantı üreticisinden henüz oluşturulmamışsa, **CONNECTIONNAMELIST** özelliği, IBM MQ ile yeni bir bağlantı kurmak için kullanılır.

**CONNECTIONNAMELIST** sistemindeki tüm IBM MQ sistemlerine erişilemiyorsa, dinleyici kapısı durur.

Dinleyici kapısı daha sonra, **RECOVERY . RETRY . INTERVAL** ileti dinleyici hizmeti özel özelliğinin belirlediği süreye ilişkin bekler ve yeniden yeniden bağlanmayı dener.

Bu yeniden bağlanma girişimi, bağlantı havuzundaki herhangi bir serbest bağlantı olup olmadığını denetler; bağlantı girişimi arasında bir bağlantı varsa, bağlantı havuzunda herhangi bir bağlantı olup olmadığını denetler. Kullanılmıyorsa, dinleyici kapısı **CONNECTIONNAMELIST** ' yi önceki gibi kullanır.

Dinleyici kapısı bir IBM MQ sistemine yeniden bağlandıktan sonra, Java EE ortamı gereken işlem temizliği gerçekleştirir ve daha sonra işlenmek üzere iletileri MDBS ' lere teslim etmeye devam eder.

İşlem temizliğinin doğru şekilde çalışması için, Java EE ortamının, başarısız olan kuyruk yöneticisine ilişkin günlüklere erişebilmeleri gerekir.

Dinleyici kapıları XA hareketlerine katılan işlemsel MDB'lerle kullanılıyorsa ve bir **çok eşgörünümli kuyruk yöneticisine** bağlanıyorsa, **CONNECTIONNAMELIST** , hem etkin hem de beklemedeki kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

This means that the Java EE environment can access the queue manager logs if the environment needs to perform transaction recovery, regardless of which queue manager the environment reconnects to following a failure.

İşlemsel Maximo İş DBM ' leri bağımsız kuyruk yöneticileriyle kullanılıyorsa, etkinleştirme belirtiminin, başarısızlığın ardından aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlandığından emin olmak için **CONNECTIONNAMELIST** özelliğinin tek bir giriş içermesi gerekir.

## CCDTURL özellik

Başlatma sırasında, dinleyici kapısı CCDT ' deki ilk girişi kullanarak **QMANAGER** özelliğinde belirtilen kuyruk yöneticisine bağlanmayı dener.

İletişimci kapısı, çizelgedeki ilk girişi kullanarak kuyruk yöneticisine bağlanamıyorsa, dinleyici kapısı kuyruk yöneticisine bağlantı yapılıncaya ya da çizelgenin sonuna ulaşıncaya kadar, ikinci girişe, vb. bir girişlere geçer.

Dinleyici kapısı CCDT ' deki girişlerden herhangi birini kullanarak belirtilen kuyruk yöneticisine bağlanamazsa, dinleyici kapısı durur.

Dinleyici kapısı daha sonra, **RECOVERY . RETRY . INTERVAL** ileti dinleyici hizmeti özel özelliğinin belirlediği süreye ilişkin bekler ve yeniden yeniden bağlanmayı dener.

Bu yeniden bağlanma girişimi CCDT ' deki tüm girişlerde daha önce olduğu gibi çalışır.

Once the Listener Port is running, it gets messages from the IBM MQ system and delivers them to an MDB for processing.

Bir ileti işlenirken kuyruk yöneticisi başarısız olursa, Java EE ortamı hatayı algılar ve dinleyici kapısını yeniden bağlamayı dener. İletişimci kapısı, yeniden bağlanma denemelerini gerçekleştirirken CCDT ' deki bilgileri kullanır.

Dinleyici kapısı CCDT ' deki tüm girişleri denerse ve hala kuyruk yöneticisine bağlanamazsa, kapı yeniden denemeden önce **RECOVERY . RETRY . INTERVAL** özelliği tarafından belirtilen süre boyunca bekler.

**MAX . RECOVERY . RETRIES** ileti dinleyici hizmeti özelliği, bir dinleyici kapısı durmadan önce yapılan art arda yeniden bağlantı girişimi sayısını tanımlar ve el ile yeniden başlatma gerektirir.

Dinleyici kapısı bir IBM MQ sistemine yeniden bağlandıktan sonra, Java EE ortamı gereken işlem temizliği gerçekleştirir ve daha sonra işlenmek üzere iletileri MDBS ' lere teslim etmeye devam eder.

İşlem temizliğinin doğru şekilde çalışması için, Java EE ortamının, başarısız olan kuyruk yöneticisine ilişkin günlüklere erişebilmeleri gerekir.

Dinleyici kapıları XA hareketlerine katılan işlemsel MDB'lerle kullanılıyorsa ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, CCDT hem etkin hem de beklemedeki kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

This means that the Java EE environment can access the queue manager logs if the environment needs to perform transaction recovery, regardless of which queue manager the environment reconnects to following a failure.

İşlemsel Maximo İş BDL ' leri bağımsız kuyruk yöneticileriyle kullanılıyorsa, dinleyici kapısının bir arizeyi izleyen aynı sistemde çalışmakta olan aynı kuyruk yöneticisine her zaman yeniden bağlandığından emin olmak için CCDT tek bir giriş içermelidir.

İletişimci kapılarıyla birlikte kullanılan CCDT ' lerdeki **AFFINITY** özelliği için varsayılan *TERCIH edilen* değerini ayarladığınızdan emin olun; böylece bağlantılar aynı etkin kuyruk yöneticisiyle aynı olur.

## **CLIENTRECONNECTOPTIONS özellik**

Dinleyici kapıları kendi yeniden bağlanma işlevselliğini sağlar. Sağlanan işlevsellik, bağlı oldukları kuyruk yöneticisi başarısız olduğunda, dinleyici kapılarının otomatik olarak bir IBM MQ sistemine yeniden bağlanmasını sağlar.

Bu yüzden, IBM MQ classes for JMS tarafından sağlanan otomatik istemci yeniden bağlanma işlevselliği desteklenmez.

You must set the **CLIENTRECONNECTOPTIONS** property to *DEVRE Dışı* for all listener ports that are used in the Java EE.

### *Kurumsal JavaBeans ve web tabanlı uygulamalar*

Kurumsal JavaBean (EJB) uygulamaları ve sunucu uygulamaları gibi web kapsayıcısında çalışan uygulamalar, IBM MQ kuyruk yöneticisine bağlantı yaratmak için bir JMS bağlantı üreticisi kullanır.

EJB ' ler ve Web tabanlı uygulamalar için aşağıdaki kısıtlamalar geçerlidir:

- **CONNECTIONNAMELIST** and **CCDTURL** are supported with restrictions
- **CLIENTRECONNECTOPTIONS** desteklenmiyor

## **CONNECTIONNAMELIST özellik**

Java EE ortamı JMS bağlantıları için bir bağlantı havuzu sağlıyorsa, bunun **CONNECTIONNAMELIST** özelliğinin davranışını nasıl etkilediği hakkında bilgi için [“Bağlantı havuzundaki CONNECTIONNAMELIST ya da CCDT ' nin kullanılması” sayfa 276 ' e bakın.](#)

Java EE ortamı bir JMS bağlantı havuzu sağlamazsa. the application uses the **CONNECTIONNAMELIST** property in the same way as Java SE applications.

Uygulamalar XA hareketlerine katılan işlemsel MDB'lerle kullanılıyorsa ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, **CONNECTIONNAMELIST** hem etkin hem de beklemedeki kuyruk yöneticisi yönetim ortamı için bir giriş içermelidir.

This means that the Java EE environment can access the queue manager logs if the environment needs to perform transaction recovery, regardless of which queue manager the environment reconnects to following a failure.

Uygulamalar bağımsız kuyruk yöneticileriyle kullanılıyorsa, uygulamanın aynı kuyruk yöneticisine her zaman yeniden bağlandığından emin olmak için, **CONNECTIONNAMELIST** özelliği tek bir giriş içermelidir. Hata nedeniyle, uygulamanın aynı kuyruk yöneticisine her zaman yeniden bağlanması gerekir.

## **CCDTURL özellik**

Java EE ortamı, JMS bağlantıları için bir bağlantı havuzu sağlıyorsa, bunun **CCDTURL** özelliğinin davranışını nasıl etkilediği hakkında bilgi için [“Bağlantı havuzundaki CONNECTIONNAMELIST ya da CCDT ' nin kullanılması” sayfa 276 konusuna bakın.](#)

Java EE ortamı bir JMS bağlantı havuzu sağlamazsa. the application uses the **CCDTURL** property in the same way as Java SE applications.

Uygulamalar XA hareketlerine katılan ve çok eşgörünümlü bir kuyruk yöneticisine bağlanıyorsa, CCDT ' nin hem etkin hem de beklemedeki kuyruk yöneticisi yönetim ortamı için bir giriş içermeleri gerekir.

This means that the Java EE environment can access the queue manager logs if the environment needs to perform transaction recovery, regardless of which queue manager the environment reconnects to following a failure.

Uygulamalar bağımsız kuyruk yöneticileriyle kullanılıyorsa CCDT tek bir giriş içermeli ve etkinleştirme belirtiminin, başarısızlığın ardından aynı sistemde çalışan aynı kuyruk yöneticisine her zaman yeniden bağlandığından emin olmak için.

## **CLIENTRECONNECTOPTIONS özellik**

Web kapsayıcısında çalışan EJB ' ler ya da uygulamalar tarafından kullanılan tüm JMS bağlantı fabrikaları için **CLIENTRECONNECTOPTIONS** özelliğini *DISABLE* olarak ayarlamalısınız.

Yeni bir kuyruk yöneticisine otomatik olarak yeniden bağlanmayı gerektiren uygulamalar, eğer kullandıkları kuyruk yöneticisi başarısız olursa, kendi yeniden bağlanma mantığını gerçekleştirmeleri gerekir. Ek bilgi için [“Java EE uygulamasında yeniden bağlantı mantığını uygulama” sayfa 277](#) başlıklı konuya bakın.

Senaryolar: [IBM MQ ile WebSphere Application Server](#)

Scenarios: [WebSphere Application Server Liberty profile with IBM MQ](#)

*İstemci kapsayıcıları içinde çalışan uygulamalar*

WebSphere Application Server gibi bazı Java EE ortamları, Java SE uygulamalarını çalıştırmak için kullanılabilir bir istemci kapsayıcısı sağlar.

Bu ortamların içinde çalışan uygulamalar, bir IBM MQ kuyruk yöneticisine bağlanmak için bir JMS bağlantı üreticisi kullanır.

İstemci taşıyıcıları içinde çalışan uygulamalar için:

- **CONNECTIONNAMELIST** ve **CCDTURL** tam olarak desteklenmektedir
- **CLIENTRECONNECTOPTIONS** tam olarak destekleniyor

## **CONNECTIONNAMELIST özellik**

Java EE ortamı JMS bağlantıları için bir bağlantı havuzu sağlıyorsa, bunun **CONNECTIONNAMELIST** özelliğinin davranışını nasıl etkilediği hakkında bilgi için [“Bağlantı havuzundaki CONNECTIONNAMELIST ya da CCDT ' nin kullanılması” sayfa 276](#) ' e bakın.

Java EE ortamı bir JMS bağlantı havuzu sağlamazsa. the application uses the **CONNECTIONNAMELIST** property in the same way as Java SE applications.

## **CCDTURL özellik**

Java EE ortamı, JMS bağlantıları için bir bağlantı havuzu sağlıyorsa, bunun **CCDTURL** özelliğinin davranışını nasıl etkilediği hakkında bilgi için [“Bağlantı havuzundaki CONNECTIONNAMELIST ya da CCDT ' nin kullanılması” sayfa 276](#) konusuna bakın.

Java EE ortamı bir JMS bağlantı havuzu sağlamazsa. the application uses the **CCDTURL** property in the same way as Java SE applications.

*Bağlantı havuzundaki CONNECTIONNAMELIST ya da CCDT ' nin kullanılması*

Bazı Java EE ortamları (örneğin, WebSphere Application Server), bir JMS bağlantı havuzu sağlar. Java SE uygulamalarını çalıştırmak için kullanılabilir taşıyıcı.

Java EE ortamında tanımlanmış bir bağlantı üreticisini kullanarak bağlantı yaratan uygulamalar, bu bağlantı üreticisine ilişkin bağlantı havuzundan varolan bir serbest bağlantıyı ya da bağlantı havuzunda uygun bir bağlantı yoksa, yeni bir bağlantı sağlar.

Bağlantı üreticisi, tanımlanmış **CONNECTIONNAMELIST** ya da **CCDTURL** özelliği ile yapılandırıldıysa, bunun etkileri olabilir.

Bağlantı üreticisi bir bağlantı oluşturmak için ilk kez kullanıldığında, Java EE ortamı **CONNECTIONNAMELIST** 'i kullanır. or the **CCDTURL** to create a new connection to the IBM MQ system. Bu bağlantı artık gerekli olmadığında, bağlantının yeniden kullanılmak üzere kullanılabilir hale geldiği bağlantı havuzuna geri gönderilir.

Bağlantı üreticisinden başka bir bağlantı yaratılırsa, Java EE ortamı, yeni bir bağlantı yaratmak için **CONNECTIONNAMELIST** ya da **CCDTURL** özelliklerini kullanmak yerine bağlantı havuzundan bağlantıyı döndürür.

Kuyruk yöneticisi yönetim ortamı başarısız olduğunda bir bağlantı kullanılıyorsa, bağlantı atılır. Ancak, bağlantı havuzunun içeriği olmayabilir; bu durumda, havuz, artık çalışmayan bir kuyruk yöneticisine bağlantı içerebilir.

Bu durumda, bağlantı üreticisinden bir bağlantı yaratma isteğinde bir sonraki istek yapıldığında, başarısız olan kuyruk yöneticisine yönelik bir bağlantı döndürülür. Kuyruk yöneticisi artık çalışır durumda olmadığı için, bağlantının atılmasına neden olan, bu bağlantıyı kullanmaya yönelik girişimler başarısız olur.

Only when the connection pool is empty, will the Java EE environment use the **CONNECTIONNAMELIST** or **CCDTURL** properties to create a new connection to IBM MQ.

**CONNECTIONNAMELIST** ve **CCDT** ' lerin JMS bağlantıları yaratmak için kullanıldığı yol nedeniyle, farklı IBM MQ sistemlerine bağlantılar içeren bir bağlantı havuzuna da sahip olmak da olanaklıdır.

Örneğin, **CONNECTIONNAMELIST** özelliği ile bir bağlantı üreticinin yapılandırıldığını varsayalım.

```
CONNECTIONNAMELIST = hostname1(port1), hostname2(port2)
```

Suppose that the first time an application tries to create a connection to a stand-alone queue manager from this connection factory, the queue manager running on the system `hostname1(port1)` is not accessible. Bu, uygulamanın, `hostname2(port2)` üzerinde çalışan kuyruk yöneticisine yönelik bir bağlantı ile sona erdiği anlamına gelir.

Başka bir uygulama şimdi ortaya çıkar ve aynı bağlantı üreticisinden bir JMS bağlantısı oluşturur. The queue manager on `hostname1(port1)` is now available, so a new JMS connection is created to this IBM MQ system and is returned to the application.

Her iki uygulama da bittiğinde, bağlantıların bağlantı havuzuna geri döndürülmesine neden olan JMS Connections 'ı kapatıyorlar.

Sonuçta, bağlantı üreticimiz için bağlantı havuzunun şu anda iki JMS bağlantısı olduğu ortaya çıktı:

- One connection to the queue manager running on `hostname1(port1)`
- One connection to the queue manager running on `hostname2(port2)`

Bu, hareket kurtarması ile ilgili sorunlara yol açabilir. Java EE sisteminin bir hareketi geriye işlemesi gerekiyorsa, bu işlem, hareket günlüklerine erişimi olan bir kuyruk yöneticisine bağlanabilmelidir.

#### *Java EE uygulamasında yeniden bağlantı mantığını uygulama*

Bir kuyruk yöneticisinin başarısız olması durumunda otomatik olarak yeniden bağlantı kurmak isteyen kurumsal JavaBeans ve web tabanlı uygulamalar, kendi yeniden bağlanma mantığını gerçekleştirmeye gerek duyarlar.

Aşağıdaki seçenekler, bunu nasıl gerçekleştirebileceğiyle ilgili daha fazla bilgi sağlar.

## Uygulamanın başarısız olmasına izin ver

This approach requires no application changes, but does require an administrative reconfiguration of the connection factory definition to include the **CONNECTIONNAMELIST** property. Ancak, bu yaklaşım, çağırının bir hatayı uygun şekilde işleyebilmesini gerektirir. Bu sorunun, 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 (JMSEException je) {
            // process exception
        }
    }
}
```

The preceding code assumes that the connection factory, this servlet is using, has the **CONNECTIONNAMELIST** property defined.

Sunucu uygulaması ilk işlemler sırasında, aynı kuyruk yöneticisine bağlanan diğer uygulamalardan havuza gönderilen bağlantıların olmadığını varsayarak, **CONNECTIONNAMELIST** özelliği kullanılarak yeni bir bağlantı yaratılır.

When the connection is released following a `close()` call, this connection is returned to the pool and reused the next time the servlet runs - without referring to the **CONNECTIONNAMELIST** - until a connection failure occurs, at which point a `CONNECTION_ERROR_OCCURRED` event is generated. Bu olay, havuzun başarısız olan bağlantıyı yok etmesi için bilgi isteminde bulunur.

When the application next runs, no pooled connection is available and the **CONNECTIONNAMELIST** is used to connect to the first available queue manager. Kuyruk yöneticisi başarısız olursa (örneğin, hata geçici bir ağ hatası değildi), sunucu uygulaması kullanılabilir olduğunda yedek eşgörünüme bağlanır.

Veritabanları gibi başka kaynaklar da uygulamaya dahil edildiyse, uygulama sunucusunun hareketi geriye işlemesi gerektiğini belirtmek uygun olabilir.

## Uygulama içindeki yeniden bağlantıyı işle

Çağırın, sunucu uygulamacığıdaki bir başarısızlığı işleyemiyorsa, yeniden bağlantı uygulamanın uygulama içinde işlenmesini sağlar. Aşağıdaki örnekte gösterildiği gibi, uygulama içindeki bir yeniden bağlantıyı işlemek için uygulamanın yeni bir bağlantı istemesini gerektirir; böylece, JNDI ' dan yukarıya bakan ve JMSCMQ0001:WebSphere MQ çağırısı, compcode '2' ('MQCC\_FAILED ') neden ' 2009 ' ('MQRC\_CONNECTION\_BROKEN') ile başarısız oldu.gibi bir JMSEException ile ilgili olan bağlantı üreticisini önbelleğe almak için bu uygulamayı önbelleğe almak 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 (JMSEException je) {
        // drive reconnection
        setupResources();
    }
}
}

```

Aşağıdaki örnekte `setupResources()`, JMS nesnelərini oluşturur ve anlık yeniden bağlantı işlemini işlemek için bir uyku ve yeniden deneme döngüsü içerir. Uygulamada, bu yöntem birçok yeniden bağlanma girişimlerini önler. Çıkış koşullarının netlik 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 (JMSEException je) {
            // sleep and then have another attempt
            try {Thread.sleep(30*1000);} catch (InterruptedException ie) {}
        }
    }
}

```

Uygulama yeniden bağlantıyı yönetiyorsa, uygulamanın diğer kaynaklarda tutulan bağlantıları (bu kaynakların diğer IBM MQ kuyruk yöneticileri ya da veritabanları gibi arka uç hizmetleri) yayınlamaması önemlidir. Yeni bir IBM MQ kuyruk yöneticisi yönetim ortamına yeniden bağlantı kurulduğunda bu bağlantıları yeniden oluşturmanız gerekir. Bağlantıları yeniden oluşturmuyorsanız, yeniden bağlanma girişimi sırasında uygulama sunucusu kaynakları gereksiz tutulur ve yeniden kullanılanlarla zaman aşımına uğramış olabilir.

## WorkManager' in kullanılması

İşleme süresinin birkaç saniyeden daha uzun olduğu, uzun ömürlü uygulamalar (örneğin, toplu işleme) için, WebSphere Application Server WorkManager kullanılabilir. WebSphere Application Server için bir kod parçası örneği aşağıdaki gibidir:

```

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 şunları 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, iş arabirimi aracılığıyla gerçekleştirilmektedir:

```

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 (JMSEException je) {
                // drive reconnection
                setupResources();
            }
        }

        public boolean isRunning() {return !sendComplete;}

        public void release() {sendComplete = true;}
    }
}

```

Toplu işleme uzun bir süre (örneğin, büyük iletiler, yavaş ağ ya da kapsamlı veritabanı erişimi) (özellikle de yavaş yedek sunucuyla birleştiğinde) çalıştırılırsa, sunucu, aşağıdaki örneğe benzer şekilde, askıda kalma iş parçacığı uyarılarını başlatmaya başlar:

WSVR0605W: İş parçacığı "WorkManager.DefaultWorkManager : 0" (00000035), 694061 milisaniye için etkin ve askıda olabilir. Sunucunun toplam içinde asılabilecek 1 iş parçacığı/iş parçacığı var.

Bu uyarılar, toplu iş büyüklüğü azaltılarak ya da askıda kalma iş parçacığı zamanasını artırarak en aza indirilebilir. Ancak, bu işlemi bir EJB (toplu gönderme için) ya da ileti odaklı bean (tüketmek ya da tüketmek ve yanıtlamak için) içinde uyguladığınızda, genel olarak tercih edilir.

Uygulama yönetimli yeniden bağlantının, çalıştırma zamanı hatalarının işlenmesi için genel bir çözüm sağlanmadığını ve uygulamanın bağlantı hatasıyla ilgili olmayan hataları işlemeye devam etmesi gerektiğini unutmayın.



Örneğin, tam (2053 MQRC\_Q\_FULL) bir kuyruğa ileti yerleştirmeye ya da geçerli olmayan güvenlik kimlik bilgilerini kullanarak bir kuyruk yöneticisine bağlanma girişiminde bulunuluyor (2035 MQRC\_NOT\_YETKILI).

Başarısız olduğunda hiçbir eşgörünüm hemen kullanılabilir olmadığında, uygulamanın 2059 MQRC\_Q\_MGR\_NOT\_AVAM hatalarıyla da ilgilenmesi gerekir. Bu, uygulamanın JMS kural dışı durumlarını, sessiz bir şekilde yeniden bağlanmayı deneyerek bildirerek raporlayarak elde edilebilir.

#### *IBM MQ classes for JMS nesne havuzlama*

Using a form of connection pooling outside of Java EE helps to reduce overall load resulting, for example, from some stand-alone applications using frameworks, or being deployed into cloud environments, and also from a greater number of client connections into QueueManagers leading to an increase in server consolidation of applications and queue managers

Java EE programlama modeli içinde, kullanımda olan çeşitli nesnelerin iyi tanımlanmış bir yaşam çevrimi vardır. Servlet daha fazla özgürlük sağlarken, ileti odaklı Bean 'ler (MDBS ' ler) en çok kısıtlanmış durumda. Bu nedenle, Java EE sunucuları içinde yer alan havuzlama seçenekleri, kullanılan çeşitli programlama modellerine uygun olur.

Java SE ile (ya da Spring gibi başka bir çerçeveye) programlama modelleri son derece esnekliktir. Bu nedenle tek bir havuzlama stratejisi hiç de uygun değil. Eğer bir çerçeveye gidilecek bir çerçeve varsa bunu göz önünde bulundurmanız gerekir, örneğin, ilkbaharda.

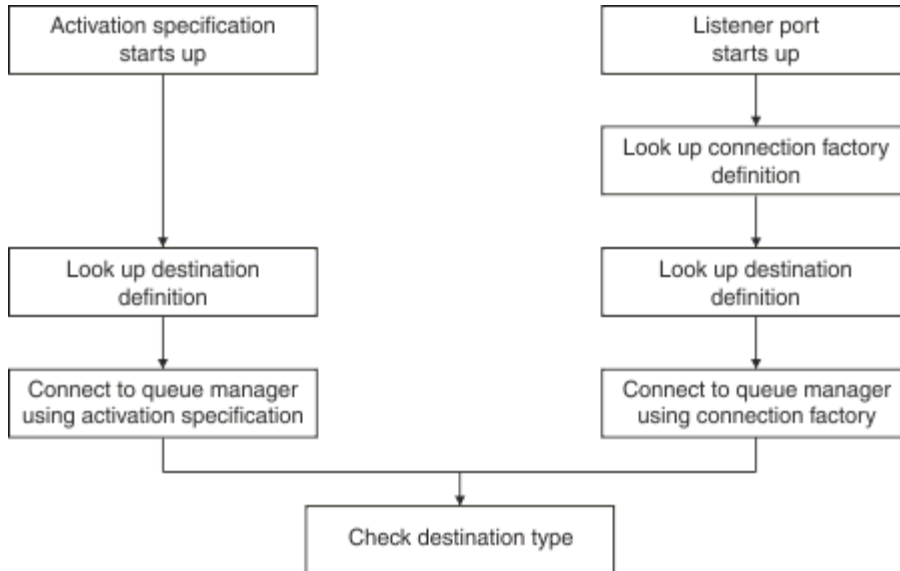
Kullanılacak havuzlama stratejisi, uygulamanızın çalışmakta olduğu ortama bağlıdır.

#### *Java EE ortamında nesne havuzlama*

Java EE uygulama sunucuları, ileti odaklı bean uygulamaları, Enterprise Java Beans and Servlet 'ler tarafından kullanılabilir bağlantı havuzlama işlevselliği sağlar.

WebSphere Application Server , performansı artırmak için bir JMS sağlayıcısıyla bağlantı havuzu sağlar. Bir uygulama JMS bağlantısı oluşturduğunda, uygulama sunucusu, boş bağlantı havuzunda bir bağlantının zaten var olup olmadığını belirler. Böyle bir durumda, bağlantı uygulamaya döndürülür; tersi durumda, yeni bir bağlantı yaratılır.

Şekil 47 sayfa 281 , hem etkinleştirme belirtilerinin, hem de dinleyici kapılarının bir JMS bağlantısı kurduğunu ve bu bağlantıyı, normal kipte iletiler için bir hedef izlemek üzere nasıl kullanacağını gösterir.



Şekil 47. Normal kip

IBM MQ ileti alışverişi sağlayıcısını kullanırken, giden ileti alışverişi gerçekleştiren uygulamalar (kurumsal Java Bean 'ler ve sunucu uygulamaları gibi) ve ileti odaklı Bean dinleyici kapısı bileşeni bu bağlantı havuzlarını kullanabilir.

IBM MQ ileti alışverişi sağlayıcısı etkinleştirme belirtileri, IBM MQ kaynak bağdaştırıcısı tarafından sağlanan bağlantı havuzlama işlevini kullanır. Ek bilgi için [WebSphere MQ kaynak bağdaştırıcısı için özelliklerin yapılandırılması](#) başlıklı konuya bakın.

“Bağlantı havuzunun kullanılmasına ilişkin örnekler” sayfa 285 explains how applications that perform outbound messaging, and listener ports, use the free pool when creating JMS connections.

“Serbest bağlantı havuzu bakımı iş parçacıkları” sayfa 288 , bir uygulama ya da dinleyici kapısı bağlantılarıyla bitirildiğinde bu bağlantılara ne olacağını açıklar.

“Havuz bakımı iş parçacığı örnekleri” sayfa 290 , JMS bağlantılarının eski haline dönüşmesini önlemek için boş bağlantı havuzunun nasıl temizleneceğini açıklar.

WebSphere Application Server , Bağlantı Üreticisi 'nin *bağlantı sayısı üst sınırı* özelliği tarafından belirtilen bir üreticiden yaratılabilecek bağlantı sayısı için bir sınır içerir. Bu özelliğin varsayılan değeri 10 'dur. Bu, bir fabrikadan herhangi bir zamanda en fazla 10 bağlantı yaratılabilir anlamına gelir.

Her üreticinin ilişkili bir serbest bağlantı havuzu vardır. Uygulama sunucusu başlatıldığında, serbest bağlantı havuzları boş olur. Bir fabrikaya ilişkin serbest havuzda bulunabilecek bağlantı sayısı üst sınırı, bağlantı sayısı üst sınırı özelliği tarafından da belirtilir.

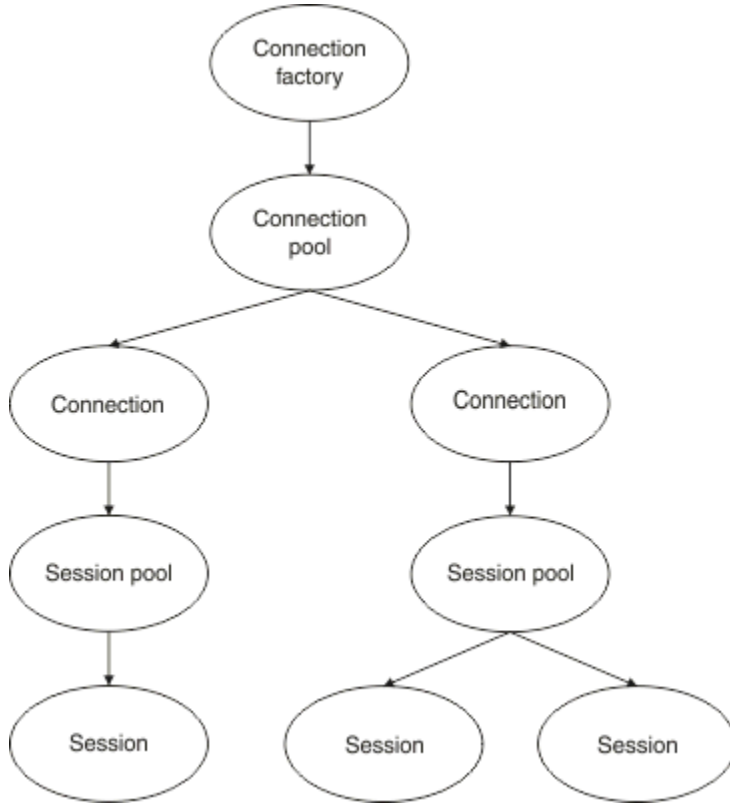
**İpucu:** JMS 2.0 ile, hem bağlantılar hem de bağlamlar yaratmak için bir bağlantı üreticisi kullanılabilir. Sonuç olarak, hem bağlantı hem de bağlamların karışımının bulunduğu bir bağlantı üreticisiyle ilişkilendirilmiş bir bağlantı havuzu olması mümkündür. Bağlantı üreticinin yalnızca bağlantı yaratmak ya da bağlamlar yaratmak için kullanılması önerilir. Bu, bağlantı havuzuna ilişkin bağlantı havuzunun yalnızca tek bir tip nesnelere içermesini sağlar; bu da havuzu daha verimli yapar.

Bağlantı havuzlama özelliğinin WebSphere Application Server' ta nasıl çalıştığı hakkında bilgi için bkz. [JMS bağlantıları için bağlantı havuzlamanın yapılandırılması](#). Diğer uygulama sunucuları için uygun uygulama sunucusu belgelerine bakın.

## **Bağlantı havuzunun nasıl kullanıldığı**

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 48 sayfa 283](#) , bu nesnelere arasındaki ilişkiyi gösterir.



Şekil 48. Bağlantı havuzları ve oturum havuzları

Bir dinleyici kapısı başlatıldığında ya da giden ileti alışverişi yapmak isteyen bir uygulama, bağlantı yaratmak için fabrikayı kullanır; 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 fabrika için boş havuzdan bağlantı almayı ve uygulamayı uygulamaya geri döndürmeyi dener.

Havuzda serbest bağlantı yoksa ve bu üreticiden yaratılan bağlantıların sayısı, o fabrikanın *bağlantı sayısı üst sınırı* özelliğinde belirtilen sınıra ulaşmadıysa, Connection Manager, uygulamanın kullanması için yeni bir bağlantı yaratır.

Ancak, bir uygulama bağlantı yaratma girişiminde bulunursa, ancak bu üreticiden yaratılan bağlantıların sayısı, fabrikanın *bağlantı sayısı üst sınırı* özelliğine eşitse, uygulama bir bağlantının kullanılabilir olmasını bekler (serbest havuza geri konmak için).

Uygulamanın bekleyeceği süre, varsayılan değeri 180 saniye olan bağlantı havuzunun *bağlantı zamanaşımı* özelliğinde belirtilir. Bu 180 saniyelik süre içinde bir bağlantı serbest havuza geri konursa, Connection Manager hemen havuzdan dışarı alır ve uygulamaya iletir. Ancak, zamanaşımı süresi aşılsa, bir *ConnectionWaitTimeoutException* yayınlanır.

Bir uygulama bağlantıyı bitirdiğinde ve şu çağrıya göre kapatır:

- **Connection.close()**
- **QueueConnection.close()**
- **TopicConnection.close()**

bağlantı aslında açık tutulmaktadır ve başka bir uygulama tarafından yeniden kullanılabilmesi için ücretsiz havuza geri dönmektedir. Bu nedenle, uygulama sunucusunda JMS uygulaması çalışmasa da, WebSphere Application Server ile JMS sağlayıcısı arasında bağlantı açık olabilir.

#### *Gelişmiş bağlantı havuzu özellikleri*

JMS bağlantı havuzlarının davranışını denetleyebilmek için kullanılacak gelişmiş özellikler vardır.

### **Dalgalanma koruması**

“Giden ileti sistemini gerçekleştiren uygulamaların bağlantı havuzunu nasıl kullanacağını” sayfa 287 , `connectionFactory.createConnection()` 'yi birleştiren `sendMessage()` yönteminin kullanımını açıklar.

Consider the situation where you have 50 EJBs all creating JMS connections from the same connection factory as part of their `ejbCreate()` method.

Bu çekirdeklerin tümü aynı anda oluşturulursa ve fabrikanın serbest bağlantı havuzunda herhangi bir bağlantı yoksa, uygulama sunucusu aynı JMS sağlayıcısına aynı anda 50 JMS bağlantısı yaratmayı dener. Sonuç, hem WebSphere Application Server hem de JMS sağlayıcısında önemli bir yüküdür.

Dalgalanma koruma özellikleri, bir bağlantı üreticisinden herhangi bir zamanda yaratılabilecek JMS bağlantılarının sayısını sınırlayarak ve ek bağlantı yaratılmasının sarsılmasını önleyerek bu durumu önleyebilir.

İki özellik kullanılarak, herhangi bir zamanda JMS bağlantı sayısının sınırlanması elde edilir:

- Dalgalanma eşiği
- Dalgalanma yaratma aralığı.

EJB uygulamaları bir bağlantı üreticisinden JMS bağlantısı yaratmayı denediğinde, bağlantı yöneticisi kaç bağlantının yaratıldığını görmek için denetler. Bu sayı, `surge threshold` özelliğinin değerinden küçükse ya da bu değere eşitse, bağlantı yöneticisi yeni bağlantıları açmaya devam eder.

Ancak, yaratılmakta olan bağlantı sayısı `surge threshold` özelliğini aşarsa, bağlantı yöneticisi yeni bir bağlantı yaratmadan ve açmadan önce `surge creation interval` özelliği tarafından belirtilen süre boyunca bekler.

### **Sıkışmış bağlantılar**

A JMS connection is considered `stuck`, if a JMS application uses that connection to send a request to the JMS provider, and the provider does not respond within a certain amount of time.

WebSphere Application Server , bu işlevi kullanmak için `stuck` JMS bağlantılarını algılamak için bir yol sağlar, üç özellik ayarlamamız gerekir:

- Sıkışan Zaman Zamanlayıcısı
- Sıkışan Süre
- Sıkışmış Eşik

“Havuz bakımı iş parçacığı örnekleri” sayfa 290 , havuz bakımı iş parçacığının belirli aralıklarla nasıl çalıştığını ve bir bağlantı fabrikasının boş havuzunun içeriğini denetleyerek, ya bir süre kullanılmayan ya da uzun süredir var olan bağlantıları nasıl aradığını denetler.

Sıkışmış bağlantıları saptamak için, uygulama sunucusu, bir bağlantı üreticisinden yaratılan tüm etkin bağlantıların durumunu denetleyen bir bağlantı iş parçacığının JMS sağlayıcısından yanıt bekliyor olup olmadığını denetleyerek de yönetir.

Sıkışan bağlantı iş parçacığı çalıştığında, `Stuck time timer` özelliği tarafından belirlenir. Bu özelliğin varsayılan değeri sıfırdır; yani, sıkışmış bağlantı algılaması hiçbir zaman çalıştırılır.

İş parçacığı yanıt bekleyen bir öge bulursa, ne kadar süre beklediğini belirler ve bu süreyi `Stuck time` özelliğinin değeriyle karşılaştırır.

JMS sağlayıcısının yanıt vermesi için alınan süre, `Stuck time` özelliğinin belirlediği süreyi aşıyorsa, uygulama sunucusu JMS bağlantısını sıkışmış olarak işaretler.

Örneğin, `jms/CF1` bağlantı üreticinin `Stuck time timer` özelliği 10 olarak ayarlansa ve `Stuck time` özelliği 15 değerine ayarlansa da.

Sıkışan bağlantı iş parçacığı her 10 saniyede bir etkin duruma gelir ve `jms/CF1` 'tan oluşturulan herhangi bir bağlantının IBM MQ' den gelen bir yanıt için 15 saniyeden daha uzun bir süre beklediğinden emin olur.

Suppose an EJB creates a JMS connection to IBM MQ using `jms/CF1`, and then tries to create a JMS Session using that connection by calling `Connection.createSession()`.

Ancak, bir şey JMS sağlayıcısının bu isteğe yanıt vermesini engelliyor. Makinenin dondurulmuş olması ya da JMS sağlayıcısında çalışan bir işlemin kilitlemesi, yeni çalışmaların işlenmesini engellemesidir:

Ten seconds after the EJB called `Connection.createSession()`, the stuck connection timer becomes active, and looks at the active connections created from `jms/CF1`.

Yalnızca bir etkin bağlantı olduğunu varsayın (örneğin, `c1`). The first EJB has been waiting 10 seconds for a response to a request it sent down to `c1`, which is less than the value of `Stuck time`, so the stuck connection timer ignores this connection and becomes inactive.

10 saniye sonra, sıkışmış bağlantı iş parçacığı yeniden etkin duruma gelir ve `jms/CF1` ile ilgili etkin bağlantılara bakar. Daha önce olduğu gibi, yalnızca tek bir bağlantı olduğunu varsayın: `c1`.

`createSession()` adlı ilk EJB 'ninsinceadlı ilk EJB' den bu yana 20 saniye olduğunu ve EJB hala yanıt bekliyor olduğunu belirtir. 20 saniye, `Stuck time` özelliğinde belirtilen süreden daha uzun, bu nedenle sıkışan bağlantı iş parçacığı `c1` 'yi sıkışmış olarak işaretler.

5 saniye sonra, IBM MQ son olarak yanıt verir ve ilk EJB 'nin bir JMS oturumu yaratmasına izin veriyorsa, bağlantı kullanımda olur.

Uygulama sunucusu, sıkışmış bir bağlantı üreticisinden yaratılan JMS bağlantılarının sayısını sayar. Bir uygulama yeni bir JMS Connection oluşturmak için o bağlantı üreticisini kullandığında ve o fabrikanın serbest havuzunda herhangi bir serbest bağlantı yoksa, bağlantı yöneticisi, sıkışan bağlantı sayısını `Stuck threshold` özelliğinin değeriyle karşılaştırır.

Sıkışan bağlantı sayısı, `Stuck threshold`, özellik için ayarlanan değerden küçükse, bağlantı yöneticisi yeni bir bağlantı oluşturur ve bunu uygulamaya verir.

Ancak, sıkışan bağlantı sayısı `Stuck threshold` özelliğinin değerine eşitse, uygulama bir kaynak kural dışı durumu alır.

## Havuz bölümleri

WebSphere Application Server , bir bağlantı üreticisine ilişkin serbest bağlantı havuzunu bölümlenize olanak sağlayan iki özellik sağlar:

- `Number of free pool partitions`, uygulama sunucusuna, serbest bağlantı havuzunu bölmek istediğiniz bölümlerin sayısını belirtir.
- `Free pool distribution table size`, bölümlerin nasıl dizinleneceğini belirler.

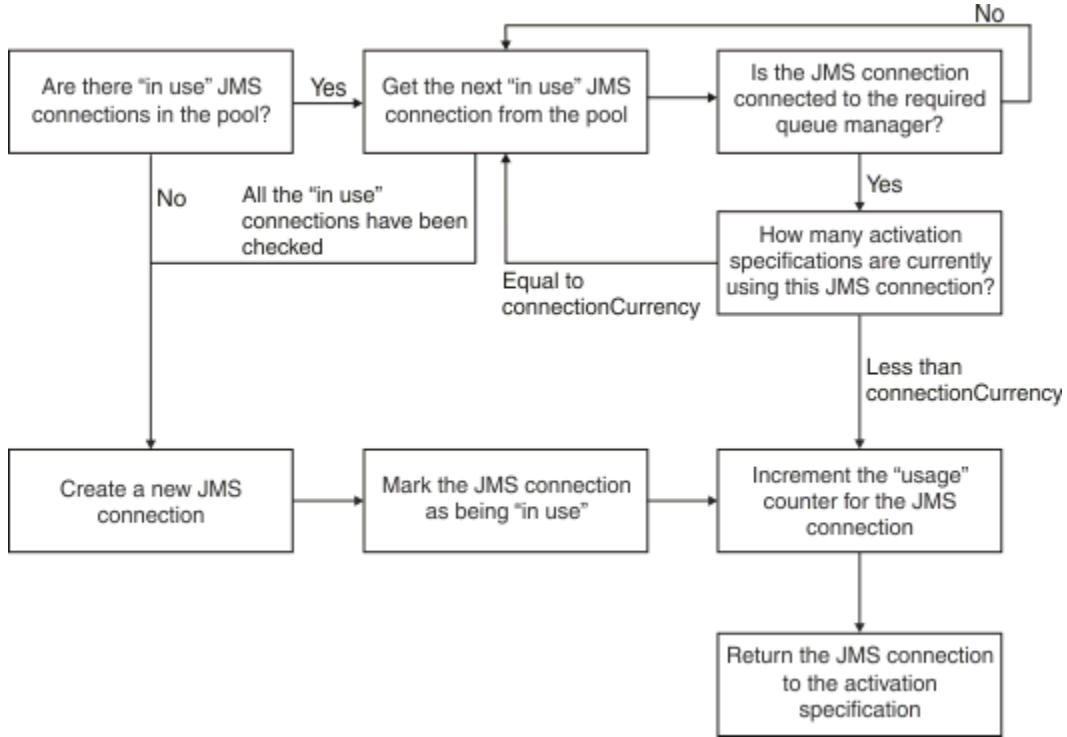
Bu özellikleri IBM Destek Merkezi 'nde değiştirmeniz istenmedikçe, bu özellikleri varsayılan olarak sıfır olan değerleri olarak bırakın.

WebSphere Application Server 'in `Number of shared partitions` adlı bir ek gelişmiş bağlantı havuzu özelliğine sahip olduğunu unutmayın. Bu özellik, paylaşılan bağlantıları saklamak için kullanılan bölümlerin sayısını belirtir. Ancak, JMS bağlantıları her zaman paylaşılmaz olarak kaldırıldığı için, bu özellik geçerli değildir.

### *Bağlantı havuzunun kullanılmasına ilişkin örnekler*

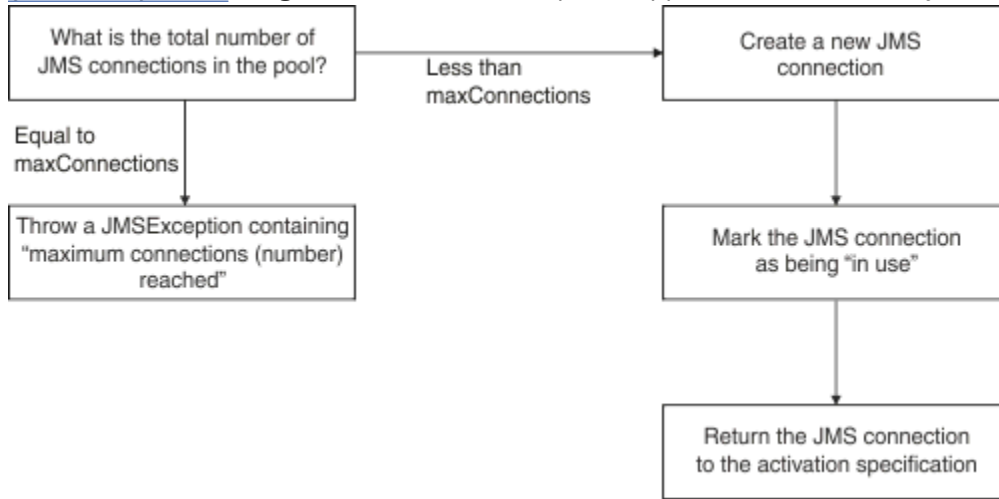
İletiyi yönlendirilen Bean dinleyici kapısı bileşeni ve giden ileti sistemini gerçekleştiren uygulamalar, bir JMS bağlantı havuzu kullanır.

Şekil 49 sayfa 286 , bağlantı havuzunun WebSphere Application Server V7.5 ve V8.0 ürünleri için nasıl çalıştığını gösterir.



Şekil 49. WebSphere Application Server V7.5 ve V8.0 -bağlantı havuzunun nasıl çalıştığı

Şekil 50 sayfa 286 , bağlantı havuzunun WebSphere Application Server V8.5 için nasıl çalıştığını gösterir.



Şekil 50. WebSphere Application Server V8.5 -bağlantı havuzunun nasıl çalıştığı

## MDB dinleyici kapıları bağlantı havuzunu nasıl kullanır?

JMS sağlayıcısı olarak IBM MQ kullanan bir WebSphere Application Server Network Deployment sisteminde konuşlandırılmış bir MDB 'nin olduğunu varsayın. MDB bir dinleyici kapısına (örneğin, jms/CF1, bağlantı sayısı üst sınırı ) 2 değerine ayarlanmış bir bağlantı üreticisi adı verilen bir dinleyici kapısına konuşlandırılır; bu, bu üreticiden herhangi bir zamanda yalnızca iki bağlantının yaratılabileceği anlamına gelir.

Dinleyici kapısı başlatıldığında, kapıjms/CF1 bağlantı üreticisini kullanarak IBM MQ' e bağlantı yaratmayı dener.

Bunu yapmak için kapı, bağlantı yöneticisinden bir bağlantı ister. Bu, jms/CF1 bağlantı üreticisi ilk kez kullanıldığı için, jms/CF1 boş bağlantı havuzunda bağlantı yoktur; bu nedenle, bağlantı yöneticisi çağrılan yeni bir bağlantı yöneticisi (örneğin, c1) yaratılır. Bu bağlantının, dinleyici kapısının tüm yaşam alanı için var olduğunu unutmayın.

Now, consider the situation where you stop the listener port using the WebSphere Application Server administrative console. Bu durumda, bağlantı yöneticisi bağlantıyı alır ve serbest havuza geri koyar. Ancak, IBM MQ ile bağlantı açık kalır.

Dinleyici kapısını yeniden başlatırsanız, kapı, kuyruk yöneticisiyle bağlantı için bağlantı yöneticisine bir kez daha sorar. Artık serbest havuzda bir bağlantının (c1) olduğundan, bağlantı yöneticisi bu bağlantıyı havuzdan alır ve dinleyici kapısı tarafından kullanılabilir duruma getirir.

Şimdi, uygulama sunucusuna konuşlandırılmış ikinci bir MDB 'nin olduğunu ve farklı bir dinleyici kapısı kullandığını varsayın.

Daha sonra, jms/CF1 bağlantı üreticisini kullanmak için de yapılandırılmış bir üçüncü dinleyici kapısı başlatmayı dener. The third listener port requests a connection from the connection manager, which looks in the free pool for jms/CF1 and finds that it is empty. Daha sonra jms/CF1 fabrikasından kaç bağlantının yaratıldığını denetler.

jms/CF1 için bağlantı sayısı üst sınırı 2 olarak ayarlandığından ve bu üreticiden iki bağlantı yaratmış olmanıza bağlı olarak, bağlantı yöneticisi bir bağlantının kullanılabilir olması için 180 saniye (bağlantı zamanaşımı özelliğinin varsayılan değeri) için bekler.

However, if you stop the first listener port, its connection c1 is put into the free pool for jms/CF1. Bağlantı yöneticisi bu bağlantıyı alır ve üçüncü dinleyiciye verir.

Şimdi ilk dinleyiciyi yeniden başlatmayı denerseniz, bu dinleyici, ilk dinleyici yeniden başlatılmadan önce diğer dinleyici kapılarından birinin durmasını beklemesi gerekir. Çalışmakta olan dinleyici kapılarının hiçbiri 180 saniye içinde durdurulmıyorsa, ilk dinleyici bir ConnectionWaitTimeoutException hatası alır ve durur.

## **Giden ileti sistemini gerçekleştiren uygulamaların bağlantı havuzunu nasıl kullanacağını**

Bu seçenek için, uygulama sunucusunda kurulu tek bir EJB (örneğin, EJB1) olduğunu varsayın. Bean, aşağıdakine göre sendMessage() adlı bir yöntem uygular:

- Creating a JMS connection to IBM MQ from a factory jms/CF1, using `connectionFactory.createConnection()`.
- Bağlantıdan bir JMS oturumu yaratılması.
- Oturumdan ileti üreticisi yaratılıyor.
- Mesaj yolluyorum.
- Üreticiyi kapatıyorum.
- Oturum kapatılıyor.
- `connection.close()` çağrılarak bağlantıyı kapatma.

Assume that the free pool for the factory jms/CF1 is empty. When the EJB is invoked for the first time, the bean attempts to create a connection to IBM MQ from the factory jms/CF1. Üretici için boş havuz boş olduğu için, bağlantı yöneticisi yeni bir bağlantı yaratır ve EJB1'e verir.

Just before the method exits, the method calls `connection.close()`. Rather than closing c1, the connection manager takes the connection and puts it into the free pool for jms/CF1.

Bir sonraki sendMessage() saati çağrıldığında, `connectionFactory.createConnection()` yöntemi uygulamaya c1 değerini döndürür.

EJB 'nin ikinci bir eşgörünümünün, ilk yönetim ortağıyla aynı anda çalıştığını varsayın. Her iki eşgörünüm de sendMessage() çağrıldığında, jms/CF1 bağlantı üreticisinden iki bağlantı oluşturulur.

Şimdi, Bean 'in üçüncü bir örneğinin yaratıldığını varsayın. Üçüncü bean `sendMessage()` 'i çağırdığında, yöntem `connectionFactory.createConnection()` ' den bir bağlantı yaratmak için `jms/CF1` ' i çağırır.

Ancak, şu anda `jms/CF1` ' tan yaratılmış iki bağlantı vardır; bu bağlantı, bu üreticiye ilişkin bağlantı sayısı üst sınırının değerini gösterir. Bu nedenle, `createConnection()` yöntemi, bir bağlantının kullanılabilir olması için 180 saniye (bağlantı zamanaşımı özelliğinin varsayılan değeri) bekler.

However, if the `sendMessage()` method for the first EJB calls `connection.close()` and exits, the connection it was using, `c1`, is put back into the free connection pool. Bağlantı yöneticisi bağlantıyı serbest havuzdan dışarı alır ve üçüncü EJB ' ye verir. Bu bean 'den `connectionFactory.createConnection()` ' ye yapılan çağrı ardından `sendMessage()` yönteminin tamamlanmasına olanak tanır.

## Aynı bağlantı havuzunu kullanan MDB dinleyici kapıları ve EJB ' ler

Önceki iki örnekte, dinleyici kapılarının ve EJB ' lerin yalıtılmış olarak bağlantı havuzunu nasıl kullanabilecekleri gösterilmektedir. Ancak, aynı uygulama sunucusu içinde çalışan bir dinleyici kapısı ve EJB, aynı bağlantı üreticisini kullanarak JMS bağlantıları yaratılabilir.

Bu durumun etkilerini göz önünde bulundurmanız gerekiyor.

Şunu anımsayacak anahtar nokta, bağlantı üreticisinin dinleyici kapısı ile EJB arasında paylaşıldığını anımsamaktadır.

Örneğin, aynı anda çalışan bir dinleyiciniz ve EJB ' nin olduğunu varsayalım. Her ikisi de `jms/CF1` bağlantı üreticisini kullanıyor; bu da, o fabrikaya ilişkin bağlantı büyüklüğü üst sınırına ulaşılan bağlantı sınırının aşıldığı anlamına gelir.

If you try to start either another listener port, or another instance of an EJB, either has to wait for a connection to be returned to the free connection pool for `jms/CF1`.

### *Serbest bağlantı havuzu bakımı iş parçacıkları*

Her bir serbest bağlantı havuzuyla ilişkilendirilmiş bir havuz bakım iş parçacığıysa, havuzdaki bağlantıların hala geçerli olduğundan emin olmak için boş havuzu izler.

Havuz bakım iş parçacığı, serbest havuzdaki bir bağlantının atılması gerektiğine karar verirse, iş parçacığı JMS bağlantısını fiziksel olarak IBM MQ' e kapatır.

## Havuz bakımı iş parçacığı nasıl çalışır

Havuz bakımı iş parçacığının davranışı, bağlantı havuzunun dört özelliği değeriyle belirlenir.

### **Yaşlanan zamanaşımı**

Bir bağlantının açık kaldığı süre.

### **En az bağlantı**

Bağlantı yöneticisinin bağlantı üreticisinin serbest havuzunda tuttuğu bağlantı sayısı alt sınırı.

### **Reap zamanı**

Havuz bakımı iş parçacığı ne sıklıkta çalıştırılır.

### **Kullanılmayan zamanaşımı**

Kapatılmadan önce, serbest havuzda bir bağlantı ne kadar süre kalır.

Varsayılan olarak, havuz bakımı iş parçacığı her 180 saniyede bir çalıştırılır, ancak bu değer, bağlantı havuzu **Reap time** özelliği ayarlanarak değiştirilebilir.

Bakım iş parçacığı, havuzdaki her bir bağlantıya bakar, havuzda ne kadar süre geçtiğini ve oluşturulduğundan ve son kullanılandan bu yana geçen sürenin ne kadar süre geçtiğini denetler.

Bağlantı, bağlantı havuzuna ilişkin **Unused timeout** özelliğinin değerinden daha uzun bir süre için kullanılmıyorsa, bakım iş parçacığı, serbest havuzdaki bağlantı sayısını denetler. Bu sayı ise:

- Greater than the value of **Minimum connections**, the connection manager closes the connection.
- **Minimum connections** değerini eşitler, bağlantı kapatılmaz ve serbest havuzda kalır.



**Minimum connections** özelliğinin varsayılan değeri 1'dir. Bu, performans nedenleriyle, bağlantı yöneticisi her zaman serbest havuzda en az bir bağlantıyı tutmaya çalışır.

**Unused timeout** özelliği, 1800 saniye varsayılan değerine sahiptir. Varsayılan olarak, bir bağlantı serbest havuza geri konursa ve en az 1800 saniye boyunca yeniden kullanılmazsa, o bağlantı kapatıldığında, serbest havuzda en az bir bağlantı bırakır.

Bu yordam, kullanılmayan bağlantıların eski haline gelmesini önler. Bu özelliği kapatmak için **Unused timeout** özelliğini sıfır olarak ayarlayın.

Bir bağlantı serbest havuzdaysa ve yaratılmasından bu yana geçen süre, bağlantı havuzuna ilişkin **Aged timeout** özelliğinin değerinden büyükse, en son kullanılmasından bu yana ne kadar süre geçtiğinden bağımsız olarak kapatılır.

Varsayılan olarak, **Aged timeout** özelliği sıfır olarak ayarlanır; yani, bakım iş parçacığı bu denetimi hiçbir zaman gerçekleştirmez. Connections that have been around for longer than the **Aged timeout** property are discarded regardless of how many connections will remain in the free pool. **Minimum connections** özelliğinin bu durumda herhangi bir etkisi olmadığını unutmayın.

## Havuz bakımı iş parçacığının devre dışı bırakılması

Önceki açıklamadan, havuz bakımı iş parçacığında, özellikle bağlantı üreticisinin serbest havuzunda çok sayıda bağlantı varsa, etkin olduğunda büyük bir iş anlaşması yaptığını görebilirsiniz.

Örneğin, **Maximum connections** özelliği her bir üretici için 10 değerine ayarlansa, üç JMS bağlantı üreticisi olduğunu varsayalım. Her 180 saniyede bir, üç havuz bakımı iş parçacığı etkin duruma gelir ve her bir bağlantı üreticisi için sırasıyla serbest havuzları tarar. Serbest havuzların birçok bağlantısı varsa, bakım iş parçacıklarının yapması gereken çok iş vardır, bu da performansı önemli ölçüde etkileyebilir.

You can disable the pool maintenance thread for an individual free connection pool by setting its **Reap time** property to zero.

Bakım iş parçacığının devre dışı bırakılması, **Unused timeout** 'un geçse bile hiçbir zaman kapatılmaması anlamına gelir. Ancak, **Aged timeout** iletiliyse, bağlantılar yine de kapatılabilir.

Bir uygulama bağlantıyı bitirdiğinde, bağlantı yöneticisi bağlantının var olduğunu görmek için denetler ve o dönem **Aged timeout** özelliğinin değerinden daha uzunsa, bağlantı yöneticisi bağlantıyı serbest havuza döndürmek yerine, bağlantıyı kapatır.

## Aged zaman aşımını işleme etkileri

Önceki bölümde açıklandığı gibi, **Aged timeout** özelliği, bağlantı yöneticisi kapanmadan önce JMS sağlayıcısıyla bağlantının ne kadar süreyle açık kalacağını belirtir.

**Aged timeout** özelliğinin varsayılan değeri sıfır, bu da bağlantının çok eski olduğu için hiçbir zaman kapatılmayacağını belirtir. You should leave the **Aged timeout** property at this value, because enabling **Aged timeout** can have transactional implications when using JMS inside of EJBs.

JMS' ta, bir işlemin birimi, bir JMS *bağlantısından* oluşturulan bir JMS *oturumdur*. Bu, JMS *connectiondeğil*, hareketlerle ilgili olarak listelenen JMS *oturumdur*.

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.

Closing a JMS connection causes any outstanding transactional work on JMS sessions to be rolled back, as described in the JMS specification. Ancak, uygulama sunucusu, bağlantıdan yaratılan JMS oturumlarının artık geçerli olmadığını bilmiyor. Sunucu bir hareketi kesinleştirmek ya da geri almak için oturumu kullanmayı denediğinde `IllegalStateException` oluşur.

**Önemli:** If you want to use **Aged timeout** with JMS connections from within EJBs, ensure that any JMS work is explicitly committed on the JMS session, before the EJB method that performs the JMS operations exits.

### *Havuz bakımı iş parçacığı örnekleri*

Havuz bakımı iş parçacıklarının nasıl çalıştığını anlamak için Enterprise Java Bean (EJB) örneğini kullanma. Message Driven Beans (MDBs) ve dinleyici kapılarını da kullanabildiğinizi unutmayın; tek ihtiyacınız olan, serbest havuzdaki bağlantıları elde etmek için kullanabileceğiniz bir yöntem.

`sendMessage()` yöntemine ilişkin ek bilgi için bkz. [“Giden ileti sistemini gerçekleştiren uygulamaların bağlantı havuzunu nasıl kullanacağını” sayfa 287](#).

Bağlantı üreticisini aşağıdaki değerlerle yapılandırdınız:

- **Reap time** varsayılan değer olan 180 saniye
- **Aged timeout** varsayılan değer olarak sıfır saniye değerini
- **Unused timeout**, 300 saniye olarak ayarlandı

Uygulama sunucusu başlatıldıktan sonra, `sendMessage()` yöntemi çağrılır.

The method creates a connection called, for example `c1`, using the factory `jms/CF1`, uses that factory to send a message, and then calls `connection.close()`, which causes `c1` to be put into the free pool.

180 saniye sonra havuz bakımı iş parçacığı başlatılır ve `jms/CF1` boş bağlantı havuzuna bakar. Havuzda serbest bağlantı `c1` vardır; bu nedenle, bakım iş parçacığı bağlantının geri konması sırasında görünür ve bunu yürürlükteki saatle karşılaştırır.

180 seconds have passed since the connection was put in the free pool, which is less than the value of the **Unused timeout** property for `jms/CF1`. Bu nedenle, bakım iş parçacığı bağlantıyı tek başına bırakır.

180 saniye sonra, havuz bakımı iş parçacığı yeniden çalışır. Bakım iş parçacığı `c1` bağlantısını bulur ve bağlantının, **Unused timeout** değer kümesinden daha uzun olan 360 saniye havuzunda bulunduğunu belirler; böylece bağlantı yöneticisi bağlantıyı kapatır.

Şimdi `sendMessage()` yöntemini yeniden çalıştırırsanız, uygulama `connectionFactory.createConnection()` 'i çağırdığında, bağlantı yöneticisi IBM MQ 'e yeni bir bağlantı yaratır; bağlantı üreticisine ilişkin serbest bağlantı havuzu boş olur.

The preceding example shown how the maintenance thread uses the **Reap time** and **Unused timeout** properties to prevent stale connections, when the **Aged timeout** property is set to zero.

### **Aged timeout** özelliği nasıl çalışır?

Aşağıdaki örnekte, aşağıdakini ayarladığınızı varsayın:

- **Aged timeout** özelliği 300 saniyeye
- **Unused timeout** özelliği sıfır değerine sahip.

`sendMessage()` yöntemini çağırıyorsunuz ve bu yöntem `jms/CF1` bağlantı üreticisinden bir bağlantı oluşturmaya çalışır.

Bu fabrikaya ilişkin serbest havuz boş olduğundan, bağlantı yöneticisi yeni bir bağlantı oluşturur (`c1`) ve uygulamayı uygulamaya döndürür. `sendMessage()` `connection.close()` çağrıldığında, `c1` serbest bağlantı havuzuna geri konmaktadır.

180 saniye sonra, havuz bakımı iş parçacığı çalışır. İş parçacığı, serbest bağlantı havuzunda `c1` seçeneğini bulur ve uzun süre önce ne kadar süre önce oluşturulduca denetler. The connection has existed for 180 seconds, which is less than **Aged timeout**, so the pool maintenance thread leaves it alone and goes back to sleep.

60 saniye sonra, `sendMessage()` yeniden çağrılır. This time, when the method calls `connectionFactory.createConnection()`, the connection manager discovers that there is a connection, `c1`, available in the free pool for `jms/CF1`. Bağlantı yöneticisi, `c1` 'u serbest havuzdan alır ve uygulamaya bu bağlantıyı verir.

The connection is returned to the free pool when `sendMessage()` exits. 120 seconds later, the pool maintenance thread wakes up again, scans the contents of the free pool for `jms/CF1` and discovers `c1`.

Although the connection was only used 120 seconds ago, the pool maintenance thread closes the connection, because the connection has been in existence for a total of 360 seconds, which is longer than the 300 second value you set for the **Aged timeout** property.

## Bağlantı alt sınırı özelliğinin havuz bakımı iş parçacığını nasıl etkilediği

“MDB dinleyici kapıları bağlantı havuzunu nasıl kullanır?” sayfa 286 örneğini yeniden kullanarak, her biri farklı bir dinleyici kapısını kullanarak uygulama sunucunuzda konuşlandırılmış iki MDBS ' nin olduğunu varsayın.

Her dinleyici kapısı, şu adresi kullanarak yapılandırduğunuz 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

Suppose that the first listener is stopped, and its connection c1 is put into the free pool. 180 seconds later, the pool maintenance thread wakes up, scans the contents of the free pool for jms/CF1, and discovers that c1 has been in the free pool for longer than the value of the **Unused timeout** property for the connection factory.

Ancak, c1 kapatılmadan önce, havuz bakım iş parçacığı, bu bağlantı atılırsa havuzda kaç bağlantının kalacağını görecektir. Since c1 is the only connection in the free connection pool, the connection manager does not close it, because doing so would make the number of connections that remain in the free pool less than the value set for **Minimum connections**.

Şimdi, ikinci dinleyicinin durduğunu varsayın. Serbest bağlantı havuzu şu anda iki ücretsiz bağlantı içeriyor: c1 ve c2.

180 saniye sonra, havuz bakımı iş parçacığı yeniden çalışır. Bu zamana kadar c1 , serbest bağlantı havuzunda 360 saniye, c2 ise 180 saniye boyunca serbest bedir.

The pool maintenance thread checks c1 and discovers that it has been in the pool for longer than the value of the **Unused timeout** property.

İş parçacığı, serbest havuzda kaç bağlantının olduğunu görmek için denetler ve bu değeri **Minimum connections** özelliğinin değeriyle karşılaştırır. Havuz iki bağlantı içerdiğinden ve **Minimum connections** 1 olarak ayarlandığından, bağlantı yöneticisi c1' i kapatır.

Bakım iş parçacığı şimdi c2 konumunda görünüyor. Bu, **Unused timeout** özelliğinin değerinden daha uzun bir süre için serbest bağlantı havuzunda da yer aldı. However, since closing c2 would leave the free connection pool with less than the set number of Minimum connections in it, the connection manager leaves c2 alone.

### *JMS bağlantıları ve IBM MQ*

Information on the use of IBM MQ as the JMS provider.

## Bağ tanımlama aktarımlarının kullanılması

Bir bağlantı üreticisi bağ tanımlama iletimi kullanacak şekilde yapılandırıldıysa, her JMS bağlantısı, IBM MQ ile bir etkileşim ( **hconn** olarak da bilinir) oluşturur. Etkileşim, kuyruk yöneticisiyle iletişim kurmak için işlemler arası iletişim (ya da paylaşılan bellek) kullanır.

## İstemci taşımanın kullanılması

Bir IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticisi istemci aktarımı kullanacak şekilde yapılandırıldığında, bu üreticiden yaratılan her bağlantı, IBM MQ olarak da bilinen yeni bir etkileşim (**hconn** olarak da bilinir) kurar.

Bir kuyruk yöneticisine IBM MQ ileti alışverişi sağlayıcısı olağan kipini kullanarak bağlanan bağlantı fabrikaları için, bağlantı üreticisinden yaratılan birden çok JMS bağlantısının bir TCP/IP bağlantısını

IBM MQ ile paylaşılması olanaklıdır. Daha fazla bilgi için bkz. [“IBM MQ classes for JMS içinde bir TCP/IP bağlantısının paylaşılması” sayfa 294.](#)

Herhangi bir zamanda JMS bağlantıları tarafından kullanılan istemci kanalı sayısı üst sınırını saptamak için, aynı kuyruk yöneticisini gösteren tüm bağlantı fabrikaları için *Bağlantı sayısı üst sınırı* özelliğinin değerini ekleyin.

Örneğin, aynı IBM MQ kanalını kullanarak aynı IBM MQ kuyruk yöneticisine bağlanmak üzere yapılandırılmış iki bağlantı fabrikasının ( *jms/CF1* ve *jms/CF2* ) olduğunu varsayalım.

Bu fabrikalar varsayılan bağlantı havuzu özelliklerini kullanıyor, bu da *Bağlantı sayısı üst sınırı* 10 olarak ayarlanıyor anlamına gelir. Bağlantıların tümü hem *jms/CF1* hem de *jms/CF2* ' den aynı anda kullanılıyorsa, uygulama sunucusu ile IBM MQ arasında 20 etkileşim olacaktır.

If the connection factory connects to the queue manager using IBM MQ messaging provider normal mode, then the maximum number of TCP/IP connections that can exist between the application server and the queue manager for these connection factories is:

```
20/the value of SHARECNV for the IBM MQ channel
```

Bağlantı üreticisi, IBM MQ ile alışverişi sağlayıcısı geçiş kipini kullanarak bağlantı kurmak üzere yapılandırıldıysa, uygulama sunucusu ile bu bağlantı fabrikaları için IBM MQ arasında TCP/IP bağlantısı sayısı üst sınırı 20 olur (iki fabrika için bağlantı havuzlarındaki her JMS bağlantısı için bir bağlantı oluşturulacaktır).

### İlgili kavramlar

[“kullanmaIBM MQ classes for JMS” sayfa 77](#)

IBM MQ classes for Java Message Service (IBM MQ classes for JMS), IBM MQ ile birlikte verilen JMS sağlayıcısıdır. *javax.jms* paketinde tanımlanan arabirimlerin yanı sıra, IBM MQ classes for JMS , JMS API ' ye iki uzantı kümesi sağlar.

### Java SE ortamında nesne havuzlama

Java SE ile (ya da Spring gibi başka bir çerçeveye) programlama modelleri son derece esnekliktir. Bu nedenle tek bir havuzlama stratejisi hiç de uygun değil. Mesela, baharda herhangi bir pooling formu içeren bir çerçeve var mı göz önünde bulunmalısınız.

Aksi takdirde, uygulama mantığı bunu alabilir. Uygulamanın kendisinin ne kadar karmaşık olduğunu kendinize sorun. Uygulamanın ve bu uygulamanın ileti sistemi bağlantısından ne talep ettiği en iyi şekilde anlaşılmalıdır. Uygulamalar genellikle temel JMS API 'si çevresinde kendi sarıcı kodlarının içinde de yazılır.

Bu, çok mantıklı bir yaklaşım olabilir ve karmaşıklığı gizleyebilirken, sorunları ortaya çıkarması göz önünde bulundurulmaya değer. Örneğin, sık sık çağrılan genel bir *getMessage()* yöntemi, yalnızca tüketicileri açmamalı ve kapatmamalıdır.

Dikkate alınması gereken noktalar:

- Uygulamanın IBM MQ' e ne kadar süreyle erişmesi gerekiyor? Her zaman, ya da ara sıra.
- İletiler kaç kez gönderilecektir? Daha az sıklıkta, IBM MQ ' a yönelik tek bir bağlantı paylaşılabilir.
- Bağlantı kopan bir kural dışı durum genellikle havuzlanmış bir bağlantının yeniden yaratılması için gereken bir işarettir. Ne hakkında?
  - Güvenlik kural dışı durumları ya da anasistemi kullanılabilir değil
  - Kuyruk tam kural dışı durumları
- Bağlantı bozuk bir kural dışı durum oluşursa, havuzdaki diğer serbest bağlantılara ne olmalıdır? Kapatılıp yeniden yaratılmalı mı?
- TLS kullanılıyorsa, örneğin, tek bir bağlantının ne kadar süreyle açık kalmasını istiyorsunuz?
- Havuzlanmış bir bağlantı, bir kuyruk yöneticisi yöneticisinin bağlantıyı saptayabilmesi ve izini geri alabilmesi gibi kendisini nasıl tanımlayacak.

Havuzlama için tüm JMS nesnelerini göz önünde bulundurmanız ve o nesneyi ne zaman mümkün kıldığı zaman havuzu olarak göz önünde bulundurmanız gerekir. Nesnelere şunlardır:

- JMS bağlantılar
- Session
- Bağlamlar
- Tüm farklı tiplerde üreticiler ve tüketiciler

İstemci iletimi, JMS bağlantıları, oturumları ve bağlamları kullanıldığında, IBM MQ kuyruk yöneticisiyle iletişim kurulurken yuvalar kullanılır. Bu nesnelere havuza yönlendirilerek, tasarruf, gelen IBM MQ bağlantılarının kuyruk yöneticisine (hConns), kanal yönetim ortamlarının sayısında bir küçülmeye (incoming) ilişkin sayılardır.

Kuyruk yöneticisine bağ tanımlarının kullanılması, ağ oluşturma katmanını tümüyle kaldırır. Ancak, birçok uygulama daha yüksek düzeyde kullanılabilir ve iş yükü dengeli, yapılandırma sağlamak için istemci iletimi kullanır.

JMS producers and consumers open destinations on the queue manager. Daha az sayıda kuyruk ya da konu açılırsa ve uygulamanın birden çok bölümü bu nesnelere kullanıyorsa, bunları havuzlama işlemi yararlı olabilir.

Bir IBM MQ perspektifinden bu işlem bir MQOPER ve MQCLOSE işlemleri dizisini kaydeder.

## Bağlantılar, oturumlar ve bağlamlar

These objects all encapsulate IBM MQ connection handles to the queue manager, and are generated from a `ConnectionFactory`. Bir uygulamaya, bağlantı sayısını ve tek bir bağlantı üreticisinden belirli bir sayıya kadar oluşturulan diğer nesnelere sınırlandırmak için bir mantık ekleyebilirsiniz.

Oluşturulan bağlantıları içermesi için uygulamada basit bir veri yapısı kullanabilirsiniz. The application code that needs to use one of these data structures can *dışarı alma* an object to use.

Hesaba aşağıdaki etkenleri alın:

- Bağlantılar havuzdan ne zaman kaldırılmalıdır? Genellikle, bağlantıda bir kural dışı durum dinleyicisi yaratın. Bu dinleyici bir kural dışı durumu işlemek için çağrıldığında, bağlantıyı yeniden yaratmalı ve o bağlantıdan yaratılan oturumlar yeniden yaratılmalıdır.
- İş yükü dengelemesi için bir CCDT kullanılırsa, bağlantılar farklı kuyruk yöneticilerine gidebilir. Bu, havuzlama gereksinimleri için geçerli olabilir.

JMS belirtiminin, birden çok iş parçacığının bir oturuma ya da bağlama aynı anda erişmesi için bir programlama hatası olduğunu belirtmiş olduğunu unutmayın. IBM MQ JMS kodu, iş parçacıklarının işlenmesi sırasında özenli olmayı dener. Ancak, bir oturum ya da bağlam nesnesinin bir kerede tek bir iş parçacığı tarafından kullanılmasını sağlamak için, uygulamaya mantık eklemelisiniz.

## Üreticiler ve tüketiciler

Oluşturulan her üretici ve tüketici kuyruk yöneticisinde bir hedef açar. Aynı hedef, çeşitli görevler için kullanılacaksa, tüketici ya da üretici nesnelere açık tutulması mantıklı olur. Yalnızca, tüm iş yapılırsa nesneyi kapatın.

Bir hedef açılıp kapatılmasına kısa işlemler olsa da, bu işlemler sık sık yapılırsa, bunlar da eklenerek eklenir.

Bu nesnelere kapsamı, yaratıldığı oturum ya da bağlam içinde yer aldığından, bu nesnelere kapsamı içinde tutulmaları gerekir. Genel olarak, uygulamalar bu tür bir şekilde anlaşılır bir şekilde yazılır.

## İzleme

Uygulamalar nesne havuzlarını nasıl izleyecek? Bunun cevabı büyük ölçüde, uygulanan havuzlamanın çözümünün karmaşıklığı ile belirlenir.

Bir JavaEE havuzlama uygulamasını dikkate aldıysanız, aşağıdakiler de içinde olmak üzere çok sayıda seçenek vardır:

- Havuzların yürürlükteki büyüklüğü
- Nesnelerin içinde harcanan zaman nesnelere
- Havuzların temizlenmesi
- Bağlantıların yenilenmesi

Ayrıca, kuyruk yöneticinde tek bir yeniden kullanılan oturumun nasıl görüneceğini de göz önünde bulunmanız gerekir. Yararlı olabilecek uygulamayı ( appName gibi) tanıtmak için bağlantı üreticisi özellikleri vardır.

“kullanmaIBM MQ classes for JMS” sayfa 77

IBM MQ classes for Java Message Service (IBM MQ classes for JMS), IBM MQ ile birlikte verilen JMS sağlayıcısıdır. javax.jms paketinde tanımlanan arabirimlerin yanı sıra, IBM MQ classes for JMS , JMS API ' ye iki uzantı kümesi sağlar.

*IBM MQ classes for JMS içinde bir TCP/IP bağlantısının paylaşılması*

Tek bir TCP/IP bağlantısını paylaşmak için bir MQI kanalının birden çok eşgörenümü yapılabilir.

Applications that are running inside the same Java runtime environment, and that use the IBM MQ classes for JMS or the IBM MQ resource adapter to connect to a queue manager by using the CLIENT transport, can be made to share the same channel instance.

Kanal örnekleri ve TCP/IP bağlantıları arasında bire bir ilişki vardır. Her kanal yönetim ortamı için bir TCP/IP bağlantısı yaratılır.

Bir kanal **SHARECNV** parametresiyle tanımlandıysa, 1 değerinden büyük bir değere ayarlandıysa, bu sayıda etkileşim kanal yönetim ortamını paylaşabilir. Bir bağlantı üreticisini ya da etkinleştirme belirtimini bu işlevi kullanacak şekilde etkinleştirmek için, **SHARECNV** özelliğini YES(EVET) olarak ayarlayın.

Bir JMS uygulaması tarafından yaratılan her JMS bağlantısı ve JMS oturumu, kuyruk yöneticisiyle kendi iletişimisini yaratır.

Bir etkinleştirme belirtimi başlatıldığında, IBM MQ kaynak bağdaştırıcısı kullanılacak etkinleştirme belirtimine ilişkin kuyruk yöneticisiyle bir etkileşim başlatır. Sunucu oturumu havuzundaki etkinleştirme belirtimiyle ilişkilendirilmiş her sunucu oturumu, kuyruk yöneticisiyle de bir etkileşim başlatır.

SHARECNV özneliği, bağlantı paylaşımına yönelik en iyi bir çalışma yaklaşımıdır. Bu nedenle, IBM MQ classes for JMS ile 0 'dan büyük bir SHARECNV değeri kullanılırsa, yeni bir bağlantı isteğinin önceden kurulmuş bir bağlantıyı her zaman paylaşacağı garanti edilmez.

## Kanal örneklerinin sayısını hesaplama

Bir uygulama tarafından yaratılan kanal somut örnekleri sayısı üst sınırını belirlemek için aşağıdaki formül kullanın:

### Etkinleştirme belirtileri

Kanal örneklerinin sayısı =  $(maxPoolDepth\_value + 1) / SHARECNV\_value$

Burada *maxPoolDepth\_value* , **maxPoolDepth** özelliğinin değeri ve *SHARECNV\_value* , etkinleştirme belirtimi tarafından kullanılan kanaldaki **SHARECNV** özelliğinin değeridir.

### Diğer JMS uygulamaları

Kanal örneklerinin 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.

## Örnekler

The following examples show how to use the formulae to calculate the number of channel instances that are created on a queue manager by applications by using either the IBM MQ classes for JMS or the IBM MQ resource adapter.

## JMS uygulama örneği

Bir JMS uygulaması bağlantısı, CLIENT iletimi kullanılarak bir kuyruk yöneticisine bağlanır ve bir JMS bağlantısı ve üç JMS oturumu yaratır. The channel that the application is using to connect to the queue manager has the **SHARECNV** property set to the value of 10. Uygulama çalışırken, uygulama ile kuyruk yöneticisi ile bir kanal yönetim ortamı arasında dört etkileşim vardır. Dört konuşmayla tüm kanal yönetim ortamı paylaşılıyor.

**V 9.1.3** IBM MQ 9.1.3' tan, uygulamalar `reconnectable` olarak yapılandırıldıysa, kanal eşgörünümleri yalnızca ilgili JMS nesneleri arasında paylaşılabilir; bu bir JMS Connection ve ilgili JMS Oturumlarıdır. Bu, bu tür uygulamaları desteklemek için ek kanal örnekleri yapılandırılmasını gerektirebilir.

For example, if an application uses a single JMS Connection and a single JMS Session, and the channel used has **SHARECNV** equal to 10, before IBM MQ 9.1.3, up to five instances of the application could share a single channel instance. This is still the case for IBM MQ 9.1.3 or later if the application is not configured to be `reconnectable`, but if the application is configured as `reconnectable`, each application instance will require its own channel instance, so five channel instances in total would be needed.

## Etkinleştirme belirtimi örneği

Bir etkinleştirme belirtimi, CLIENT iletimi kullanılarak kuyruk yöneticisine bağlanır. Etkinleştirme belirtimi, **maxPoolDepth** özelliği 10 değerine ayarlanmış şekilde yapılandırılır. Etkinleştirme belirtiminin kullanmak üzere yapılandırıldığı kanalda **SHARECNV** özelliği 10 değerine ayarlanmış olmalıdır. Etkinleştirme belirtimi çalışırken 10 ileti eşzamanlı olarak işlenirken, etkinleştirme belirtimi ile kuyruk yöneticisi arasındaki etkileşim sayısı 11 (sunucu oturumları için 10 etkileşim ve etkinleştirme belirtimi için bir) olur. Etkinleştirme belirtimi tarafından kullanılan kanal eşgörünümlerinin sayısı 2 'dir.

## Etkinleştirme belirtimi örneği

Bir etkinleştirme belirtimi, CLIENT iletimi kullanılarak kuyruk yöneticisine bağlanır. Etkinleştirme belirtimi, **maxPoolDepth** özelliği 5 olarak ayarlanmış şekilde yapılandırılır. Etkinleştirme belirtiminin kullanmak üzere yapılandırıldığı kanalda **SHARECNV** özelliği 0 olarak ayarlanmış olmalıdır. Etkinleştirme belirtimi çalıştırılırken ve koşut zamanlı 5 ileti işlenirken, etkinleştirme belirtimi ile kuyruk yöneticisi arasındaki etkileşim sayısı 6 (sunucu oturumları için beş etkileşim ve etkinleştirme belirtimi için bir) olur. Kanalin üzerindeki **SHARECNV** özelliği 0 değerine ayarlandığından, etkinleştirme belirtimi tarafından kullanılan kanal eşgörünümlerinin sayısı 0 'tır, her etkileşim kendi kanal yönetim ortamını kullanır.

## İlgili görevler

[“WebSphere Application Server ile IBM MQ arasında oluşturulan TCP/IP bağlantılarının sayısı belirleniyor” sayfa 477](#)

Paylaşma etkileşimleri özelliğini kullanarak, birden çok etkileşim MQI kanalı yönetim ortamlarını paylaşabilir; bu, TCP/IP bağlantısı olarak da bilinir.

*IBM MQ classes for JMS içinde istemci bağlantıları için bir kapı aralığı belirtme*

Uygulamanızın bağlanabileceği bir kapı aralığı belirtmek için LOCALADDRESS özelliğini kullanın.

Bir IBM MQ classes for JMS uygulaması istemci kipinde bir IBM MQ kuyruk yöneticisine bağlanmayı denediğinde, bir güvenlik duvarı yalnızca belirtilen kapılardan ya da bir kapı aralığından kaynaklanan bağlantılara izin verebilir. Bu durumda, uygulamanın bağlanabileceği bir kapı ya da kapı aralığı belirtmek için bir 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. Burada, bir uygulama içinden özelliğin ayarlanmasını gösteren bir örnek vardır:

```
mqConnectionFactory.setLocalAddress("192.0.2.0(2000,3000)");
```

Uygulama daha sonra bir kuyruk yöneticisine bağlandığında, uygulama 192.0.2.0(2000)- 192.0.2.0(3000) aralığındaki yerel bir IP adresine ve kapı numarasına bağlanır.

Birden çok ağ arabirimine sahip bir sistemde, bir bağlantı için hangi ağ arabiriminin kullanılması gerektiğini belirlemek 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 sınırladığınızda bağlantı hataları ortaya çıkabilir. Bir hata oluşursa, MQRC\_Q\_MGR\_NOT\_AVAAD IBM MQ neden kodunu ve aşağıdaki iletiyi içeren yerleşik MQException ile bir JMSEException yayınlanır:

```
LOCAL_ADDRESS_PROPERTY kısıtlaması 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 gerekenlerden başka bağlantılar yaratabileceği için, her zaman bir kapı aralığı belirtmeyi düşünebilirsiniz. Genel olarak, bir uygulama tarafından oluşturulan her oturum için bir kapı ve IBM MQ classes for JMS üç ya da dört ek kapı gerektirebilir. Bir bağlantı hatası ortaya çıkarsa, kapı aralığını artırın.

IBM MQ classes for JMS' ta varsayılan olarak kullanılan bağlantı havuzlama, kapıların yeniden kullanılabilmesi hız üzerinde bir etkiye sahip olabilir. Sonuç olarak, bağlantı noktaları serbest bırakılırken bir bağlantı hatası ortaya çıkabilir.

#### *IBM MQ classes for JMS içinde kanal sıkıştırması*

Bir IBM MQ classes for JMS uygulaması, bir ileti üstbilgisini ya da verilerini sıkıştırmak için IBM MQ olanağını kullanabilir.

Bir IBM MQ kanalında akan verilerin sıkıştırılmasına ilişkin sıkıştırma, kanalın performansını artırabilir ve ağ trafiğini azaltabilir. IBM MQ ile verilen işlevi kullanarak, ileti kanallarında ve MQI kanallarında akan verileri sıkıştırabilirsiniz. Her iki tipteki bir kanalda, üstbilgi verilerini ve ileti verilerini birbirinden bağımsız olarak sıkıştırabilirsiniz. Varsayılan olarak, bir kanalda veri sıkıştırılmaz.

An IBM MQ classes for JMS application specifies the techniques that can be used for compressing header or message data on a connection by creating a java.util.Collection object. Her sıkıştırma tekniği, kaynak grubundaki bir Tamsayı nesnesidir ve uygulamanın kaynak grubuna sıkıştırma tekniklerini eklediği sıra, uygulama bağlantıyı yarattığında sıkıştırma tekniklerinin kuyruk yöneticisiyle anlaşmalı olduğu sıradır. The application can then pass the collection to a ConnectionFactory object by calling the setHdrCompList() method, for header data, or the setMsgCompList() method, for message data. Uygulama hazır olduğunda, bağlantı yaratabilir.

Aşağıdaki kod parçaları açıklanan yaklaşımı gösterir. İlk kod parçası, üstbilgi veri sıkıştırmasının nasıl gerçekleştirileceğini gösterir:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(WMQConstants.WMQ_COMPHDR_SYSTEM));
.
.
((MQConnectionFactory) cf).setHdrCompList(headerComp);
.
.
connection = cf.createConnection();
```

İkinci kod parçası, ileti veri sıkıştırmasının nasıl gerçekleştirileceğini 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 yaratıldığında sırasıyla RLE, daha sonra ZLIBLIHH olarak kararlaştırılır. Seçilen sıkıştırma tekniği, Connection nesnesinin geçerlik süresi boyunca değiştirilemez. Bir bağlantıda sıkıştırmayı kullanmak için, Connection nesnesi yaratılmadan önce setHdrCompList() ve setMsgCompList() yöntemlerinin çağrılması gerekir.

#### IBM MQ classes for JMS' da iletileri zamanuyumsuz olarak koyma

Olağan durumda, bir uygulama bir hedefe ileti gönderdiğinde, uygulamanın, kuyruk yöneticisinin isteği işlediğini doğrulamasını beklemesi gerekir. İletileri zamanuyumsuz olarak koymak yerine, bazı durumlarda ileti alışverişi başarımını artırabilirsiniz. Bir uygulama bir iletiyi zamanuyumsuz olarak yerleştirdiğinde, kuyruk yöneticisi her çağrımın başarısını ya da hatasını döndürmez, ancak bunun yerine düzenli aralıklarla hata denetimi yapabilirsiniz.

Bir hedefin, kuyruk yöneticisinin iletiyi güvenli bir şekilde alıp almadığı belirlenmeden, uygulamaya denetim işlevinin geri döndürülüp döndürülmeyeceği, aşağıdaki özelliklere bağlıdır:

#### JMS hedef özelliği ( PUTASYCALLI ) (kısa ad-PAALD).

PUTASYNCALLOWED controls whether JMS applications can put messages asynchronously, if this option is allowed by the underlying queue or topic that the JMS destination represents.

#### IBM MQ kuyruğu ya da konu özelliği DEFPRESP (Varsayılan put yanıt tipi).

DEFPRESP, kuyruğa ileti koyan uygulamaların ya da konuya ilişkin ileti yayınlayıp zamanuyumsuz olarak koyma işlevselliğini kullanabileceğini belirtir.

Aşağıdaki çizelge, PUTASINCALLI ve DEFPRESP özelliklerinin olası değerlerini ve zamanuyumsuz koyma işlevini etkinleştirmek için kullanılan değerlerin birleşimlerini göstermektedir:

Çizelge 45. İletilerin bir hedefe zamanuyumsuz bir şekilde konması durumunda, PUTASINCALLI ve DEFPRESP özelliklerinin bir araya gelmesiyle nasıl birleştirilir.

IBM MQ kuyruk özelliği	PUTASYNCALDIC = NO.	PUTASINCALLI = YES (EVET)	PUTASYNCALLID = AS_DEST ya da AS_Q_DEF ya da AS_T_DEF
DEFPRESP=SYNC	Zamanuyumsuz put işlevi etkin değil	Zamanuyumsuz koyma işlevi etkinleştirildi	Zamanuyumsuz put işlevi etkin değil
DEFPRESP=ASYN	Zamanuyumsuz put işlevi etkin değil	Zamanuyumsuz koyma işlevi etkinleştirildi	Zamanuyumsuz koyma işlevi etkinleştirildi

For messages sent in a transacted session, the application ultimately determines whether the queue manager has received the messages safely when it calls commit().

Bir uygulama, hareket eden bir oturum içinde kalıcı iletiler gönderirse ve bir ya da daha fazla ileti güvenli bir şekilde alınmazsa, işlem kesinleştirilemez ve kural dışı durum üretir. Ancak, bir uygulama hareket edilen bir oturum içinde kalıcı olmayan iletiler gönderirse ve bir ya da daha fazla ileti güvenli bir şekilde alınmazsa, işlem başarıyla kesinleştirilir. Uygulama, kalıcı olmayan iletilerin güvenli bir şekilde ulaşılmadığına ilişkin herhangi bir geri bildirim almaz.

İşlem dışı 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.

If a check discovers that one or more messages were not received safely, and the application has registered an exception listener with the connection, IBM MQ classes for JMS calls the onException() method of the exception listener to pass a JMS exception to the application.

JMS kural dışı durumu, JMSWMQ0028 hata koduna sahip ve bu kod aşağıdaki 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 sıfır olup bu, bu tür denetlerin yapılmadığı anlamına gelir.

Bu eniyileme, istemci kipinde bir kuyruk yöneticisine bağlanan ve bir dizi iletiyi hızlı bir şekilde gönderip göndermesi gereken bir uygulamaya en çok yarar sağlar, ancak gönderilen her ileti için kuyruk yöneticisinden hemen geribildirim gerektirmez. Ancak, bir uygulama bağ tanımları kipindeki bir kuyruk yöneticisine bağlansa da, bu eniyilemeyi yine de kullanabilir, ancak beklenen performans avantajı da bu kadar büyük değildir.

#### *Using read ahead with IBM MQ classes for JMS*

IBM MQ tarafından sağlanan okuma öncesinde okuma işlevi, bir işlemin dışında alınan kalıcı olmayan iletilerin, bir uygulama tarafından istekte bulunmadan önce IBM MQ classes for JMS ' ye gönderilmesini sağlar. IBM MQ classes for JMS , iletileri bir iç arabelleğiyle saklar ve uygulama bunları sorduğunda, iletileri uygulamaya geçirir.

Bir hareketin dışındaki bir hedeften gelen iletileri almak için MessageConsumers ya da MessageListeners kullanan IBM MQ classes for JMS uygulamaları, ileriye okuma işlevini kullanabilir. İleriye okumanın kullanılması, bu nesnelere kullanan uygulamaların ileti aldıklarında daha iyi bir performansa neden olacak şekilde yararlanmasını sağlar.

Whether an application that uses MessageConsumers or MessageListeners can use read ahead depends upon the following properties:

#### **JMS hedef özelliği READAHEADINE (kısa ad-RAALD).**

READAHEADIZIN, JMS hedefinin, JMS hedefinin temsil ettiği temel kuyruk ya da konu ile ilgili olarak, bir hareketin dışında kalıcı olmayan iletileri alırken ya da göz atarken okuma öncesinde okuma kullanıp kullanamayacağını denetler.

#### **IBM MQ kuyruğu ya da konu özelliği DEFREADA (Varsayılan okuma değeri).**

DEFREADA, bir hareketin dışındaki kalıcı olmayan iletilerin alılıp alılmadığını belirten uygulamaların önden okuma kullanabileceğini belirtir.

Aşağıdaki çizelge, READAHEADIZIN ve DEFREADA özelliklerine ilişkin olası değerleri ve okuma öncesinde okuma işlevini etkinleştirmek için kullanılan değerlerin birleşimlerini göstermektedir:

<i>Çizelge 46. READAHEADIZIN ve DEFREADA özelliklerinin, bir hareketin dışında kalıcı olmayan iletiler alınırken ya da göz atılırken önceden okunup okunmamasını saptamak için birleştirilmesine neden olur.</i>			
<b>IBM MQ kuyruk özelliği</b>	<b>READAHEADALLOWED = YES</b>	<b>READAHEADALLOWED = NO</b>	<b>AS_DEST ya da AS_Q_DEF ya da AS_T_DEF</b>
DEFREADA = NO	Okuma tamamlama işlevi etkin	Okuma öncesinde okuma işlevi etkinleştirilmedi	Okuma öncesinde okuma işlevi etkinleştirilmedi
DEFREADA = EVET	Okuma tamamlama işlevi etkin	Okuma öncesinde okuma işlevi etkinleştirilmedi	Okuma tamamlama işlevi etkin
DEFREACA = DEVRE DIŞI	Okuma öncesinde okuma işlevi etkinleştirilmedi	Okuma öncesinde okuma işlevi etkinleştirilmedi	Okuma öncesinde okuma işlevi etkinleştirilmedi

İleriye okuma işlevi etkinleştirilirse, uygulama tarafından bir MessageConsumer ya da MessageListener yaratıldığında, IBM MQ classes for JMS , MessageConsumer ya da MessageListener ' in izlediği hedef için bir iç arabellek yaratır. Her MessageConsumer ya da

MessageListener için bir iç arabellek var. Uygulama, aşağıdaki yöntemlerden birini çağırdığında, kuyruk yöneticisi kalıcı olmayan iletileri IBM MQ classes for JMS ' e göndermeye başlar:

- `MessageConsumer.receive()`
- `MessageConsumer.receive(long timeout)`
- `MessageConsumer.receiveNoWait()`
- `Session.setMessageListener(MessageListener listener)`

The IBM MQ classes for JMS automatically returns the first message back to the application, by the method call that the application has made. Kalıcı olmayan diğer iletiler, hedef için yaratılmış iç arabellekte IBM MQ classes for JMS tarafından saklanır. Uygulama bir sonraki iletiyi işlemeye istediğinde, IBM MQ classes for JMS iç arabelleğindeki bir sonraki iletiyi döndürür.

The IBM MQ classes for JMS requests more non-persistent messages from the queue manager when the internal buffer is empty.

The internal buffer that is used by the IBM MQ classes for JMS is deleted when an application closes a `MessageConsumer`, or the JMS Session that a `MessageListener` is associated with.

`MessageConsumer` için, iç arabelleğindeki işlenmemiş iletiler kaybedilir.

When using `MessageListeners`, what happens to the messages in the internal buffer depends upon the JMS destination property `READAHEADCLOSEPOLICY` (short name - `RACP`). The default value of the property is `DELIVER_ALL`, which means that the JMS session that was used to create the `MessageListener` is not closed until all of the messages in the internal buffer are delivered to the application. Özellik `DELIVER_CURRENT` olarak ayarlandıysa, geçerli ileti uygulama tarafından işlendikten sonra JMS oturumu kapatılır ve iç arabelleğindeki geri kalan iletiler atılır.

#### *IBM MQ classes for JMS için yayınları alıkoyma*

Bir IBM MQ classes for JMS istemcisi, alıkonan yayınları kullanmak üzere yapılandırılabilir.

Bir yayıncı, konuya ilgi duyan gelecekteki abonelere gönderilebilmesi için bir yayının kopyasının saklaması gerektiğini belirtebilir. Bu, `JMS_IBM_RETAIN` tamsayı özelliği 1 değerine ayarlanarak IBM MQ classes for JMS içinde yapılır. Sabit değerler, `com.ibm.msg.client.jms.JmsConstants` arabiriminde bu değerler için tanımlanmış olmalıdır. Örneğin, alıkonan bir yayın olarak ayarlamak için *İltadlı* bir ileti oluşturduysanız şu kodu kullanın:

```
// set as a retained publication
msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION);
```

Şimdi iletiyi normal olarak gönderebilirsiniz. `JMS_IBM_RETAIN`, alınan bir iletide de sorgulanabilir. Bu nedenle, alınan iletinin alıkonan bir yayın olup olmadığını sorgulamak mümkündür.

### ***XA support in IBM MQ classes for JMS***

JMS , bir JEE kapsayıcısı içinde desteklenen bir hareket yöneticisiyle bağ tanımları ve istemci kiplerindeki XA uyumlu işlemleri destekler.

Bir uygulama sunucusu ortamında XA işlevselliğine gereksinim duyarsanız, uygulamanızı uygun şekilde yapılandırmanız gerekir. Dağıtılmış hareketleri kullanmak için uygulamaların nasıl yapılandırılacağına ilişkin bilgi edinmek için uygulama sunucunuzun kendi belgelerine bakın.

IBM MQ kuyruk yöneticisi, JMS için hareket yöneticisi olarak işlev göremiyor.

### ***JMS 2.0 işlevselliğini kullanma***

JMS 2.0 , IBM MQ classes for JMS' a birçok yeni işlev alanı sunar.

IBM MQ 8.0 ya da sonraki yayın düzeylerine ilişkin bir JMS uygulaması geliştirirken, bu işlevin kuyruk yöneticinizde etkisini göz önünde bulundurmanız gerekebilir.

#### ***İlgili kavramlar***

IBM MQ Java dil arabirimleri

### JMS 2.0 teslim gecikmesi

JMS 2.0 ile bir ileti gönderirken bir teslim gecikmesi belirleyebilirsiniz. Kuyruk yöneticisi, belirtilen teslim gecikmesi geçinceye kadar iletiyi sunmaz.

Bir uygulama, `MessageProducer.setDeliveryDelay(long deliveryDelay)` ya da `JMSProducer.setDeliveryDelay(long deliveryDelay)` kullanarak bir ileti gönderdiğinde milisaniye cinsinden bir teslim gecikmesini belirtebilir. Bu değer, iletinin gönderildiği zamana eklenir ve başka bir uygulamanın bu iletiyi alabileceği en eski zamanı verir.

IBM MQ 8.0 ve sonraki yayın düzeylerinde, teslim gecikmesi, tek bir iç konaklama kuyruğu kullanılarak gerçekleştirilir. Sıfır olmayan teslim gecikmesi olan iletiler, teslim gecikmesini ve hedef kuyrukla ilgili bilgileri belirten bir üstbilgiyle bu kuyruğa yerleştirilir. Teslim gecikmesi işlemcisi adı verilen kuyruk yöneticisinin bir bileşeni, aşama kuyruğunda iletileri izler. Bir iletinin teslim gecikmesi tamamlandığında, ileti aşama kuyruğundan alınır ve hedef kuyruğa yerleştirilir.

## İleti alışverişi istemcileri

Teslim gecikmesinin IBM MQ uygulaması yalnızca JMS istemcisini kullanırken kullanılmak üzere sağlanır. Teslim gecikmesini IBM MQ ile kullanıyorsanız, aşağıdaki kısıtlamalar geçerli olur. Bu kısıtlamalar `MessageProducers` ve `JMSProducers`'ye eşit olarak geçerlidir, ancak `JMSRuntimeException`, `JMSProducers` durumunda atılır.

- Any attempt to call `MessageProducer.setDeliveryDelay` with a nonzero value when connected to a queue manager earlier than IBM MQ 8.0, results in a `JMSEException` with a `MQRC_FUNCTION_NOT_SUPPORTED` message.
- `MQBND_BIND_NOT_FIXED` dışında bir **DEFBIND** değeri olan kümelenmiş hedefler için teslim gecikmesi desteklenmez. `MessageProducer` 'un sıfır olmayan bir teslim gecikme süresi varsa ve bu gereksinimi karşılamayan bir hedefe göndermek için bir girişimde bulunulursa, arama sonucunda bir `MQRC_OPTIONS_ERROR` iletisi içeren bir `JMSEException` 'de arama sonuçları olur.
- Önceden belirlenmiş sıfır olmayan bir teslim gecikmesinden daha az ya da tam tersi olarak, bir `MQRC_EXPIRY_ERROR` iletisi içeren bir `JMSEException` ile sonuçlanacak bir süre için zaman ayarlamaya çalışılır. Bu denetleme, seçilen tam işlem kümesine bağlı olarak, `setTimeToLive` ya da `setDeliveryDelay` ya da `send` yöntemlerinin çağrısında gerçekleştirilir.
- Alıkonan yayınların ve teslimat gecikmesinin kullanılması desteklenmez. Bu ileti, bir `MQRC_OPTIONS_ERROR` iletisiyle `JMSEException` içinde `msg.setIntProperty(JmsConstants.JMS_IBM_RETAIN, JmsConstants.RETAIN_PUBLICATION)` sonuçları kullanılarak alıkonacaksa, teslim gecikmesi içeren bir ileti yayınlanmaya çalışıldı.
- Teslim gecikmesi ve ileti gruplaması desteklenmiyor ve bu birleşim sonuçlarını `MQRC_OPTIONS_ERROR` iletisiyle `JMSEException` ile kullanma girişiminde bulunuldu.

Any failure to send a message with delivery delay results in the client throwing a `JMSEException` with a suitable error message, for example queue full. Bazı durumlarda, hata iletisi hedef hedefe ya da aşama kuyruğuna ya da her ikisine de uygulanabilir.

**Not:** IBM MQ , iş birimi kesinleştirilmemiş olsa da, bir iş birimine ileti yerleştiren uygulamaların yeniden aynı iletiyi almak için bir ileti yerleştirmesini sağlar. Bu teknik, iş birimi kesinleştirilinceye ve sonuç olarak hedef hedefe gönderilmeden, ileti aşama kuyruğuna yerleştirilmediği için teslimat gecikmesiyle birlikte çalışmaz.

## Yetkilendirme

IBM MQ , uygulamanın sıfır olmayan bir teslim gecikmesi ile bir ileti gönderdiğinde, özgün hedef hedefinde yetkilendirme denetimlerini gerçekleştirir. Uygulama yetkilendirilmediyse, gönderme işlemi başarısız olur. Kuyruk yöneticisi bir iletinin teslim gecikmesinin tamamlandığını algıladığında, hedef kuyruğu açar. Bu noktada herhangi bir yetki denetimi gerçekleştirilmedi.

## SYSTEM.DDELAY.LOCAL.QUEUE

Sistem kuyruğu, SYSTEM.DDELAY.LOCAL.QUEUE(Kuyruk), teslim gecikmesini uygulamak için kullanılır.

- **Multi** Multiplatforms, SYSTEM.DDELAY.LOCAL.QUEUE vardır. Sistem kuyruğu, MAXMSGL ve MAXDEPTH özniteliklerinin beklenen yük için yeterli olması için değiştirilmelidir.
- **z/OS** IBM MQ for z/OS, SYSTEM.DDELAY.LOCAL.QUEUE , yerel ve paylaşılan kuyruklara teslim gecikmesi ile gönderilen iletiler için bir hazırlama kuyruğu olarak kullanılır. z/OSüzerinde, kuyruk yaratılmalı ve MAXMSGL ve MAXDEPTH özniteliklerinin beklenen yük için yeterli olması için tanımlanmalıdır.

Bu kuyruk yaratıldığında, mümkün olduğunca az sayıda kullanıcının bu kuyruğa erişimi olması için bu kuyruk güvenli kılınmalıdır. Kuyruğa erişim, yalnızca bakım ve izleme amacıyla olmalıdır.

Bir JMS uygulaması tarafından sıfır olmayan bir teslim gecikmesi ile bir ileti gönderildiğinde, bu ileti yeni bir ileti tanıtıcısıyla bu kuyruğa konmaktadır. Özgün ileti tanıtıcısı, iletinin ilinti tanıtıcısında yer alıyor. Bu ilinti tanıtıcısı, bir uygulamanın, gerektiğinde konaklatma kuyruğundan bir ileti almasını sağlar; örneğin, büyük bir teslim gecikmesi yanlışlıkla kullanıldıysa.

## z/OSile ilgili önemli noktalar

### z/OS

Sisteminiz z/OSişletim sisteminde çalışıyorsa, teslim gecikmesini kullanmak istiyorsanız, dikkate alınması gereken ek noktalar vardır.

Teslimat gecikmesi kullanılacaksa, sistem kuyruğu SYSTEM.DDELAY.LOCAL.QUEUE tanımlı olmalıdır. Bu değer, beklenen yükü için yeterli olan bir depolama sınıfından ve INDXTYPE (NONE) ve MSGDLVSQ (FIFO) belirtimiyle tanımlanmalıdır. Sistem kuyruğunun bir örnek tanımlaması, CSQ4INSG JCL ' de sağlanır, bu tanımlamayı geçersiz kılar.

## Paylaşılan kuyruklar

Teslim gecikmesi, iletileri paylaşılan kuyruklara göndermek için desteklenir. Ancak, hedef kuyruğun paylaşıp paylaşılmamasından bağımsız olarak, yalnızca tek bir özel konaklama kuyruğu kullanılır. Gecikme tamamlanınca, gecikmiş iletiyi hedef paylaşılan kuyruğuna göndermek için, o özel kuyruğa sahip olan kuyruk yöneticisi çalışıyor olmalıdır.

**Not:** Kalıcı olmayan bir ileti, bir teslim gecikmesi ile paylaşılan bir kuyruğa konursa ve konaklatma kuyruğunun sahibi olan kuyruk yöneticisi sona erdirilirse, özgün ileti kaybolur. Bir paylaşılan kuyruğa teslim gecikmesiyle gönderilen ve kalıcı olmayan iletilerin, teslim gecikmesi olmadan gönderilen kalıcı olmayan iletiler, paylaşılan bir kuyruğa gönderilmeden kaybedilmesi olasılığı daha yüksektir.

## Hedef hedef çözümü

İleti bir kuyruğa gönderilirse, çözüm iki kez yönlendirilir; bir kez JMS uygulaması tarafından ve kuyruk yöneticisi tarafından bir kez, iletiyi aşama kuyruğundan alır ve hedef kuyruğa gönderir.

Target subscriptions for publications are matched when the JMS application calls the send method.

Kuyruk tanımlamasına göre kalıcılık ya da önceliğe sahip bir ileti gönderilirse, değer ikinci olarak değil, ilk çözünürlükte ayarlanır.

## Süre bitim aralığı

Teslim gecikmesi, süre bitimi özelliğinin davranışını korur, MQMD.Expiry. Örneğin, bir ileti, 20 bin ms süre bitimi ve 5.000 ms teslimat gecikmesi içeren bir JMS uygulamasından konulduysa ve geçen 10.000 ms süresinden sonra, MQMD.expiry alanının değeri yaklaşık bir saniyenin yaklaşık 50 saniyesi olabilir. Bu değer, iletinin yerleştirdiği zamandan 15 saniyenin geçeceğini, bu sürenin dolduğunda da zaman geçtiğini gösterir.

Konaklatma kuyruğunda bir ileti sona ererse ve MQRO\_EXPIRATION\_\* seçeneklerinden biri ayarlandıysa, oluşturulan rapor, uygulama tarafından gönderilen özgün ileti için, teslim gecikme bilgilerini içeren üstbilgi kaldırılır.

## Teslim gecikmesi işlemcisinin durdurulması ve başlatılması

**z/OS** z/OS üzerinde, teslim gecikmesi işlemcisi, kuyruk yöneticisi MSTR adres alanına tümleştirilmiştir. Kuyruk yöneticisi başlatıldığında, teslim gecikmesi işlemcisi de başlatılır. Konaklatma kuyruğu kullanılabilirse, kuyruğu açar ve işlenmek üzere iletilerin gelmesini bekler. Konaklatma kuyruğu tanımlı değilse ya da alma işlemi için geçersiz kılındıysa ya da başka bir hata oluşursa, teslim gecikme süresi işlemcisi kapanır. Konaklatma kuyruğu daha sonra tanımlanırsa ya da değiştirilmek üzere değiştirilirse, teslim gecikmesi işlemcisi yeniden başlatılır. Teslim gecikmesi işlemcisi başka herhangi bir nedenle kapatılırsa, hazırlama kuyruğunun PUT özniteliğinin ETKİN ' den DISABLE olarak değiştirilmesi ve yeniden ENABLE değerine geri çevrilerek yeniden başlatılması gerekir. Teslim gecikmesi işlemcisini herhangi bir nedenle durdurmanız gerekiyorsa, hazırlama kuyruğunun PUT özniteliğini DISABLE olarak ayarlayın.

**Multi** On Multiplatforms, the delay processor starts with the queue manager, and is automatically restarted in the event of a recoverable failure.

## Hedef kuyruğa konmamanın başarısız olması

Gecikmesi tamamlandığında, gecikmeli bir ileti hedef kuyruğa konulamazsa, ileti, rapor seçeneklerinde belirtildiği gibi ele alınır: ya atılır ya da çıkmaz mektup kuyruğuna gönderilir. Bu işlem başarısız olursa, iletiyi daha sonra yerleştirmek için bir girişimde bulunmaya çalışılır. İşlem başarılı olursa, bir kural dışı durum raporu oluşturulur ve rapor istenirse, belirtilen kuyruğa gönderilir. Rapor iletisi gönderilemediyse, rapor iletisi ölü mektup kuyruğuna gönderilir. Raporu ölü mektup kuyruğuna yollamak başarısız olursa ve ileti kalıcıdır ise, tüm değişiklikler atılır ve özgün ileti geriye işlenir ve daha sonra yeniden teslim edilir. İleti kalıcı olmayan bir iletirse, rapor iletisi atılır, ancak diğer değişiklikler kesinleştirilir. Bir abonenin aboneliği kaldırıldığı için ya da bağlantısı kesilen bir abonenin bağlantısı kesildiğinden, gecikmeli bir yayın teslim edilemezse, ileti sessiz bir şekilde atılır. Rapor iletileri daha önce açıklanan şekilde oluşturulabilmekte.

Gecikmeli bir yayın bir aboneye teslim edilemezse ve bunun yerine, ölü mektup kuyruğuna konursa ve çıkış kuyruğu kuyruğuna konması başarısız olursa, ileti atılır.

To reduce the likelihood of the put to the target queue failing after the delivery delay has completed, the queue manager performs some basic checks when the JMS client sends a message with a nonzero delivery delay. Bu denetimler, iletinin izin verilen ileti uzunluğu üst sınırından büyükse ve kuyruk dolduysa, kuyruğun geçersiz kılındığını ve geçersiz kılındığını da içerir.

## Yayınla/abone ol

Matching of a publication to available subscriptions occurs when the JMS application sends a message with a nonzero delivery delay. Eşleşen her abone için SYSTEM.DDELAY.LOCAL.QUEUE kuyruğu, teslim gecikmesi tamamlanmaya kadar alıkonacağı yer. Bu abonelerden biri başka bir kuyruk yöneticisi için yetkili abonelikse, teslim gecikmesi tamamlandıktan sonra o kuyruk yöneticisindeki fan-out işlemi gerçekleşir. Bu, diğer kuyruk yöneticisinde abonelere, abone olunmadan önce yayınlanmış olan yayınların gönderilmesine neden olabilir. Bu, JMS 2.0 belirtiminden bir sapmadır.

Yayınla/abone olma ile teslim gecikmesi yalnızca, hedef konu (N) PMSGDLV = ALLAIL ile yapılandırıldıysa desteklenir. Diğer değerleri MQRC\_PUBLICATION\_FAILURE hatasıyla kullanma girişiminde bulunuldu. Teslim gecikmesi işlemcisi iletiyi hedef kuyruğa yerleştirirken başarısız olursa, sonuç "Hedef kuyruğa koyma başarısız" bölümünde anlatıldığı gibi olur.

## Rapor iletileri

Tüm rapor seçenekleri, teslim işlemcisi tarafından, dikkate alınmayan, ancak hedef kuyruğa gönderildiği sırada iletilenden farklı olan, teslim işlemcisi tarafından desteklenir ve iletilir:

- MQRO\_COA\*
- MQRO\_COD\*
- MQRO\_PAN/MQRO\_NAN
- MQRO\_ACTIVITY

#### *Eşkopyalanmış ve paylaşılan abonelik*

IBM MQ 8.0 ya da sonraki bir yayın düzeyiyle, birden çok tüketicinin aynı aboneliğe erişmesine izin vermek için iki yöntem vardır. Bu iki yöntem, eşkopyalanmış abonelikler kullanılarak ya da paylaşılan abonelikler kullanılarak olur.

## **Eşkopyalanmış**

Eşkopyalanmış abonelik bir IBM MQ uzantısıdır. Eşkopyalanmış abonelikler, farklı Java sanal makinelerinde (JVM ' ler) birden çok tüketicinin aboneliğe eşzamanlı olarak erişmelerini sağlar. Bu davranış, **CLONESUPP** özelliği bir connectionFactory nesnesindeki Etkin olarak ayarlanarak kullanılabilir. Varsayılan olarak **CLONESUPP** Devre Dışı' dır. Eşkopyalanmış abonelikler yalnızca dayanıklı aboneliklerde etkinleştirilebilir. **CLONESUPP** etkinleştirilirse, bu connectionFactory kullanılarak yapılan her bir sonraki bağlantı eşkopyalandı.

Bir ya da daha fazla sayıda tüketici bu abonelikten ileti almak için yaratıldıysa, kalıcı bir abonelik, aynı abonelik adı belirtilerek yaratıldıysa, eşkopyalanmış olarak düşünülebilmektedir. This can be done only if the connection under which the consumers were created has **CLONESUPP** set to Etkin on the MQConnectionFactory. Aboneliğin konusu üzerinde bir ileti yayınlandığında, bu iletinin bir kopyası aboneliğe gönderilir. Bu ileti, tüketicilerden herhangi biri için kullanılabilir, ancak bunu yalnızca bir kişi alır.

**Not:** Eşkopyalanmış aboneliklerin etkinleştirilmesi JMS belirtimini genişletir.

## **Paylaşılan abonelikler**

JMS 2.0 belirtimi, bir konu aboneliğinden gelen iletilerin birden çok tüketici arasında paylaşılmasına olanak sağlayan paylaşılan abonelikler sunar. Abonelikten gelen her ileti, bu abonelikte bulunan tüketicilerden yalnızca birine teslim edilir. Paylaşılan abonelikler, JMS 2.0 API ' ya ilgili çağrıyla etkinleştirilir.

API ' ler aşağıdaki yöntemlerden birini kullanarak çağrılabilir:

- Bir Java SE uygulamasından (ya da Java EE Client Container).
- Bir sunucu uygulamasından ya da bir MDB ' nin uygulanmasından.

JMS 2.0 belirtimi, bir MDB 'yi sharedSubscription' dan herhangi bir standart şekilde 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 429](#) ve [“sharedSubscription özelliğinin nasıl tanımlamaya ilişkin örnekler” sayfa 444](#).

Bir paylaşılan abonelik etkinleştirilirse, paylaşılamaz olarak kullanılamaz.

Paylaşılan abonelikler, dayanıklı ya da dayanıklı olmayan abonelikler olarak yaratılabilir. Olağan JMS yapılandırışındaki kuyruk yöneticisi tarafında ayrı ayrı nesne yaratmak için gereken herhangi bir koşul yoktur; gereken nesnelere devingen olarak yaratılır.

## **Paylaşılan ya da eşkopyalanmış abonelikler arasında**

Paylaşılan ya da klonlanmış abonelikler kullanıp kullanmayacağınızı belirlerken, her ikisinin de avantajlarını göz önünde bulundurun. IBM MQ ' in belirli bir uzantısı yerine, belirtim tanımlı davranış olduğu için, paylaşılan abonelikler de kullanılabilir.

Aşağıdaki tabloda, paylaşılan ve eşkopyalanmış abonelikler arasında karar verirken dikkate alınacak bazı noktalar yer almaktadır:

Çizelge 47. Paylaşılan abonelikler ve eşkopyalanmış abonelikler arasında seçim yaparken dikkat edilecek noktalar

Paylaşılan Aboneler	Eşkopyalanmış
Paylaşılan abonelikler, JMS 2.0 belirtiminin standart bir parçasıdır.	Eşkopyalanmış abonelikler, belirli bir IBM MQ uzantıdır.
Paylaşılan abonelikler, belirtik API yöntemi çağrılarını kullanılarak yaratılır.	Cloned subscriptions are controlled administratively at the ConnectionFactory level.
Paylaşılan abonelikler, dayanıklı ya da dayanıklı olamaz.	Eşkopyalanmış abonelikler yalnızca dayanıklı olabilir.
Paylaşılan abonelikler, bireysel abonelik temelinde belirtik olarak yaratılır.	Eşkopyalanmış abonelikler, işlevin etkinleştirildiği bir bağlantı altında tüm kalıcı abonelikler için kullanılır.
Bir abonelik paylaşılan olarak oluşturulduysa, daha sonra paylaşılmayan olarak değiştirilemez (ya da tersi).	Bir abonelik, sahip olan bağlantının <b>CLONESUPP</b> özelliği değişirse, yeniden açıldığında, eşkopyalanmış olarak, eşkopyalanmış olarak değiştirilebilir.

### İlgili kavramlar

Aboneler ve abonelikler

Abonelik dayanıklılığı

### İlgili görevler

JMS 2.0 paylaşılan aboneliklerinin kullanılması

### İlgili başvurular

“sharedSubscription özelliğinin nasıl tanımlamaya ilişkin örnekler” sayfa 444

Bir WebSphere Liberty server.xml dosyası içinde bir etkinleştirme belirtiminin sharedSubscription özelliğini tanımlayabilirsiniz. Diğer bir seçenek olarak, ek açıklamaları kullanarak bir ileti odaklı Bean (MDB) içinde özelliği tanımlayabilirsiniz.

### ÇALIŞTIRMA

#### SupportMQExtensions özelliği

JMS 2.0 belirtimi, belirli davranışların çalışma biçimiyle ilgili değişiklikleri tanıtır. IBM MQ 8.0 ve daha sonra, bu değiştirilen davranışları önceki somutlamaya geri döndürmek için TRUE olarak ayarlanabilen com.ibm.mq.jms.SupportMQExtensions özelliğini içerir.

Three areas of functionality are reverted by setting SupportMQExtensions to *Doğru*:

#### İleti önceliği

İletilere öncelik atanabilir: 0 - 9. JMS 2.0 öncesinde iletiler, bir kuyruğun varsayılan önceliğinin kullanıldığını gösteren -1 değerini de kullanabilirdi. JMS 2.0 does not allow a message priority of -1 to be set. SupportMQExtensions değerinin açılması, -1 değerinin kullanılmasını sağlar.

#### İSTEMCİ TANITICISI

JMS 2.0 belirtimi, boş değerli olmayan istemci tanıtıcılarının bir bağlantı gerçekleştirdiklerinde benzersizlik olup olmadığını denetimlerini gerektirir. SupportMQExtensions' in açılması, bu gereksinimin göz ardı edildiğine ve bir istemci tanıtıcısının yeniden kullanılabileceği anlamına gelir.

#### NoLocal

JMS 2.0 belirtimi, bu sabit açık olduğunda, bir tüketici aynı istemci tanıtıcısı tarafından yayınlanan iletileri alamayabilmenizi gerektirir. JMS 2.0 öncesinde, bu öznitelik, kendi bağlantısı tarafından yayınlanan iletilerin alınmasını önlemek için bir abonede ayarlanmıştır. SupportMQExtensions ' in açılması bu davranışı önceki somutlamaya geri çevirir.



com.ibm.mq.jms.SupportMQExtensions özelliği, com.ibm.mqjms.jar içinde yer alan bir boole özelliği. Bu özellik aşağıdaki gibi ayarlanabilir:

```
java -Dcom.ibm.mq.jms.SupportMQExtensions=true
```

Bu özellik, **java** komutu üzerinde standart bir JVM System özelliği olarak ya da IBM MQ classes for JMS yapılandırma dosyası içinde yer alır.

### İlgili kavramlar

[“IBM MQ classes for JMS yapılandırma dosyası” sayfa 88](#)

Bir IBM MQ classes for JMS yapılandırma dosyası, IBM MQ classes for JMS' u yapılandırmak için kullanılan özellikleri belirtir.

### İlgili başvurular

[“JMS istemci davranışını yapılandırmak için kullanılan özellikler” sayfa 94](#)

JMS istemcisinin davranışını yapılandırmak için bu özellikleri kullanın.

#### *Using shared subscriptions in JMS 2.0*

JMS 2.0 , tek bir aboneliğin birden çok tüketici arasında paylaşıldığı shared subscriptions kavramını, yalnızca bir yayın alan tüketicinin herhangi bir zamanda herhangi bir noktada bir yayın almasına neden olur. IBM MQ classes for JMS.

IBM MQ 8.0 ya da sonraki yayın düzeylerine ilişkin bir JMS uygulaması geliştirirken, bu işlevin kuyruk yöneticinizde etkisini göz önünde bulundurmanız gerekebilir.

Paylaşılan aboneliklerin arkasındaki fikir temel olarak yükü birden çok tüketici arasında paylaşır. Sürekli abonelik birden çok tüketici arasında da paylaşılabilir.

Örneğin, aşağıda bir:

- Subscription SUB, subscribing to a topic FIFA2014/UPDATES to receive football match updates, being shared by three consumers C1, C2, and C3
- Producer P1 publishing on the FIFA2014/UPDATES topic

FIFA2014/UPDATES üzerinde bir yayın yapıldığında, yayın yalnızca üç tüketicinin (C1 ya da C2 ya da C3) biri tarafından alınır, ancak tümü için bu yayın alınmaz.

Aşağıdaki örnek, paylaşılan aboneliklerin kullanımını gösterir ve yalnızca ileti gövini almak için JMS 2.0, Message.receiveBody()'daki ek API kullanımını da gösterir.

Bu örnek, FIFA2014/UPDATES konusuna ve bir yayıncı iş parçacıklarına paylaşılan bir abonelik yaratan üç abone iş parçacığı yaratır.

```
package mqv91Samples;

import javax.jms.JMSEException;

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
 */
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 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 stadiumAttendance = 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 Attendance " + stadiumAttendance;
        stadiumAttendance += 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(JMSEException 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 Simplified API using IBM MQ v91 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](#)

## IBM MQ classes for JMS Uygulama Sunucusu Tesisleri

This topic describes how IBM MQ classes for JMS implements the ConnectionConsumer class and advanced functionality in the Session class. Ayrıca, bir sunucu oturum havuzunun işlevi de özetlenir.

**Önemli:** Bu bilgiler yalnızca başvuru içindir. An application must not be written to use this interface: it is used within the IBM MQ resource adapter to connect to Java EE servers. Pratik bağlantı bilgileri için bkz. [“IBM MQ kaynak bağdaştırıcısının kullanılması” sayfa 415.](#)

IBM MQ classes for JMS , *Java Message Service Belirtimi* ' nde belirtilen Application Server Faciliti'lerini (ASF) destekler (bkz. [Oracle Technology Network for Java Developers](#) ). Bu belirtim, bu programlama modeli içindeki üç rolü tanımlar:

- **JMS sağlayıcısı** , ConnectionConsumer ve gelişmiş Oturum işlevselliği sağlar.
- **Uygulama sunucusu** , ServerSessionhavuzu ve ServerSession işlevlerini sağlar.
- **İstemci uygulaması** , JMS sağlayıcısının ve uygulama sunucusu kaynağının işlevselliğini 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, bir uygulama sunucusunun ConnectionConsumer arabirimini kullanarak JMS uygulaması ile yakın bir şekilde bütünleşmesini sağlar. Bu özellik iletilerin eşzamanlı işlenmesini sağlar. Tipik olarak, bir uygulama sunucusu bir iş parçacığı havuzu yaratır ve JMS uygulaması bu iş parçacıklarının kullanımına sunulan iletileri yapar. JMS' un farkında olan bir uygulama sunucusu ( WebSphere Application Servergibi), ileti odaklı Bean 'ler gibi üst düzey ileti sistemi işlevselliği sağlamak için bu özelliği kullanabilir.

Normal uygulamalar ConnectionConsumer' ı 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 geldiğinde, JMS havuzdaki bir iş parçacığı seçer ve ona bir ileti kümesi gönderir. Bunu yapmak için JMS , ilişkili bir MessageListener' ın onMessage() yöntemini çalıştırır.

Her biri kayıtlı bir MessageListenerile 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 gereklidir.

### ASF ile bir uygulamanın planlanması

Bu bölümde aşağıdakiler de içinde olmak üzere bir uygulamanın nasıl planlanmanız gerektiğini anlatılıyor:

- [“ASF kullanarak noktadan noktaya ileti sistemine ilişkin genel ilkeler” sayfa 309](#)
- [“ASF kullanarak ileti alışverişi yayınlama/abone olma genel ilkeleri” sayfa 309](#)
- [“ASF ' de kuyruktan iletilerin kaldırılması” sayfa 310](#)

- ASF ' deki zehirli mesajların işlenmesi. Bkz. [“IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 209.](#)

#### ASF kullanarak noktadan noktaya ileti sistemine ilişkin genel ilkeler

ASF kullanarak noktadan noktaya ileti alışverişi hakkında genel bilgi için bu konuyu kullanın.

Bir uygulama QueueConnection nesnesinden bir ConnectionConsumer yarattığında, bir JMS kuyruk nesnesi ve bir seçici dizgisi belirtir. ConnectionConsumer (ConnectionConsumer), ilişkili ServerSessionHavuzundaki oturumlara ileti sağlamaya başlar. İletiler kuyruğa gönderilir ve seçiciyle eşleşirlerse, ilişkili ServerSessionHavuzundaki oturumlara teslim edilir.

IBM MQ terimlerinde, kuyruk nesnesi yerel kuyruk yöneticisinde bir QLOCAL ya da QALIAS ile ifade eder. Bu bir QALIAS ise, QALIAS 'ın bir QLOCAL 'a gönderme yapmalıdır. Tam olarak çözümlenen IBM MQ QLOCAL, *temeldeki QLOCAL* olarak bilinir. A ConnectionConsumer is said to be *etkin* if it is not closed and its parent QueueConnection is started.

Her biri farklı seçicilere sahip birden çok ConnectionConsumers için, temeldeki QLOCAL ile aynı şekilde çalıştırılacak şekilde kullanılabilir. Performansı korumak için, istenmeyen iletilerin kuyruğun üzerinde birikmemesi gerekir. İstenmeyen iletiler, etkin olmayan ConnectionConsumer ' in eşleşen bir seçiciye sahip olduğu iletilerdir. Bu istenmeyen iletilerin kuyruktan kaldırılması için QueueConnectionFactory 'yi ayarlayabilirsiniz (ayrıntılar için bkz. [“ASF ' de kuyruktan iletilerin kaldırılması” sayfa 310](#) ). Bu davranışı aşağıdaki iki yoldan birini kullanarak ayarlayabilirsiniz:

- QueueConnectionFactory 'yi MRET (NO) değerine ayarlamak için JMS denetim aracını kullanın.
- Programınızda şunu kullanın:

```
MQQueueConnectionFactory.setMessageRetention(WMQConstants.WMQ_MRET_NO)
```

Bu ayarı değiştirmezseniz, varsayılan değer bu tür istenmeyen iletilerin kuyruğun üzerinde tutulmasını sağlar.

IBM MQ kuyruk yöneticisini ayarladığınızda, aşağıdaki noktaları göz önünde bulundurun:

- Paylaşılan giriş için temeldeki QLOCAL etkinleştirilmelidir. Bunu yapmak için, aşağıdaki MQSC komutunu kullanın:

```
ALTER QLOCAL( your.qlocal.name ) SHARE GET(ENABLED)
```

- Kuyruk yöneticinizin etkinleştirilmiş bir ölü harf kuyruğu olmalıdır. Bir ConnectionConsumer , bir iletiyi ölü-mektup kuyruğuna yerleştirirken bir sorunla karşılaştıysa, temel QLOCAL duraklarından ileti teslimi durur. Bir kuyruk-harf kuyruğu tanımlamak için aşağıdaki adresi kullanın:

```
ALTER QMGR DEADQ( your.dead.letter.queue.name )
```

- ConnectionConsumer çalıştıran kullanıcı, MQOO\_SAVE\_ALL\_CONTEXT ve MQOO\_PASS\_ALL\_CONTEXT ile MQOP gerçekleştirme yetkisine sahip olmalıdır. Ayrıntılar için, altyapınıza ilişkin IBM MQ belgelerine bakın.
- Kuyruğun üzerinde istenmeyen iletiler bırakılırsa, bunlar sistem başarımını düşürmektedir. Therefore, plan your message selectors so that between them, the ConnectionConsumers will remove all messages from the queue.

MQSC komutlarıyla ilgili ayrıntılar için [MQSC komutları](#) başlıklı konuya bakın.

#### ASF kullanarak ileti alışverişi yayınlama/abone olma genel ilkeleri

ConnectionConsumers , belirtilen bir Konu için ileti alır. ConnectionConsumer , dayanıklı ya da kalıcı olmayan bir değer olabilir. ConnectionConsumer tarafından kullanılan kuyruğu ya da kuyrukları belirlemeniz gerekir.

Bir uygulama TopicConnection nesnesinden bir ConnectionConsumer yarattığında, bir Konu nesnesini ve bir seçici dizgisini belirtir. ConnectionConsumer daha sonra, abone olunan konuya ilişkin 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 (Bağlantı Tüketicisi) yaratabilir. This ConnectionConsumer receives messages that have been published on the Topic since the durable ConnectionConsumer was last active. Bu, Konu üzerindeki seçiciyle eşleşen tüm iletileri alır. Ancak, ConnectionConsumer okuma öbekini kullanıyorsa, istemci arabelleğindeki kalıcı olmayan iletileri kaybederse, bu ileti kapandığında kaybedilir.

IBM MQ classes for JMS , IBM MQ ileti alışverişi sağlayıcısı geçiş kipinse, kalıcı olmayan ConnectionConsumer abonelikleri için ayrı bir kuyruk kullanılır. TopicConnectionÜreticisi üzerindeki CCSUB yapılandırılabilir seçeneği, kullanılacak kuyruğu belirtir. Olağan durumda, CCSID ' ler aynı TopicConnectionüreticisini kullanan tüm ConnectionConsumers tarafından kullanılmak üzere tek bir kuyruk belirtir. Ancak, her bir ConnectionConsumer , ardından bir yıldız (\*) işareti ve ardından bir kuyruk adı öneki belirtilerek geçici bir kuyruk oluşturmak mümkündür.

IBM MQ classes for JMS , IBM MQ ileti alışverişi sağlayıcısı geçiş kipinse, Konudaki CCDSUB özelliği, kalıcı abonelikler için kullanılacak kuyruğu belirtir. Yine, varolan bir kuyruk ya da bir kuyruk adı öneki ve ardından bir yıldız işareti (\*) kullanılabilir. Önceden var olan bir kuyruğu belirlerseniz, Konu 'a abone olan tüm dayanıklı ConnectionConsumers bu kuyruğu kullanır. Bir kuyruk adı öneki ve ardından bir yıldız işareti (\*) belirtirseniz, bir kuyruk ilk kez, belirli bir adla yaratılan kalıcı bir ConnectionConsumer ' nin yaratıldığını belirtir. Bu kuyruk, daha sonra aynı adda bir kalıcı ConnectionConsumer yaratılırsa yeniden kullanılır.

IBM MQ kuyruk yöneticisini ayarladığınızda, aşağıdaki noktaları göz önünde bulundurun:

- Kuyruk yöneticinizin etkinleştirilmiş bir ölü harf kuyruğu olmalıdır. Bir ConnectionConsumer , bir iletiyi ölü-mektup kuyruğuna yerleştirirken bir sorunla karşılaştıysa, temel QLOCAL duraklarından ileti teslimi durur. Bir kuyruk-harf kuyruğu tanımlamak için aşağıdaki adresi kullanın:

```
ALTER QMGR DEADQ( your.dead.letter.queue.name )
```

- ConnectionConsumer çalıştıran kullanıcı, MQOO\_SAVE\_ALL\_CONTEXT ve MQOO\_PASS\_ALL\_CONTEXT ile MQOP gerçekleştirme yetkisine sahip olmalıdır. Ayrıntılar için, altyapınıza ilişkin IBM MQ belgelerine bakın.
- Ayrı, adanmış bir kuyruk oluşturarak, tek bir ConnectionConsumer için performansı en iyi duruma getirebilirsiniz. Bu, fazladan kaynak kullanımı maliyetidir.

#### *ASF ' de kuyruktan iletilerin kaldırılması*

Bir uygulama ConnectionConsumers'ı kullandığında, JMS ' un bazı durumlarda kuyruktan iletileri kaldırmasına gerek olabilir.

Bu durumlar aşağıdaki gibidir:

#### **Hatalı biçimlendirilmiş ileti**

JMS , ayrıştırılabilecek bir ileti olabilir.

#### **Zehir mesajı**

Bir ileti geriletme eşiğine ulaşabilir, ancak ConnectionConsumer bunu, geriletme kuyruğunda istekte bulunmaya başarısız olur.

#### **İlgili ConnectionConsumeryok**

Noktadan noktaya ileti alışverişi için, QueueConnectionFactory istenmeyen iletileri alıkoymayacak şekilde ayarlandığında, ConnectionConsumerstarafından istenmeyen bir ileti gönderilir.

Bu durumlarda, ConnectionConsumer iletiyi kuyruktan kaldırma girişiminde bulunur. İletinin MQMD ' nin rapor alanındaki yok etme seçenekleri tam davranışı ayarlar. Bu seçenekler şunlardır:

#### **MQRO\_DEAD\_LETTER\_Q**

İleti, kuyruk yöneticisinin ölü harf kuyruğu için istekte bulunmuyor. Bu varsayılandır.

#### **MQRO\_DISCARD\_MSG**

İleti atılır.

ConnectionConsumer bir rapor iletisi de oluşturur ve bu ileti, iletinin MQMD ' nin rapor alanına da bağlıdır. This message is sent to the message's ReplyToQ on the ReplyToQmgr. Rapor iletisi gönderilirken bir hata varsa, ileti, bunun yerine ölü-mektup kuyruğuna gönderilir. İletinin MQMD kümesi ayrıntılarının

rapor alanındaki kural dışı durum raporu seçenekleri, rapor iletisinin ayrıntılarına ayarlanır. Bu seçenekler şunlardır:

#### **MQRO\_EXCEPTION**

Özgün iletinin MQMD ' yi içeren bir rapor iletisi oluşturulur. Herhangi bir ileti gövdesi verisi içermiyor.

#### **MQRO\_EXCEPTION\_WITH\_DATA**

MQMD ' yi, herhangi bir MQ üstbilgilerini ve 100 baytlık gövde verilerini içeren bir rapor iletisi oluşturulur.

#### **MQRO\_EXCEPTION\_WITH\_FULL\_DATA**

Özgün iletiden tüm verileri içeren bir rapor iletisi oluşturulur.

#### **varsayılan**

Rapor iletisi oluşturulmadı.

Rapor iletileri oluşturulduğunda, aşağıdaki seçenekler onurlandırılır:

- MQRO\_NEW\_MSG\_ID
- MQRO\_PASS\_MSG\_ID
- MQRO\_COPY\_MSG\_ID\_TO\_CORREL\_ID
- MQRO\_PASS\_COREL\_ID

Bir zehir iletisi yeniden istenemezse, belki de ölü-mektup kuyruğu tam ya da yetki yanlış belirtilmiş olduğundan, iletinin kalıcılığına bağlıdır. İleti kalıcı değilse, ileti atılır ve herhangi bir rapor iletisi oluşturulmadı. İleti kalıcıysa, iletilerin o hedef durakları dinleyerek tüm bağlantı tüketicilerine teslim edilir. Bu tür bağlantı tüketicileri kapatılmalı ve sorun, yeniden oluşturulabilmesi ve ileti tesliminin yeniden başlatılabilmesi için çözülmeye önce çözülmelidir.

Herhangi bir sorun oluşmadığından emin olmak için, bir ölü-mektup kuyruğu tanımlanması ve düzenli olarak kontrol etmek önemlidir. Özellikle, ölü harf kuyruğunun derinlik üst sınırına ulaşmadığından ve ileti boyutu üst sınırının tüm iletiler için yeterince büyük olduğundan emin olun.

Bir ileti, ölü-mektup kuyruğuna istekte bulunduğu anda, bu ileti önünde bir IBM MQ ölme harfi üstbilgisi (MQDLH) vardır. MQDLH biçimiyle ilgili ayrıntılar için [MQDLH-Dead-letter header](#) başlıklı konuya bakın. Bir ConnectionConsumer 'in ölü-mektup kuyruğuna yerleştiği ya da bir ConnectionConsumer ' in oluşturduğu rapor iletilerini aşağıdaki alanlar tarafından tanımlayabilir:

- PutApplTipi MQAT\_JAVA (0x1C)
- PutApplAd: "MQ JMS ConnectionConsumer "

Bu alanlar, ölülerin kuyruğunda bulunan iletilerin MQDLH 'de ve rapor iletilerinin MQMD' de yer alıyor. MQMD 'nin geri bildirim alanı ve MQDLH' nin neden alanı, hatayı açıklayan bir kod içerir. Bu kodlarla ilgili ayrıntılı bilgi için bkz. ["ASF ' deki neden ve geri bildirim kodları"](#) sayfa 312. Diğer alanlar, [MQDLH-Dead-letter üstbilgisi](#) içinde anlatıldığı gibidir.

#### *ASF ' de zehirli iletilerin işlenmesi*

Application Server Facilitis (Uygulama Sunucusu Tesisi) içinde, zehirli ileti işleme IBM MQ classes for JMS içindeki başka bir yerde biraz farklı şekilde işlenir.

IBM MQ classes for JMS' ta zehirli ileti işleme hakkında bilgi için bkz. ["IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi"](#) sayfa 209.

Application Server Facilitis (ASF) olanağını kullandığınızda, MessageConsumeryerine ConnectionConsumer, zehirli iletileri işler. ConnectionConsumer , kuyruğun BackoutThreshold ve BackoutRequeueQName özelliklerine göre iletilmekte olan iletileri sağlar.

Bir uygulama ConnectionConsumers' ı kullandığında, bir iletinin yedekleneceği koşullar, uygulama sunucusunun sağladığı oturma bağlıdır:

- Oturum, AUTO\_RELSE ya da DUP\_OK\_RELSE ile birlikte işlem dışı olduğunda, bir ileti yalnızca sistem hatasından sonra ya da uygulama beklenmedik bir şekilde sona erdirildikten sonra yedeklenir.
- When the session is non-transacted with CLIENT\_ACKNOWLEDGE, unacknowledged messages can be backed out by the application server calling Session.recover().

Typically, the client implementation of MessageListener or the application server calls Message.acknowledge(). Message.acknowledge() acknowledges all messages delivered on the session so far.

- When the session is transacted, unacknowledged messages can be backed out by the application server calling Session.rollback().
- Uygulama sunucusu bir XASSession sağladıysa, iletiler dağıtılmış bir harekete bağlı olarak kesinleştirilir ya da yedeklenir. Uygulama sunucusu, işlemin tamamlanması için sorumluluk alır.

### **İlgili kavramlar**

[“IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 209](#)

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 312](#) ve [“ASF ' deki neden ve geri bildirim kodları” sayfa 312](#)de içinde olmak üzere hata işlenmesinin çeşitli yönleri ele gelir.

#### *ASF ' deki hata koşullarından kurtarma*

Bir ConnectionConsumer , ciddi bir hata ortaya çıkardığında, aynı QLOCAL duraklarına ilgi içeren tüm ConnectionConsumers iletisine ileti teslim eder. Bu gerçekleştiğinde, etkilenen bağlantıya kayıtlı olan tüm ExceptionListener ' ler bildirim gönderilir. Bir uygulamanın bu hata koşullarından kurtulması için iki yol vardır.

Tipik olarak, ConnectionConsumer , bir iletiyi ölü-mektup kuyruğuna gönderemezse ya da QLOCAL ' tan ileti okurken hata ortaya çıktığında bu tür ciddi bir hata oluşur.

Etkilenen bağlantıya kayıtlı herhangi bir ExceptionListener (ExceptionListener) bildirildiğinden, sorunun nedenini belirlemek için bunları kullanabilirsiniz. Bazı durumlarda, sistem yöneticisinin sorunu çözmek için müdahale etmesi gerekir.

Bu hata koşullarından kurtulmak için aşağıdaki tekniklerden birini kullanın:

- Etkilenen tüm ConnectionConsumers'ta close () ' u arayın. Uygulama yeni ConnectionConsumers ' ı ancak tüm etkilenen ConnectionConsumers kapatıldıktan sonra yaratabilir ve sistem sorunları çözülmüş olur.
- Etkilenen tüm Connections 'da stop () ' u arayın. Tüm Connections durdurulduktan ve herhangi bir sistem sorunu çözüldükten sonra, uygulama Connections ' ı başarıyla start () ' e verebilir.

#### *ASF ' deki neden ve geri bildirim kodları*

Bir hatanın nedenini belirlemek için neden ve geri bildirim kodlarını kullanın. ConnectionConsumer tarafından oluşturulan ortak neden kodları burada verilmiştir.

Bir hatanın nedenini belirlemek için aşağıdaki bilgileri kullanın:

- Herhangi bir rapor iletisinde geribildirim kodu
- Ölü-mektup kuyruğunda herhangi bir iletinin MQDLH ' de neden kodu

ConnectionConsumers , aşağıdaki neden kodlarını oluşturur.

### **MQRC\_BACKOUT\_THRESHOLD\_ULAS (0x93A; 2362)**

#### **Neden**

İleti, QLOCAL üzerinde tanımlanan Arka Uç Eşiğine ulaştı, ancak Geri Dönme Kuyruğu tanımlanmadı.

Geri Alma Kuyruğu tanımlayamadığınız platformlarda, ileti 20 'nin JMStanımlı geriletme eşiğine ulaşmıştır.

#### **İşlem**

Bu istenmiyorsa, ilgili QLOCAL için Geri Kayan Kuyruğu tanımlayın. Ayrıca, birden çok geri tepenin nedenini de arayın.



## **MQRC\_MSG\_NOT\_MATCHED (0x93B; 2363)**

### **Neden**

Noktadan noktaya ileti sisteminde, kuyruğu izleyen ConnectionConsumers için seçicilerden hiçbiriyle eşleşmeyen bir ileti vardır. Performansı korumak için, ileti, ölü-mektup kuyruğuna istekte bulunmaya devam eder.

### **İşlem**

To avoid this situation, ensure that ConnectionConsumers using the queue provide a set of selectors that deal with all messages, or set the QueueConnectionFactory to retain messages.

Diğer bir seçenek olarak, iletinin kaynağını araştırın.

## **MQRC\_JMS\_FORMAT\_ERROR (0x93C; 2364)**

### **Neden**

JMS , kuyruğun üzerindeki iletiyi yorumlayamaz.

### **İşlem**

İletinin kökenini araştırın. JMS , genellikle BytesMessage ya da TextMessageolarak beklenmeyen bir biçime ilişkin iletiler sağlar. Bazen, ileti çok kötü biçimlendirilirse bu hata başarısız olur.

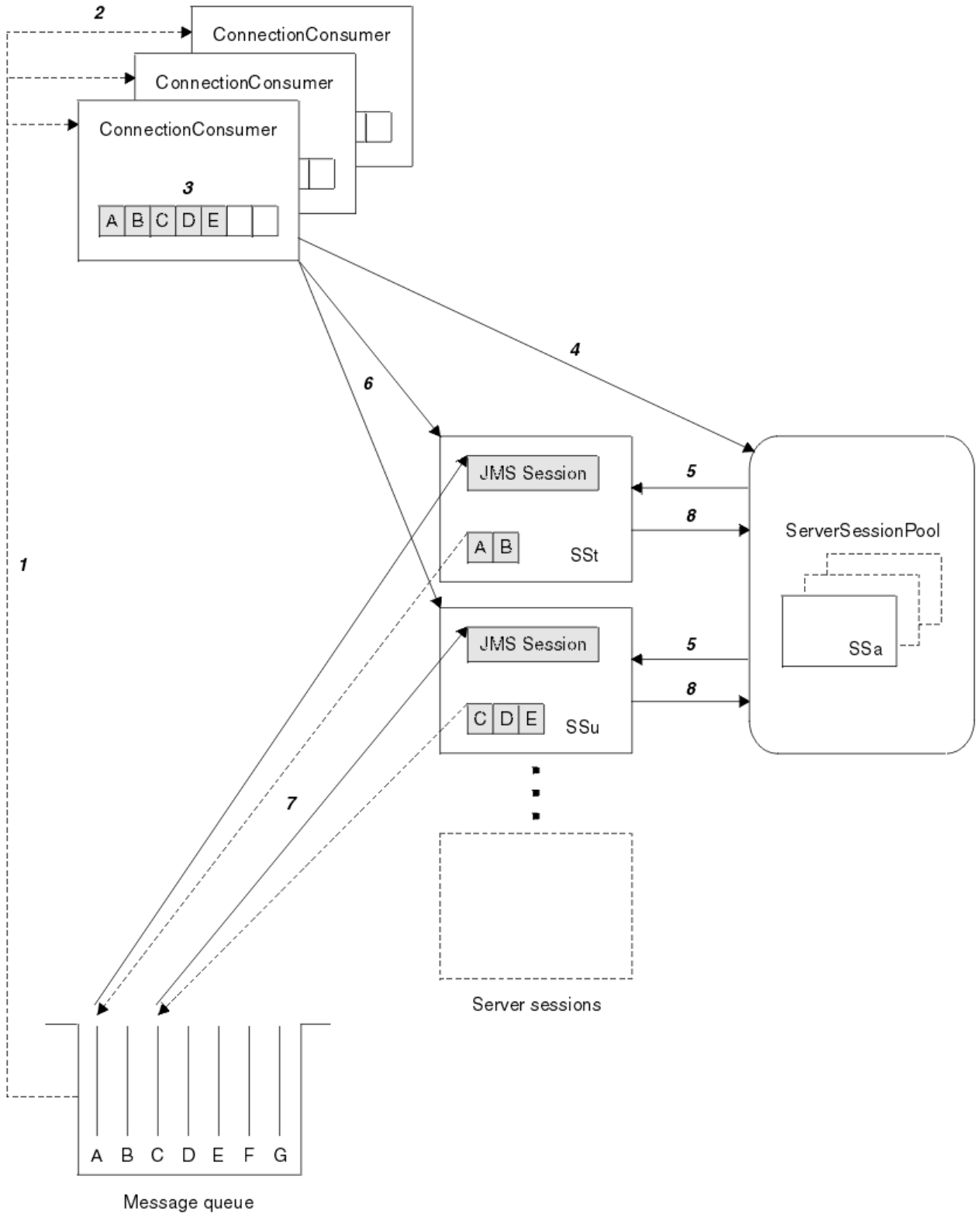
Bu alanlarda görüntülenen diğer kodların nedeni, başarısız olan bir iletiyi geri alma kuyruğuna yeniden istekte bulunmaya çalışmalarından kaynaklanır. Bu durumda, kodda, istekte bulunanın başarısız olmasının nedeni açıklanmaktadır. Bu hataların nedenini tanılamak için [API tamamlama ve neden kodları](#) konusuna bakın.

Rapor iletisi ReplyToQ ' ya konulamazsa, bu ileti, ölü-mektup kuyruğuna konabilecektir. Bu durumda, MQMD ' nin geri bildirim alanı bu konuda açıklandığı gibi tamamlanır. MQDLH 'daki neden alanı, rapor iletisinin ReplyToQ' ya neden yerleştirilememesini açıklar.

### ***The function of a server session pool in AFS***

Bu konuda, bir sunucu oturumu havuzunun işlevi özetlenir.

[Şekil 51 sayfa 314](#) , ServerSessionhavuzu ve ServerSession işlevselliğinin ilkelerini özetler.



Şekil 51. ServerSessionhavuzu ve ServerSession işlevi

1. ConnectionConsumers, kuyruktan ileti başvurularını alır.
2. Her ConnectionConsumer, belirli ileti başvurularını seçer.
3. ConnectionConsumer arabelleği, seçilen ileti başvurularını bulundurur.
4. ConnectionConsumer, ServerSessionHavuzundan bir ya da daha fazla ServerSessions isteğinde bulunur.

5. ServerSessions , ServerSessionHavuzundan ayrılır.
6. ConnectionConsumer , ServerSessions ' a ileti başvuruları atar ve çalışmakta olan ServerSession iş parçacığını başlatır.
7. Her ServerSession , gönderme yapılan iletileri kuyruktan alır. Bu, bunları JMS Oturumuyla ilişkili MessageListener ' dan onMessage yöntemine geçirir.
8. İşlemin tamamlanmasından sonra, ServerSession havuza geri döndürülür.

Bir uygulama sunucusu, olağan durumda ServerSessionhavuzu ve ServerSession işlevlerini sağlar.

## **CICS OSGi JVM sunucusunda IBM MQ classes for JMS kullanılması**

CICS Open Services Gateway Initiative (OSGi) Server ortamı içinde çalışan uygulamalar için standartlara dayalı ileti sistemi desteği, IBM MQ classes for JMSkullanımı aracılığıyla sağlanır.



**Uyarı:** İşletmelerinizin kullandığı CICS sistemine ilişkin sistem gereksinimlerini denetleyin. Ayrıntılı bilgi için [CICS Transaction Server için ayrıntılı sistem gereksinimleri](#) başlıklı konuya bakın.

From IBM MQ 8.0, IBM MQ supports using the IBM MQ classes for JMS in certain versions of the CICS Open Services Gateway initiative (OSGi) Java Virtual Machine (JVM) server.


Bu konu, bir JVM sunucu ortamında IBM MQ classes for JMS ' in nasıl ayarlanmasına ilişkin bir tanıtıdır.

Sisteminizin ayarına ve yapılandırılmasına ilişkin ayrıntılı bilgi için CICS belgelerinde [Bir OSGi JVM sunucusunda IBM MQ classes for JMS ' nin kullanılması](#) belgesine bakın.

### **Genel sınırlamalar**

Bir CICS OSGi JVM sunucusunda IBM MQ classes for JMS kullanılırken aşağıdaki kısıtlamalar geçerli olur:

- İstemci kipi bağlantıları desteklenmez.
- Bağlantılar yalnızca IBM WebSphere MQ 7.1 ya da IBM MQ 8.0 ya da sonraki kuyruk yöneticilerine desteklenir. Bağlantı üreticisine ilişkin **PROVIDERVERSION** özneliği belirtilmemelidir ya da 7 'den büyük ya da bu değere eşit bir değer olmalıdır.
- XA bağlantısı üreticilerinin ( com . ibm . mq . jms .MQXAConnectionFactorygibi) kullanılması desteklenmez.
- CICS OSGi JVM sunucusunda IBM MQ classes for JMS kullanımı yalnızca CICS 5.2 ya da sonraki yayın düzeylerinde desteklenir. CICS 5.2kullanıyorsanız, [APAR PI32151uygulamasını](#) uygulamanız gerekir.

 Before IBM MQ 9.1.0 if you are a Long Term Support user, or before IBM MQ 9.0.1 if you are a Continuous Delivery user, using the IBM MQ classes for JMS in a Liberty JVM server environment is not supported.

### **İlgili görevler**

JMS **PROVIDERVERSION** özelliğini yapılandırma

## **CICS Liberty JVM sunucusunda IBM MQ classes for JMS kullanılması**

From IBM MQ 9.1.0, Java programs running in a CICS Liberty JVM server can use the IBM MQ classes for JMS to access IBM MQ.

IBM MQkaynak bağdaştırıcısının bir IBM MQ 9.1.0 sürümünü kullanıyor olmalısınız. Kaynak bağdaştırıcısını Fix Central ' den edinebilirsiniz (bkz. [“Kaynak bağdaştırıcısının Libertyçiine kurulması”](#) sayfa 423).

There are two flavors of Liberty Profile JVMs available in CICS 5.3 and later, the types of connection possible to IBM MQ are restricted as follows:

### **CICS Liberty Standart**

- IBM MQ kaynak bağdaştırıcısı, CLIENT kipinde IBM MQ ' un herhangi bir hizmet içi sürümüne bağlanabilir.

- The IBM MQ resource adapter can connect to any in-service version of IBM MQ for z/OS in BINDINGS mode when there is no CICS connection (active CICS MQCONN resource definition) to the same queue manager from the same CICS region.

### **CICS Liberty Tümleşik**

- IBM MQ kaynak bağdaştırıcısı, CLIENT kipinde IBM MQ ' un herhangi bir hizmet içi sürümüne bağlanabilir.
- Bağ tanımlama kipi bağlantısı desteklenmiyor.

Sisteminizin ayarlanmasını ve yapılandırılmasına 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 kullanılıyor**

Standards-based messaging support within an IMS 13 environment is provided through the use of IBM MQ classes for JMS.

İşletmelerinizin kullandığı IMS sistemi için sistem gereksinimlerini denetleyin. Ek bilgi için bkz. [IMS 13 için genel planlama bilgileri](#)

From IBM MQ 8.0.0 Fix Pack 4, IBM MQ supports using the IBM MQ classes for JMS in IMS version 13 and later.

This set of topics describes how to set up the IBM MQ classes for JMS in an IMS environment, and the API restrictions that apply when using the classic (JMS 1.1) and simplified (JMS 2.0) interfaces. API ' ye özgü bilgilerin bir listesi için bkz. [“JMS API kısıtlamaları” sayfa 320](#) .

**Not:** Benzer sınırlamalar kalıt (JMS 1.0.2) etki alanına özgü arabirimler için geçerlidir, ancak bunlar burada özel olarak tanımlanmaz.

### **Desteklenen IMS bağımlı bölgeleri**

Aşağıdaki bağımlı bölge tipleri desteklenmektedir:

- MPR
- BMP
- IFP
- JMP (31 bit Java sanal makineleri (JVM 'ler), 64 bit JVM' ler desteklenmez)
- JBP (yalnızca 31 bit JVM 'ler, 64 bit JVM ' ler desteklenmez)

Aşağıdaki konularda özellikle söz edilmedikçe, IBM MQ classes for JMS tüm bölge tiplerinde aynı davranışı gösterir.

### **Desteklenen Java Sanal Makineleri**

IBM MQ classes for JMS , Java Platform, Standard Edition 7 (Java SE 7) ya da sonraki bir sürümü gerektirir.

### **Diğer sınırlamalar**

The following restrictions apply when using IBM MQ classes for JMS in an IMS environment:

- İstemci kipi bağlantıları desteklenmez.
- Bağlantılar yalnızca IBM MQ ileti sistemi sağlayıcısı Normal ya da Version 8 kipi kullanılarak IBM MQ 8.0 kuyruk yöneticilerine desteklenir.

Bağlantı üreticisine ilişkin **PROVIDERVERSION** özneliği belirtilmemelidir ya da 7 'den büyük ya da bu değere eşit bir değer olmalıdır.

- XA bağlantısı üreticilerinin ( com.ibm.mq.jms.MQXAConnectionFactory gibi) kullanılması desteklenmez.

## İlgili görevler

[IBM MQ 'ı IMSolarak tanımlama](#)

### **Setting up the IMS adapter for use with IBM MQ classes for JMS**

IBM MQ classes for JMS , diğer programlama dilleri tarafından kullanılan IBM MQ-IMS bağdaştırıcılarını kullanır. Bu bağdaştırıcı, IMS External Subsystem Attach Facility (ESAF) olanağını kullanır.

## Başlamadan önce

Aşağıdaki yordamı tamamlamadan önce, ilgili kuyruk yöneticileri ve IMS denetim ve bağımlı bölgeleri için IMS bağdaştırıcısını [IMS bağdaştırıcısının ayarlanması](#) içinde açıklandığı şekilde yapılandırılmalısınız.



**Uyarı:** Diğer amaçlar için dinamik sınırlı kod öbeğine gereksinim duyarsanız, devingen sınırlı kod öbeği oluşturmayı tanımlayan adımı gerçekleştirmeniz gerekmez.

IMS bağdaştırıcısını yapılandırdıktan sonra, aşağıdaki yordamı gerçekleştirin.

## Yordam

1. Update the LIBPATH variable in the member of your IMS PROCLIB that is referenced by the ENVIRON parameter in your dependent region JCL (for example, DFSJVMEV) so that it includes the IBM MQ classes for JMS native libraries.

That is, the zFS directory that contains `libmqjims.so`. Örneğin, DFSJVMEV aşağıdaki gibi görünebilir; burada son satır, IBM MQ classes for JMS yerel kitaplıklarını içeren dizindir:

```
LIBPATH=>
/java/java71_31/J7.1/bin/j9vm:>
/java/java71_31/J7.1/bin:>
/ims13/dbdc/imsjava/classic/lib:>
/ims13/dbdc/imsjava/lib:>
/mqm/V8R0M0/java/lib
```

2. Add the IBM MQ classes for JMS to the class path of the JVM, used by your IMS dependent region, by updating the `java.class.path` option.

Bunu, [IMS PROCLIB veri kümesi için DFSJVMMS üyesi](#) içindeki yönergeleri izleyerek gerçekleştirin.

Örneğin, kalın harfinin güncellemeyi gösterdiği yeri aşağıdaki gibi kullanabilirsiniz:

```
-Djava.class.path=/ims13/dbdc/imsjava/imsutm.jar:/ims13/dbdc/imsjava/imsudb.jar:
/mqm/V8R0M0/java/lib/com.ibm.mq.allclient.jarLIBPATH_SUFFIX=MQ_INSTALLATION_PATH
```

**Not:** While there are many different jar files available in the directory containing the IBM MQ classes for JMS, you need only the `com.ibm.mq.allclient.jar` file.

3. IBM MQ classes for JMS' ın kullanılmasını sağlayacak IMS bağımlı bölgelerini durdurun ve yeniden başlatın.

## Sonraki adım

Bağlantı fabrikalarını ve hedeflerini oluşturun ve yapılandırın.

Bağlantı üreticilerinin ve hedeflerin IBM MQ uygulamalarını somutlaştırabilecek üç olası yaklaşım vardır. Ayrıntılar için bkz. "[IBM MQ classes for JMS uygulamasındaki bağlantı fabrikalarını ve hedeflerini oluşturma ve yapılandırma](#)" sayfa 184.

Bu üç yaklaşımın tümü IMS ortamında geçerli olduğunu unutmayın.

## İlgili görevler

[IMS bağdaştırıcısının ayarlanması](#)

[IBM MQ 'ı IMSolarak tanımlama](#)

## Hareket işleme davranışı

Messages sent and received by the IBM MQ classes for JMS in an IMS environment are always associated with the IMS unit of work (UOW) that is active on the current task.

That UOW can only be completed by calling the commit or rollback methods on an instance of the `com.ibm.ims.dli.tm.Transaction` object, or by the IMS task ending normally in which case the UOW is implicitly committed. IMS görevi olağan dışı sona ererse, UOW geriye işlenir.

Bunun sonucu olarak, **transacted** ve **acknowledgeMode** bağımsız değişkenlerinin değerleri, `Connection.createSession`ya da `ConnectionFactory.createContext` yöntemlerinden herhangi birini çağırırken yoksayıdır. Ayrıca, aşağıdaki yöntemler desteklenmez. Aşağıdaki yöntemlerden herhangi birini, oturum durumunda bir `IllegalStateException` 'de sonuçlansa da:

- `javax.jms.Session.commit()`
- `javax.jms.Session.recover()`
- `javax.jms.Session.rollback()`

JMS bağlam durumunda bir `IllegalStateException` ve::

- `javax.jms.JMSContext.commit()`
- `javax.jms.JMSContext.recover()`
- `javax.jms.JMSContext.rollback()`

Bu davranışın bir kural dışı durumu vardır. Bir oturum ya da JMS bağlamı aşağıdaki mekanizmalardan biri kullanılarak yaratılırsa:

- `Connection.createSession(false, Session.AUTO_ACKNOWLEDGE)`
- `Connection.createSession(Session.AUTO_ACKNOWLEDGE)`
- `ConnectionFactory.createContext(JMSContext.AUTO_ACKNOWLEDGE)`

Bu durumda, o oturumun ya da JMS bağlamının işleyişi aşağıdaki gibidir:

- Gönderilen iletiler, IMS UOW 'nin dışına gönderilir. Yani, bunlar hedef hedefinde hemen kullanılabilir ya da sağlanan teslim gecikmesi aralığı tamamlanınca kullanılabilir.
- Oturum ya da JMS bağlamını yaratan bağlantı üreticisine `SYNCPOINTALLGETS` özelliği belirtilmediği sürece, IMS UOW dışında, kalıcı olmayan iletiler alınmayacak.
- Kalıcı iletiler her zaman IMS UOW içinde alınır.

Örneğin, UOW geri yuvarlansa bile bir kuyruğa denetleme iletisi yazmak istediğinizde bu yararlı olabilir.

## IMS eşitleme noktalarının etkileri

IBM MQ classes for JMS , ESAF kullanımını sağlayan, var olan IBM MQ bağdaştırıcısı desteğine sahip olur. Bu, bir eşitleme noktası oluştuğunda IMS bağdaştırıcısı tarafından kapatılmakta olan tüm açık tutamaçlar da dahil olmak üzere, belgelenmiş davranışların geçerli olduğu anlamına gelir.

Ek bilgi için [“IMS uygulamalarında eşitleme noktaları” sayfa 66](#) başlıklı konuya bakın.

Bu noktayı göstermek için, bir JMP ortamında çalışan aşağıdaki kodu göz önünde bulundurun. The second call to `mp.send()` results in a `JMSException` as the `messageQueue.getUnique(inputMessage)` code results in all open IBM MQ connection and object handles being closed.

Similar behavior is observed if the `getUnique()` call was replaced with `Transaction.commit()`, but not if `Transaction.rollback()` was used.

```
//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 JMSEException 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 bağdaştırıcısını kullanırken dikkat edilmesi gereken noktalar***

Aşağıdaki kısıtlamalardan haberdar olmanız gerekir. Her kuyruk yöneticisi için tek bir bağlantı tanıtıcısına sahip olabilirsiniz. Hem JMS , hem de yerel kod kullanıldığında, IBM MQ ile etkileşimde etkileri vardır. Bağlantı kimlik doğrulaması ve yetkilendirmesine ilişkin sınırlamalar vardır.

### **Her kuyruk yöneticisi için bir bağlantı tanıtıcısı**

IMS bağımlı bölgelerde, belirli bir kuyruk yöneticisine belirli bir kuyruk yöneticisine yalnızca bir bağlantı tanıtıcısı kullanılabilir. Aynı kuyruk yöneticisine bağlanma girişimleri, var olan tanıtıcıyı yeniden kullanır.

Bu davranış, yalnızca IBM MQ classes for JMSkullanan bir uygulamada sorun yaratmamalı, bu davranış IBM MQ classes for JMS ile etkileşimde bulunan uygulamalarda, COBOL ya da C gibi dillerde yazılmış yerel kodda MQI ' yı kullanırken, IBM MQile etkileşimde bulunan uygulamalarda sorunlara neden olabilir.

### **Hem JMS , hem de yerel kod kullanıldığında IBM MQ ile etkileşimde bulunulması**

Yerel ya da Java kodu bırakılmadan önce hem IBM MQ işlevselliğini kullanan, hem de IBM MQ bağlantısının kapatılmadığında Java kodu ve yerel kod bırakıldığında sorunlar ortaya çıkabilir.

Örneğin, aşağıdaki sözde koda, kuyruk yöneticisine yönelik bir bağlantı tanıtıcısı başlangıçta IBM MQ classes for JMS kullanılarak Java kodunda kurulur. Bağlantı tanıtıcısı, COBOL kodunda yeniden kullanılır ve MQDISC çağırısı tarafından geçersiz kılınmaktadır.

IBM MQ classes for JMS bağlantısının bir sonraki kullanımında, bir MQRC\_HCONN\_ERROR neden koduna sahip bir JMSEnception bağlantı tanıtıcısını kullanın.

```
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 sonuçlanabilen diğer benzer kullanım örüntüleri de vardır.

Yerel ve Java kodu arasındaki IBM MQ bağlantı tanıtıcılarını paylaşmak mümkün olsa da (örneğin, bir önceki örnek, bir MQDISC çağırısı olmasa işe yarayacaktı), en iyi uygulama Java 'dan yerel kodla ya da diğer tarafa çevrilmeden önce bağlantı tanıtıcılarını kapatmanın en iyi uygulamasıdır.

## Bağlantı kimlik doğrulaması ve yetkilendirmesi

JMS belirtimi, bir bağlantı ya da JMS bağlam nesnesi oluştururken kimlik doğrulama ve yetkilendirme için bir kullanıcı adının ve parolanın belirtilmesine olanak sağlar.

Bu, IMS ortamında desteklenmez. Atılmakta olan bir JMS Exception içinde kullanıcı adı ve parola sonuçları belirtilirken bağlantı yaratma girişiminde bulunuluyor. Attempting to create a JMS context, while specifying a user name and password, results in a JMSEnception being thrown.

Bunun yerine, bir IMS ortamından IBM MQ ' e bağlanırken kimlik doğrulama ve yetkilendirme için mevcut mekanizmalar kullanılmalıdır.

Daha fazla bilgi için z/OS üzerinde güvenliğin ayarlanması başlıklı konuya bakın. Özellikle, kullanılacak kullanıcı kimliklerinin açıklandığı [Güvenlik denetimi için kullanıcı kimlikleri belgesine](#) bakın.

### İlgili görevler

[z/OS üzerinde güvenliğin ayarlanması](#)

### JMS API kısıtlamaları

From a JMS specification perspective, the IBM MQ classes for JMS treat IMS as a Java EE compliant application server, that always has a JTA transaction in progress.

Örneğin, JMS belirtimi bir JEE EJB 'de ya da Web kapsayıcısında arayamayabileceğiniz, bir JTA işlemi devam ederken javax.jms.Session.commit() 'ı hiçbir zaman IMS' de arayamaz olabilirsiniz.

Bu, “Hareket işleme davranışı” sayfa 318'ta tanımlananlara ek olarak, JMS API' ya yönelik aşağıdaki kısıtlamalarla sonuçlanır.

### Klasik API kısıtlamaları

- javax.jms.Connection.createConnectionConsumer(javax.jms.Destination, String, javax.jms.ServerSessionPool, int) her zaman bir JMSEnception atar.
- javax.jms.Connection.createDurableConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int) her zaman bir JMSEnception atar.
- Bağlantı önceden var olan bir oturuma sahipse, javax.jms.Connection.createSession 'un tüm üç değişkeni her zaman bir JMSEnception 'yi atar.



- `javax.jms.Connection.createSharedConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` her zaman bir `JMSEException`atar.
- `javax.jms.Connection.createSharedDurableConnectionConsumer(javax.jms.Topic, String, String, javax.jms.ServerSessionPool, int)` her zaman bir `JMSEException`atar.
- `javax.jms.Connection.setClientID()` her zaman bir `JMSEException`atar.
- `javax.jms.Connection.setExceptionListener(javax.jms.ExceptionListener)` her zaman bir `JMSEException`atar.
- `javax.jms.Connection.stop()` her zaman bir `JMSEException`atar.
- `javax.jms.MessageConsumer.setMessageListener(javax.jms.MessageListener)` her zaman bir `JMSEException`atar.
- `javax.jms.MessageConsumer.getMessageListener()` her zaman bir `JMSEException`atar.
- `javax.jms.MessageProducer.send(javax.jms.Destination, javax.jms.Message, javax.jms.CompletionListener)` her zaman bir `JMSEException`atar.
- `javax.jms.MessageProducer.send(javax.jms.Destination, javax.jms.Message, int, int, long, javax.jms.CompletionListener)` her zaman bir `JMSEException`atar.
- `javax.jms.MessageProducer.send(javax.jms.Message, int, int, long, javax.jms.CompletionListener)` her zaman bir `JMSEException`atar.
- `javax.jms.MessageProducer.send(javax.jms.Message, javax.jms.CompletionListener)` her zaman bir `JMSEException`atar.
- `javax.jms.Session.run()` her zaman bir `JMSRuntimeException`atar.
- `javax.jms.Session.setMessageListener(javax.jms.MessageListener)` her zaman bir `JMSEException`atar.
- `javax.jms.Session.getMessageListener()` her zaman bir `JMSEException`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

Java ortamında IBM MQ kullanın. IBM MQ classes for Java allow a Java application to connect to IBM MQ as an IBM MQ client, or connect directly to an IBM MQ queue manager.

### Not:

IBM , IBM MQ classes for Java 'de başka bir geliştirme yapmayacaktır ve bunlar, IBM MQ 8.0' ta teslim edilen düzeyde işlevsel olarak sabitlenirler. IBM MQ classes for Java ' u kullanan var olan uygulamalar tam olarak desteklenmeye devam eder, ancak yeni özellikler eklenmez ve geliştirmeler için istek reddedilir. Tam olarak desteklenen bir şekilde, hataların IBM MQ Sistem Gereksinimleri üzerindeki değişikliklerle birlikte belirlendiği değişiklikler ile birlikte düzeltileceği anlamına gelir.

IBM MQ classes for Java , IMSiçinde desteklenmiyor.

IBM MQ classes for Java , WebSphere Libertyiçinde desteklenmiyor. 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 iki alternatif API ' den biridir. Diğer API: IBM MQ classes for JMS.

IBM MQ 8.0' dan IBM MQ classes for Java , Java 7 ile oluşturulmuştur.

Java 7 yürütme ortamı, daha önceki sınıf dosyası sürümlerinin çalıştırılmasını destekler.

IBM MQ classes for Java encapsulate the Message Queue Interface (MQI), the native IBM MQ API and use a similar object model to the C++ and .NET interfaces to IBM MQ.

Programlanabilir seçenekler, IBM MQ classes for Java 'in aşağıdaki yöntemlerden birini kullanarak IBM MQ ' a bağlanmasını sağlar:

- In istemci kipi as an IBM MQ MQI client by using Transmission Control Protocol/Internet Protocol (TCP/IP)
- Java Yerel Arabirimi 'ni (JNI) kullanarak doğrudan IBM MQ ' a bağlanmak için [bağ tanımları kipinde](#)

**Not:** Otomatik istemci yeniden bağlantısı IBM MQ classes for Javatarafından desteklenmiyor.

## İstemci kipi bağlantısı

Bir IBM MQ classes for Java uygulaması, istemci kipini kullanarak desteklenen herhangi bir kuyruk yöneticisine bağlanabilir.

İstemci kipinde bir kuyruk yöneticisine bağlanmak için, bir IBM MQ classes for Java uygulaması kuyruk yöneticisinin çalıştığı sistemde ya da farklı bir sistemde çalıştırılabilir. Her durumda IBM MQ classes for Java , TCP/IP üzerinden kuyruk yöneticisine bağlanır.

İstemci kipi bağlantılarını kullanmak üzere uygulamaların nasıl yazılabilmesiyle ilgili daha fazla bilgi için bkz. [“IBM MQ classes for Java bağlantı kipleri” sayfa 345.](#)

## Bağ tanımları kipi bağlantısı

When used in bindings mode, IBM MQ classes for Java uses the Java Native Interface (JNI) to call directly into the existing queue manager API, rather than communicating through a network. In most environments, connecting in bindings mode provides better performance for IBM MQ classes for Java applications than connecting in client mode, by avoiding the cost of TCP/IP communication.

Bağ tanımları kipinde bağlanmak için IBM MQ classes for Java komutunu kullanan uygulamaların, bağlantı kurdukları 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 üzere yapılandırılmalıdır; ek bilgi için [“IBM MQ classes for Java Kitaplıklar” sayfa 331 ' e bakın.](#)

Bağ tanımları kipi bağlantılarını kullanmak üzere uygulamaların nasıl yazılabilmesiyle ilgili daha fazla bilgi için bkz. [“IBM MQ classes for Java bağlantı kipleri” sayfa 345.](#)

## İlgili kavramlar

[IBM MQ Java dil arabirimleri](#)

[“kullanmaIBM MQ classes for JMS” sayfa 77](#)

IBM MQ classes for Java Message Service (IBM MQ classes for JMS), IBM MQ ile birlikte verilen JMS sağlayıcısıdır. javax.jms paketinde tanımlanan arabirimlerin yanı sıra, IBM MQ classes for JMS , JMS API ' ye iki uzantı kümesi sağlar.

## İlgili görevler

[IBM MQ classes for Java uygulamalarının izlenmesi](#)

[Java ve JMS sorun giderme](#)

## 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 ' yi kullanabilir.

**Not:** IBM MQ classes for Java ' ı kullanan var olan uygulamalar tam olarak desteklenmeye devam etse de, yeni uygulamalar IBM MQ classes for JMS' i kullanılmalıdır. Features that have recently been added to IBM

MQ, such as asynchronous consume and automatic reconnection, are not available in the IBM MQ classes for Java, but are available in the IBM MQ classes for JMS. Daha fazla bilgi için bkz “[Neden IBM MQ classes for JMS kullanmalıyım?](#)” sayfa 78.

**Not:** IBM MQ classes for Java , IBM MQ 8.0' ta teslim edilen düzeyde işlevsel olarak sabitlenmektedir. IBM MQ classes for Java ' u kullanan var olan uygulamalar tam olarak desteklenmeye devam eder, ancak bu API dengelenir, bu nedenle yeni özellikler eklenmez ve geliştirmeler reddedilmeye yönelik istekler için istekte bulunmaz. Tam olarak desteklenen bir şekilde, hataların IBM MQ Sistem Gereksinimleri üzerindeki değişikliklerle birlikte belirlendiği değişiklikler ile birlikte düzeltileceği anlamına gelir.



## IBM MQ classes for Java için önkoşullar

IBM MQ classes for Java' u kullanmak için bazı diğer yazılım ürünlerine gereksininiz vardır.

IBM MQ classes for Java ile ilgili önkoşullarla ilgili bilgi için [IBM MQ](#) web sayfasına bakın.

IBM MQ classes for Java uygulamalarını geliştirmek için bir Java Geliştirme Takımı 'na (JDK) gereksinim duyarsınız. İşletim sisteminizle desteklenen JDKS ' lerin ayrıntıları [IBM MQ](#) bilgilerinde bulunabilir.

IBM MQ classes for Java uygulamalarını çalıştırmak için aşağıdaki yazılım bileşenlerine gereksinim duyarsınız:

- Bir kuyruk yöneticisine bağlanan uygulamalar için bir IBM MQ kuyruk yöneticisi
- Uygulamaları çalıştırdığınız her sistem için bir Java Runtime Environment (JRE). IBM MQ ile birlikte uygun bir JRE sağlanır.
-  IBM i için, işletim sisteminin 30. seçeneği olan QShell
-  z/OS için, UNIX and Linux System Services (USS)

FIPS 140-2 sertifikalı şifreleme modülleri kullanmak için TLS bağlantılarına gereksinim duyuyorsanız, IBM Java JSSE FIPS sağlayıcısına (IBMJSSEFIPS) gerek duyarsınız. Sürüm 1.4.2 ya da sonraki sürümlerde bulunan her IBM JDK ve JRE, IBMJSSEFIPS ' leri içerir.

You can use Internet Protocol Version 6 (IPv6) addresses in your IBM MQ classes for Java applications. IPv6 ise supported by your Java virtual machine (JVM) and the TCP/IP implementation on your operating system.

## Running IBM MQ classes for Java applications within Java EE

Java EE' ta IBM MQ classes for Java kullanılmadan önce dikkate alınması gereken belirli kısıtlamalar ve tasarım konuları vardır.

IBM MQ classes for Java , bir Java Platform, Enterprise Edition (Java EE) ortamında kullanıldığında kısıtlamalara sahiptir. Bir Java EE ortamının içinde çalışan bir IBM MQ classes for Java uygulamasını tasarlarken, gerçekleştirirken ve yönetirken dikkate alınması gereken ek noktalar da vardır. Bu kısıtlamalar ve dikkat edilmesi gereken noktalar aşağıdaki bölümlerde açıklanmaktadır.

### JTA hareketleri kısıtlamaları

The only supported transaction manager for applications using IBM MQ classes for Java is IBM MQ itself. Although an application under JTA control can make use IBM MQ classes for Java, any work performed through these classes is not controlled by the JTA units of work. Bunun yerine, JTA arabirimleri aracılığıyla uygulama sunucusu tarafından yönetilenlerden ayrı olarak yerel iş birimleri oluştururlar. Özellikle, JTA hareketinin herhangi bir geriye işlenmesi gönderilen ya da alınan iletilerin geri alınmasına neden olmaz. Bu kısıtlama, uygulama ya da bean tarafından yönetilen hareketler ve taşıyıcı tarafından yönetilen işlemler ve tüm Java EE kapsayıcıları için geçerlidir. İletim alışverişi çalışmalarını, uygulama sunucusu eşgüdümü işlemler içinde IBM MQ ile doğrudan gerçekleştirmek için, bunun yerine IBM MQ classes for JMS kullanılmalıdır.

## İş parçacığı yaratma

IBM MQ classes for Java , çeşitli işlemler için iç iş parçacıkları yaratır. Örneğin, bir yerel kuyruk yöneticisinde doğrudan çağırmak için BAĞLAMALAR kipinde çalışırken, çağrılar IBM MQ classes for Javatarafından dahili olarak oluşturulan bir 'işçi' iş parçacığında yapılır. Örneğin, bir bağlantı havuzundan kullanılmayan bağlantıları temizlemek ya da sonlandırılmış yayınlama/abone olma uygulamalarına ilişkin abonelikleri kaldırmak için diğer iş parçacıkları içeride yaratılabilir.

Bazı Java EE uygulamaları (örneğin, EJB ve Web kapsayıcılarında çalıştırılanlar) yeni iş parçacıkları yaratmamalıdır. Bunun yerine, uygulama sunucusu tarafından yönetilen ana uygulama iş parçacıklarına ilişkin tüm işler gerçekleştirilmelidir. Uygulamalar IBM MQ classes for Java' u kullandığında, uygulama sunucusu uygulama kodunu ve IBM MQ classes for Java kodunu ayırt edemeyebilir; bu nedenle, daha önce tanımlanan iş parçacıkları uygulamanın kapsayıcı belirtimiyle uyumlu olmasını sağlar. IBM MQ classes for JMS , bu Java EE belirtilmelerini bozmaz ve bunun yerine kullanılabilir.

## Güvenlik kısıtlamaları

Security policies implemented by an application server might prevent certain operations that are undertaken by the IBM MQ classes for Java API, such as creating and operating new threads of control (as described in the preceding sections).

Örneğin, uygulama sunucuları genellikle varsayılan olarak Java Security tarafından devre dışı bırakılır ve uygulama sunucusuna özgü bazı yapılandırma yoluyla etkinleştirilmesine izin verir (bazı uygulama sunucuları aynı zamanda Java Security içinde kullanılan ilkelerin daha ayrıntılı bir şekilde yapılandırılmasına olanak tanır). Java Güvenliği etkinleştirildiğinde, IBM MQ classes for Java , uygulama sunucusu için tanımlanan Java Güvenlik ilkesi okuma kurallarını bozabilir ve API, çalışabilmesi için gereksinim duyduğu tüm iş parçacıklarını oluşturamayabilir. İş parçacığı yönetimiyle ilgili sorunları önlemek için, Java güvenliğinin etkinleştirildiği ortamlarda IBM MQ classes for Java kullanımı desteklenmez.

## Uygulama yalıtma konuları

Java EE ortamında uygulamaların çalıştırılması amaçlanan bir avantaj, uygulama yalıtımsıdır. The design and implementation of IBM MQ classes for Java predate the Java EE environment. IBM MQ classes for Java , uygulama yalıtma kavramını desteklemeyen bir şekilde kullanılabilir. Bu alandaki dikkat edilecek noktalara ilişkin örnekler arasında şunlar yer alır:

- MQEnvironment sınıfındaki statik (JVM process-wide) ayarlarının kullanımı; örneğin:
  - bağlantı kimliği 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 ilişkin MQEnvironment özelliklerinin değiştirilmesi, aynı özellikleri kullanan diğer uygulamaları da etkiler. Java EEgibi çok sayıda uygulama ortamında çalışırken, her bir uygulamanın, süreç genelindeki MQEnvironment sınıfında yapılandırılan özellikleri varsayılan olarak kullanmak yerine, belirli bir özellik kümesiyle MQQueueManager nesnelerini yaratma yoluyla kendi ayrı yapılandırmasını kullanması gerekir.

- MQEnvironment sınıfı, aynı JVM işlemi içinde IBM MQ classes for Java kullanan tüm uygulamalarda genel olarak hareket eden bir dizi durağan yöntem sunar ve belirli uygulamalar için bu davranışı geçersiz kılmanın yolu yoktur. Örnekler:
  - Anahtar deposunun yeri gibi TLS özelliklerini yapılandırma
  - istemci kanal çıkışlarını yapılandırma
  - tanılama izlemesini geçerli ya da geçersiz kılma
  - Kuyruk yöneticilerine yönelik bağlantıların kullanımını eniyilemek için kullanılan varsayılan bağlantı havuzunu yönetme

Bu tür yöntemlerin çağrılması, aynı Java EE ortamında çalışan tüm uygulamaları etkiler.

- Bağlantı havuzlama, aynı kuyruk yöneticisine birden çok bağlantı oluşturma işlemini eniyilemek için etkinleştirilir. Varsayılan bağlantı havuzu yöneticisi, süreç genelinde ve birden çok uygulama tarafından paylaşılır. Changes to connection pool configuration, such as replacing the default connection manager for one application using the MQEnvironment.setDefaultConnectionManager() method therefore affects other applications running in the same Java EE application server.
- TLS, MQEnvironment sınıfı ve MQQueueManager nesne özellikleri kullanılarak IBM MQ classes for Java kullanan uygulamalar için yapılandırılıyor. Uygulama sunucusunun yönetilen güvenlik yapılandırmasıyla bütünleştirilmedi. IBM MQ classes for Java ' u uygun güvenlik düzeyinizi sağlamak için uygun bir şekilde yapılandırdığınızdan ve uygulama sunucusu yapılandırmasını kullanmadığınızdan emin olmanız gerekir.

## Bağ tanımları kipi kısıtlamaları

IBM MQ and WebSphere Application Server can be installed on the same machine such that the major versions of the queue manager and of the IBM MQ resource adapter (RA) shipped in WebSphere Application Server are different. Bir IBM MQ RA düzeyi 7.0.1olan WebSphere Application Server 7.0yönetim ortamı için, IBM WebSphere MQ 6 kuyruk yöneticisiyle aynı makineye kurulabilir.

Kuyruk yöneticisi ve kaynak bağdaştırıcısı ana sürümleri farklıysa, bağ tanımları bağlantıları kullanılamaz. Kaynak bağdaştırıcısı kullanılarak, WebSphere Application Server kaynağından kuyruk yöneticisine yönelik tüm bağlantılar istemci tipi bağlantılarını kullanmalıdır. Sürümler aynıysa, bağ tanımları bağlantıları kullanılabilir.

## IBM MQ classes for Java' da karakter dizgisi dönüştürmeleri

IBM MQ classes for Java , doğrudan karakter dizesi dönüştürmesi için CharSetEncoders ve CharSetDecoders ' ı doğrudan kullanır. Karakter dizesi dönüştürmesi için varsayılan davranış, iki sistem özelliği ile yapılandırılabilir. Eşlenebilir karakterler içeren iletilerin işlenmesi, com.ibm.mq.MQMDaracılığıyla 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 gerçekleştirilmiştir.

Bu yöntemlerin herhangi birini kullanarak, bozuk biçimli ya da çevrilemeyen verilerin varsayılan yerine ( REPLACE) ilişkin sonuçlar elde edilebilir. This behavior can obscure errors in applications, and lead to unexpected characters, for example ?, in translated data.

IBM MQ 8.0'tan bu tür sorunları daha önce ve daha etkin bir şekilde algılamak için IBM MQ classes for Java , CharSetEncoders ve CharSetDecoders ' yi doğrudan kullanır ve yanlış biçimli ve çevrilemeyen verilerin işlenmesini açık bir şekilde yapılandırır. Varsayılan davranış, uygun bir MQException yayınlarak REPORT ' e ait olur.

## Yapılandırılıyor

Translating from UTF-16 (the character representation used in Java) to a native character set, such as UTF-8, is termed encoding, while translating in the opposite direction is termed decoding.

Şu anda kod çözme işlemi, CharSetDecoders için varsayılan davranışı, kural dışı durum yayınlamak hata bildirme davranışını ele alır.

Bir ayar, hem kodlamada hem de kod çözülürken hata işlenmesini denetlemek üzere bir java.nio.charset.CodingErrorAction belirlemek için kullanılır. Diğer bir ayar, kodlamayı kodlarken, yeni baytı ya da byte 'ları denetlemek için kullanılır. Kod çözme işlemlerinde varsayılan Java yerine koyma dizgisi kullanılacak.

## Configuration of untranslatable data handling in IBM MQ classes for Java

IBM MQ 8.0' tan com.ibm.mq.MQMD , aşağıdaki iki alanı içerir:

## **byte [] unMappableDeğişirme**

Bir giriş karakteri çevrelemeyse, kodlanmış bir dizgiye yazılacak byte sırası ve REPLACE seçeneğini belirtmiş olmanız gerekir.

### **Varsayılan: "?" .getBytes()**

Varsayılan Java yerine koyma dizgisi, kodu çözme işlemlerinde kullanılır.

## **java.nio.charset.CodingErrorAction unmappableAction**

Kodlama ve kod çözme ile ilgili çevrelemeyen veriler için yapılacak işlemi belirtir:

**Varsayılan değer: CodingErrorAction.REPORT;**

## **Sistem varsayılanlarını ayarlamak için sistem özellikleri**

IBM MQ 8.0' tan, karakter dizgisi dönüştürmeyle ilgili varsayılan davranışı yapılandırmak için aşağıdaki iki Java sistem özelliği kullanılabilir.

### **com.ibm.mq.cfg.jmqi.UnmappableCharacterAction**

Kodlama ve kod çözme işlemleri için çevrelemeyen veriler için yapılacak işlemi belirtir. Değer REPORT, REPLACE ya da IGNORE olabilir.

### **com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement**

Bir kodlama işleminde bir karakter eşlenemediğinde uygulanacak ikame byte'ları ayarlar ya da alır. Varsayılan Java yerine koyma dizgisi, kod çözme işlemlerinde kullanılır.

Java karakteri ile yerel byte gösterimleri arasındaki karışıklığı önlemek için, yerel karakter kümesindeki yerine koyma byte 'ı gösteren ondalık sayı olarak `com.ibm.mq.cfg.jmqi.UnmappableCharacterReplacement` değerini belirtmeniz gerekir.

For example, the decimal value of ?, as a native byte, is 63 if the native character set is ASCII-based, such as ISO-8859-1, while it is 111 if the native character set is EBCDIC.

**Not:** Note that if an MQMD or MQMessage object has either the **unmappableAction** or **unMappableReplacement** fields set, then the values of these fields take precedence over the Java system properties. Bu, Java sistem özellikleri tarafından belirtilen değerlerin, gerekirse her ileti için geçersiz kılınabilmesini sağlar.

## **İlgili kavramlar**

"IBM MQ classes for JMS' da karakter dizgisi dönüştürmeleri" sayfa 123

IBM MQ classes for JMS , doğrudan karakter dizesi dönüştürmesi için CharsetEncoders ve CharsetDecoders ' ı doğrudan kullanır. Karakter dizesi dönüştürmesi için varsayılan davranış, iki sistem özelliği ile yapılandırılabilir. Eşlenebilir karakterler içeren iletilerin işlenmesi, UnmappableCharacter işlemi ve yerine koyma byte 'ları ayarlanmak üzere ileti özellikleri kullanılarak yapılandırılabilir.



## **Kurma ve Yapılandırma IBM MQ classes for Java**

This section describes the directories and files that are created when you install IBM MQ classes for Java, and tells you how to configure IBM MQ classes for Java after installation.

### **IBM MQ classes for Java için kurulu olan**

The latest version of IBM MQ classes for Java is installed with IBM MQ. Bunun yapıldığından emin olmak için varsayılan kuruluş seçeneklerini geçersiz kılmanız gerekebilir.

IBM MQ kuruluşuna ilişkin daha fazla bilgi için bkz.

-  [IBM MQ ürününü kurma](#)
-  [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 yer alır.

Programlanabilir Komut Biçimi (PCF) gibi standart ileti üstbilgileri için destek, `com.ibm.mq.headers.jar` adlı JAR dosyasında bulunur.

Programlanabilir Komut Biçimi (PCF) desteği, `com.ibm.mq.pcf.jar` JAR dosyasında bulunur.

**Not:** IBM MQ classes for Java , bir uygulama sunucusu içinde kullanılması önerilmez. Bu ortamda çalıştırılırken geçerli olan kısıtlamalara ilişkin bilgi için bkz. “Running IBM MQ classes for Java applications within Java EE” sayfa 323. Daha fazla bilgi için [J2EE/JEE Ortamlarında WebSphere MQ Java Arabirimlerini Kullanmabaşlıklı konuyu](#) bakın.

**Önemli:** “IBM MQ classes for Java yeniden yerelleşme JAR dosyaları” sayfa 327’inde açıklanan yeniden yerelleşme JAR dosyaları dışında, IBM MQ classes for Java JAR dosyalarının ya da yerel kitaplıkların diğer makinelere ya da IBM MQ classes for Java ' in kurulu olduğu bir makinede farklı bir konuma kopyalanması desteklenmez.

#### *IBM MQ classes for Java yeniden yerelleşme JAR dosyaları*

Yeniden yerelleşme JAR dosyaları, IBM MQ classes for Java' u çalıştırmak için gereken sistemlere taşınabilir.

#### **Önemli:**

- Yeniden yereldeki JAR dosyaları içinde açıklanan yeniden yerelleşme JAR dosyaları dışında, IBM MQ classes for Java JAR dosyalarının ya da yerel kitaplıkların diğer makinelere ya da IBM MQ classes for Java ' in kurulu olduğu bir makineden farklı bir konuma kopyalanması desteklenmez.
- Sınıf yükleyici çakışmalarını önlemek için, yeniden yerelleşme JAR dosyalarının aynı Java yürütme ortamı içindeki birden çok uygulama içinde paketlemek için önerilmez. Bu senaryoda, IBM MQ yeniden yerelleştirilebilir JAR dosyalarının Java yürütme ortamının sınıf yolunda (classpath) kullanılabilir olduğunu göz önünde bulundurun.
- relocatable gibi, Java EE uygulama sunucularına konuşlandırılan uygulamalar içindeki yeniden yerelleşme JAR dosyalarını eklemeyin; örneğin, WebSphere Application Server. Bu ortamlarda, IBM MQ kaynak bağdaştırıcısı IBM MQ classes for Java ile birlikte konuşlandırılmalı ve kullanılmamalıdır. Note that WebSphere Application Server embeds the IBM MQ resource adapter, so there is no need to deploy it manually into this environment. Buna ek olarak, IBM MQ classes for Java , WebSphere Liberty’inde desteklenmez. Daha fazla bilgi için bkz. “Liberty ve IBM MQ kaynak bağdaştırıcısı” sayfa 420.
- Uygulamalarınız içindeki yeniden yerelleşme JAR dosyalarını paketliyorsanız, tüm önkoşul JAR dosyalarını Yeniden yerelleşme JAR dosyalarıyla başlıklı konuda açıklandığı gibi içerdiğinizden emin olun. You should also ensure that you have appropriate procedures to update the bundled JAR files as part of application maintenance, to ensure that the IBM MQ classes for Java remain current and known issues are re-mediated.

## **Yereldeki JAR dosyaları yeniden yerlesin**

Within an enterprise, the following files can be moved to systems that need to run IBM MQ classes for Java applications:

- `com.ibm.mq.allclient.jar`
- `com.ibm.mq.traceControl.jar`
- `bcpkix-jdk15on.jar`
- `bcprov-jdk15on.jar`
- `V9.1.0.9` `bcutil-jdk15on.jar`
- `V9.1.2` `org.json.jar`

Bouncy Castle güvenlik sağlayıcısı ve CMS desteği JAR dosyaları gereklidir. Daha fazla bilgi için bkz. [AMS ile IBM dışı JRE ' ler için destek](#). Aşağıdaki JAR dosyaları gereklidir:

- `bcpkix-jdk15on.jar`
- `bcprov-jdk15on.jar`
- `V9.1.0.9` `bcutil-jdk15on.jar`

## V 9.1.2

IBM MQ classes for Java uygulamanız JSON biçiminde bir CCDT kullanıyorsa, `org.json.jar` dosyası gereklidir.

`com.ibm.mq.allclient.jar` dosyası, IBM MQ classes for JMS, IBM MQ classes for Javave PCF ve Üstbilgiler Sınıflarını içerir. Bu dosyayı yeni bir konuma taşırsanız, bu yeni konumu yeni IBM MQ Fix Packs ile alıkoymak için gerekli adımları gerçekleştirdiğinizden emin olun. Ayrıca, geçici bir düzeltme alıyorsanız, bu dosyanın kullanılmasının IBM Desteği tarafından bilinen bir şekilde sağlandığından emin olun.

`com.ibm.mq.allclient.jar` dosyasının sürümünü belirlemek için şu komutu kullanın:

```
java -jar com.ibm.mq.allclient.jar
```

Aşağıdaki örnek, bu komuttan bazı örnek çıktıları göstermektedir:

```
C:\Program Files\IBM\MQ_1\java\lib>java -jar com.ibm.mq.allclient.jar
Name:      Java Message Service Client
Version:   9.1.0.0
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:   9.1.0.0
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:   9.1.0.0
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:   9.1.0.0
Level:     p000-L140428.1
Build Type: Production
Location:  file:/C: /Program Files/IBM/MQ_1/java/lib/com.ibm.mq.allclient.jar
```





`com.ibm.mq.traceControl.jar` dosyası, IBM MQ classes for JMS uygulamalarına ilişkin izlemeyi devingen olarak denetlemek için kullanılır. Daha fazla bilgi için bkz. [Java için IBM MQ sınıflarını ve JMS için IBM MQ sınıfları](#) kullanarak çalışan bir işlemdeki izlemeyi denetleme.

### 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 platforma göre değişir.


## IBM MQ classes for Java dosyaları için kuruluş dizinleri

Çizelge 48 sayfa 328 , IBM MQ classes for Java dosyalarının kurulu olduğu yeri gösterir.

Çizelge 48. IBM MQ classes for Java kuruluş dizinleri	
Platform	Dizin
 AIX	<code>MQ_INSTALLATION_PATH/java/lib</code>
	<code>/QIBM/ProdData/mqm/java/lib</code>
 Linux	<code>MQ_INSTALLATION_PATH/java/lib</code>
 Solaris	<code>MQ_INSTALLATION_PATH/java/lib</code>
 Windows	<code>MQ_INSTALLATION_PATH\java\lib</code>



Çizelge 48. IBM MQ classes for Java kuruluş dizinleri (devamı var)







Platform	Dizin
 z/OS	MQ_INSTALLATION_PATH/mqm/V8R0M0/java /lib

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

### Örnekler için kuruluş dizinleri

Some sample applications, such as the Installation Verification Programs (IVPs), are supplied with IBM MQ. Çizelge 49 sayfa 329 , örnek uygulamaların kurulu olduğu yeri gösterir. IBM MQ classes for Java örnekleri, wmqjavaadlı bir altdizinde yer alıyor. PCF örnekleri, pcfadlı bir altdizinde yer alıyor.

Çizelge 49. Örnek dizinleri







Platform	Dizin
 AIX	MQ_INSTALLATION_PATH/samp/wmqjava/
 IBM i	/QIBM/ProdData/mqm/java/samples
 Linux	MQ_INSTALLATION_PATH/samp/wmqjava/
 Solaris	MQ_INSTALLATION_PATH/samp/wmqjava/
 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 temsil eder.

### JRE için kuruluş dizinleri

IBM MQ classes for JMS , Java Runtime Environment (JRE) için bir Java 7 (ya da üstü) gerektirir. IBM MQ ile birlikte uygun bir JRE kurulur. Çizelge 50 sayfa 329 , bu JRE ' nin kurulu olduğu yeri gösterir. To run Java programs such as the provided samples, using this JRE, either explicitly invoke JRE\_LOCATION/bin/java or add JRE\_LOCATION/bin to the PATH environment (or equivalent) for your platform, where JRE\_LOCATION is the directory given in Çizelge 50 sayfa 329.

Çizelge 50. JRE dizinleri

Platform	Dizin
 AIX	MQ_INSTALLATION_PATH/java/jre
 IBM i	/QIBM/ProdData/mqm/java/jre
 Linux	MQ_INSTALLATION_PATH/java/jre
 Solaris	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 temsil eder.







### IBM MQ classes for Java ile ilgili ortam deęişkenleri

IBM MQ classes for Java uygulamalarını çalıştırmak istiyorsanız, bunların sınıf yolu IBM MQ classes for Java ve örnek dizinlerini içermelidir.

IBM MQ classes for Java için, çalıştırılacak uygulamalar (sınıf yolu) uygun IBM MQ classes for Java dizinini içermelidir. Örnek uygulamaları çalıştırmak için, sınıf yolunun uygun örnek dizinlerini de içermesi gerekir. Bu bilgi, Java başlatma komutunda ya da CLASSPATH ortam deęişkeninde sağlanabilir.

**Önemli:** -Xbootclasspathdli Java seçeneğinin ayarlanması, IBM MQ classes for Java' un desteklenmemesi.

Çizelge 51 sayfa 330 , örnek uygulamalar da içinde olmak üzere, IBM MQ classes for Java uygulamalarını çalıştırmak için her platformda kullanılacak uygun CLASSPATH ayarını gösterir.

Platform	SPATH ayarı
 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:
 Linux	CLASSPATH= MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.jar: MQ_INSTALLATION_PATH/samp/wmqjava/samples:
 Solaris	CLASSPATH= MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.jar: MQ_INSTALLATION_PATH/samp/wmqjava/samples:
 Windows	CLASSPATH= MQ_INSTALLATION_PATH\Java\lib\com.ibm.mq.jar; MQ_INSTALLATION_PATH\tools\wmqjava\samples;
 z/OS	CLASSPATH= MQ_INSTALLATION_PATH/mqm/V9R1M0/java/lib/com.ibm.mq.jar: MQ_INSTALLATION_PATH/mqm/V9R1M0/java/samples/wmqjava: MQ_INSTALLATION_PATH/mqm/V9R1M0/java/samples/pcf

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduęu üst düzey dizini temsil eder.

-Xlint seçeneğini kullanarak derleyip derlendiyse, com.ibm.mq.es.e.jar dosyasının yok olduğunu bildiren bir ileti görebilirsiniz. Uyarıyı yoksayabilirsiniz. Bu dosya yalnızca Advanced Message Security' u yüklediyseniz kullanılabilir.

IBM MQ classes for JMS ile verilen komut dosyaları aşağıdaki ortam deęişkenlerini kullanır:

#### MQ\_JAVA\_DATA\_PATH


Bu ortam deęişkeni, günlük ve izleme çıkışına ilişkin dizini belirtir.

#### MQ\_JAVA\_INSTALL\_PATH

Bu ortam deęişkeni, IBM MQ classes for Java kuruluş dizinlerindeki gibi, IBM MQ classes for Java ' in kurulu olduęu dizini belirtir.

#### MQ\_JAVA\_LIB\_PATH

Bu ortam deęişkeni, Her altyapıya ilişkin IBM MQ classes for Java kitaplıklarının yer alanında gösterildięi gibi, IBM MQ classes for Java kitaplıklarının saklandığı dizini belirtir. Some scripts supplied with IBM MQ classes for Java, such as IVTRun, use this environment variable.

 Windows' ta, kuruluş sırasında tüm ortam deęişkenleri otomatik olarak ayarlanır.

**UNIX** UNIX işletim sistemi üzerinde, ortam değişkenlerini ayarlamak için **setjmsenv** komut dosyasını (32 bit JVM kullanıyorsanız) ya da **setjmsenv64** (64 bit JVM kullanıyorsanız) kullanabilirsiniz.

**Linux** **UNIX** UNIX and Linux' ta bu komut dosyaları `MQ_INSTALLATION_PATH/java/bin` dizinindedir.

**IBM i** IBM üzerinde, `QIBM_MULTI_YIVY` ortam değişkeni Ydeğerine ayarlanmalıdır. Daha sonra birden çok iş parçacıklı uygulamaları, tek iş parçacıklı uygulamaları çalıştırdığınız şekilde çalıştırabilirsiniz. Ek bilgi için [IBM MQ olanağının Java ve JMS ile kurulması](#) başlıklı konuya bakın.

IBM MQ classes for Java , bir Java 7 Java Runtime Environment (JRE) gerektirir. IBM MQ ile kurulan uygun bir JRE ' nin yeriyile ilgili bilgi için bkz. “IBM MQ classes for Java için kuruluş dizinleri” sayfa 328.

#### IBM MQ classes for Java Kitaplıklar

IBM MQ classes for Java kitaplıklarının yeri altyapıya göre değişir. Bir uygulamayı başlattığınızda bu konumu belirtin.

Java Native Interface (JNI) kitaplıklarının yerini belirlemek için, aşağıdaki biçimi kullanarak **java** komutunu kullanarak uygulamanızı başlatın:

```
java -Djava.library.path= library_path application_name
```

Burada *kitaplık\_yolu* , JNI kitaplıklarını içeren IBM MQ classes for Java' ın yoludur. [Çizelge 52 sayfa 331](#) , her platform için IBM MQ classes for Java kitaplıklarının yerini gösterir. Bu çizelgede `MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Platform	IBM MQ classes for Java kitaplıklarını içeren dizin
<b>AIX</b> AIX	<code>MQ_INSTALLATION_PATH/java/lib</code> (32 bit kitaplık) <code>MQ_INSTALLATION_PATH/java/lib64</code> (64 bit kitaplık)
<b>Linux</b> Linux (x86 platformu)	<code>MQ_INSTALLATION_PATH/java/lib</code>
<b>Linux</b> Linux (POWER, x86-64 ve zSeries s390x altyapıları)	<code>MQ_INSTALLATION_PATH/java/lib</code> (32 bit kitaplık) <code>MQ_INSTALLATION_PATH/java/lib64</code> (64 bit kitaplık)
<b>Solaris</b> Solaris (x86-64 ve SPARC platformları)	<code>MQ_INSTALLATION_PATH/java/lib</code> (32 bit kitaplık) <code>MQ_INSTALLATION_PATH/java/lib64</code> (64 bit kitaplık)
<b>Windows</b> Windows	<code>MQ_INSTALLATION_PATH\Java\lib</code> (32 bit kitaplıklar) <code>MQ_INSTALLATION_PATH\Java\lib64</code> (64 bit kitaplıklar)
<b>z/OS</b> z/OS	<code>MQ_INSTALLATION_PATH/mqm/V8ROM0/java/lib</code> (32 bit ve 64 bit kitaplıklar)

#### Not:

- Solaris** **Linux** **AIX** AIX, Linux (Power platformu), ya da Solaris, 32 bit kitaplıklarını ya da 64 bit kitaplıklarını kullanır. Use the 64-bit libraries only if you are running your application in a 64-bit Java virtual machine (JVM) on a 64-bit platform. Tersisi durumda, 32 bit kitaplıkları kullanın.
- Windows** On Windows, you can use the PATH environment variable to specify the location of the IBM MQ classes for Java libraries instead of specifying their location on the **java** command.

3. **IBM i** To use IBM MQ classes for Java in bindings mode on IBM i, ensure that the library QMQMJAVA is in your library list.
4. **z/OS** z/OS' ta, 32 bit ya da 64 bit Java sanal makinesi (JVM) kullanabilirsiniz. Kullanılacak kitaplıkları belirtmenize gerek yoktur; IBM MQ classes for Java hangi JNI kitaplıklarının yükleneceğini kendisinin belirleyebileceğini belirtir.

### İlgili kavramlar

kullanma [IBM MQ classes for Java](#)

IBM MQ classes for Java kurulduktan sonra, kendi uygulamalarınızı çalıştırmak için kuruluşunuzu yapılandırabilirsiniz.

*IBM MQ classes for Java ile OSGi desteği*

OSGi, uygulamaların kod paketleri olarak konuşlandırılmasını destekleyen bir çerçeve sağlar. Üç OSGi kod paketi IBM MQ classes for Java' nin bir parçası olarak sağlanır.

OSGi, kod paketleri biçiminde gelen uygulamaların konuşlandırılmasını destekleyen genel amaçlı, güvenli ve yönetilen Java çerçevesini sağlar. OSGi uyumlu aygıtlar, paketleri yükleyebilir ve kurabilir ve artık gerekli olmadığında bunları kaldırabilir. Çerçeve, kod paketlerinin dinamik ve ölçeklenebilir bir şekilde kurulup 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' i kullanmalarına izin vermek için JAR dosyaları.

#### **com.ibm.mq.osgi.allclient\_version\_number.jar**

Bu JAR dosyası, uygulamaların hem IBM MQ classes for JMS hem de IBM MQ classes for Java' yi kullanmasını ve PCF iletilerini işleme kodunu da kapsamasını sağlar.

#### **com.ibm.mq.osgi.allclientprereqs\_version\_number.jar**

Bu JAR dosyası, com.ibm.mq.osgi.allclient\_version\_number.jar ile ilgili önkoşulları sağlar.

Burada *version\_number* , kurulu IBM MQ sürüm sayısıdır.

Bu paketler, IBM MQ kurulumunuzun `java/lib/OSGi` alt dizinine ya da Windows üzerindeki `java\lib\OSGi` klasörüne kurulur.

From IBM MQ 8.0, use the bundles `com.ibm.mq.osgi.allclient_8.0.0.0.jar`, and `com.ibm.mq.osgi.allclientprereqs_8.0.0.0.jar` for any new applications. Bu kod paketlerinin kullanılması, aynı OSGi çerçevesi içinde hem IBM MQ classes for JMS hem de IBM MQ classes for Java ' ı çalıştırmayacak şekilde kısıtlamayı kaldırır. Ancak diğer tüm kısıtlamalar yine de geçerli olur. IBM MQ 8.0 öncesi sürümler için, IBM MQ classes for JMS ya da IBM MQ classes for Java kullanılmasının kısıtlanması geçerlidir.

Diğer dokuz kod paketi de IBM MQ kurulumunuzun `java/lib/OSGi` alt dizinine ya da Windows üzerindeki `java\lib\OSGi` klasöründe kurulu olmalıdır. Bu paketler IBM MQ classes for JMS' in bir parçasıdır ve IBM MQ classes for Java kod paketi yüklü olan bir OSGi yürütme ortamına yüklenmemelidir. IBM MQ classes for Java OSGi paketi IBM MQ classes for JMS kod paketlerini de içeren bir OSGi yürütme ortamına yüklenirse, aşağıdaki örnekte gösterildiği gibi hatalar IBM MQ classes for Java kod paketi ya da IBM MQ classes for JMS kod paketlerini kullanan uygulamalar çalıştırıldığında ortaya çıkar:

```
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; bir OSGi Yayın Düzeyi 3 ortamında çalışmıyor.

OSGi yürütme ortamı gereken DLL kütüklerini ya da paylaşılan kitaplıkları bulabilmesi için, sistem yolunu ya da kitaplık yolunu doğru olarak ayarlamalısınız.

If you use the OSGi bundle for the IBM MQ classes for Java, channel exit classes written in Java are not supported because of an inherent problem in loading classes in a multiple class loader environment such as OSGi. Bir kullanıcı paketi IBM MQ classes for Java paketinin farkında olabilir, ancak IBM MQ

classes for Java kod paketi herhangi bir kullanıcı kod paketinin farkında değildir. Sonuç olarak, bir IBM MQ classes for Java kod paketinde kullanılan sınıf yükleyici, bir kullanıcı kod paketindeki bir kanal çıkış sınıfını yükleyemez.

OSGi hakkında daha fazla bilgi için [OSGi alliance web sitesine](#) bakın.

#### z/OS üzerinde IBM MQ classes for Java kuruluşu

z/OS üzerinde, yürütüm sırasında kullanılan STEPLIB, IBM MQ SCSQAUTH ve SCSQANLE kitaplıklarını içermelidir.

UNIX and Linux System Services olanağından, aşağıdaki örnekte gösterildiği gibi `.profile` 'ta bir çizgi kullanarak bu kitaplıkları ekleyebilirsiniz; IBM MQ'ı kurarken seçtiğiniz üst düzey veri kümesi niteleyicisiyle `th1qua1` yerine bir satır ekleyin:

```
export STEPLIB=th1qua1.SCSQAUTH:th1qua1.SCSQANLE:$STEPLIB
```

Diğer ortamlarda, genellikle başlatma JCL 'yi, STEPLIB bitişirmesine SCSQAUTH eklemek için düzenlemeniz gerekir:

```
STEPLIB DD DSN=th1qua1.SCSQAUTH,DISP=SHR  
        DD DSN=th1qua1.SCSQANLE,DISP=SHR
```

#### IBM MQ classes for Java yapılandırma dosyası

Bir IBM MQ classes for Java yapılandırma dosyası, IBM MQ classes for Java' u yapılandırmak için kullanılan özellikleri belirtir.

Bir IBM MQ classes for Java yapılandırma dosyasının biçimi, standart bir Java özellikler dosyasının biçimidir.

A sample configuration file, `mqjava.config`, is supplied in the `bin` subdirectory of the IBM MQ classes for Java installation directory. Bu dosya, desteklenen tüm özellikleri ve bunların varsayılan değerlerini belgeler.

**Not:** IBM MQ kuruluşu ilerideki bir Düzeltme Paketine yükseltildiğinde örnek yapılanış kütüğünün üzerine yazılır. Bu nedenle, örnek yapılanış kütüğünün bir kopyasını uygulamalarınızla birlikte kullanmak için yapmanız önerilir.

Bir IBM MQ classes for Java yapılandırma dosyasının adını ve konumunu seçebilirsiniz. Uygulamanıza başladığınızda, aşağıdaki biçimi kullanarak 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 yerini belirten bir URL 'dir. Şu tiplerin URL 'leri desteklenir: `http`, `file`, `ftp` ve `jar`.

Aşağıdaki örnek bir **java** komutunu göstermektedir:

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/mqjava.config MyAppClass
```

This command identifies the IBM MQ classes for Java configuration file as the file `D:\mydir\mqjava.config` on the local Windows system.

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 tanıtmıyorsa ya da yapılandırma dosyası bulunamazsa, IBM MQ classes for Java tüm özellikler için varsayılan değerleri kullanır. If required, you can override any property in the configuration file by specifying it as a system property on the **java** command.

Bir uygulama ile kuyruk yöneticisi ya da aracı arasındaki desteklenen aktarımlardan herhangi biriyle bir IBM MQ classes for Java yapılandırma dosyası kullanılabilir.

## Bir IBM MQ MQI client yapılandırma dosyasında belirtilen özelliklerin geçersiz kılınması

Bir IBM MQ MQI client yapılandırma dosyası, IBM MQ classes for Java' u yapılandırmak için kullanılan özellikleri de belirtebilir. Ancak, bir IBM MQ MQI client yapılandırma dosyasında belirtilen özellikler yalnızca, bir uygulama istemci kipinde bir kuyruk yöneticisine bağlandığında geçerlidir.

If required, you can override any attribute in an IBM MQ MQI client configuration file by specifying it as a property in an IBM MQ classes for Java configuration file. To override an attribute in an IBM MQ MQI client configuration file, use an entry with the following format in the IBM MQ classes for Java configuration file:

```
com.ibm.mq.cfg.stanza.propName=propValue
```

Girdideki değişkenler aşağıdaki anlamlara sahiptir:

### **stanza**

Özniteliği içeren IBM MQ MQI client yapılandırma dosyasındaki stanza adının adı.

### **propName**

IBM MQ MQI client yapılandırma dosyasında belirtildiği gibi özniteliğin adı.

### **propValue**

IBM MQ MQI client yapılandırma dosyasında belirtilen özniteliğin değerini geçersiz kılan özelliğin değeri.

Alternatively, you can override an attribute in an IBM MQ MQI client configuration file by specifying the property as a system property on the **java** command. Özelliği bir sistem özelliği olarak belirtmek için önceki biçimi kullanın.

Bir IBM MQ MQI client yapılandırma dosyasında yalnızca aşağıdaki öznitelikler IBM MQ classes for Java ile ilgilidir. Diğer öznitelikleri belirtmiş ya da geçersiz kıyorsanız, bu herhangi bir etkisi olmaz. Specifically, note that the ChannelDefinitionFile and ChannelDefinitionDirectory in the İstemci yapılandırma dosyasının STANA kısmı are not used. CCDT ' yi IBM MQ classes for Java ile nasıl kullanabilmeye ilişkin ayrıntılar için [“Using a client channel definition table with IBM MQ classes for Java”](#) sayfa 349 konusuna bakın.

Çizelge 53. İstemci yapılandırma dosyasında hangi özniteliğin bulunduğunu içeren	
Stanza	Öznitelik
<a href="#">İstemci yapılandırma dosyasının STANA kısmı</a>	Put1DefaultAlwaysSync
<a href="#">İstemci yapılandırma dosyasının STANA kısmı</a>	PasswordProtection
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	ExitsDefaultYolu
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	ExitsDefaultPath64
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	JavaExitsClasspath
<a href="#">İstemci yapılandırma dosyasının JMQI kısmı</a>	useMQCSPauthentication
<a href="#">İstemci yapılandırma dosyasınınMessageBuffer kısmı</a>	MaximumSize
<a href="#">İstemci yapılandırma dosyasınınMessageBuffer kısmı</a>	PurgeTime
<a href="#">İstemci yapılandırma dosyasınınMessageBuffer kısmı</a>	UpdatePercentage
<a href="#">İstemci yapılandırma dosyasının TCP stanzası</a>	ClntRcvBuffSize
<a href="#">İstemci yapılandırma dosyasının TCP stanzası</a>	ClntSndBuffSize

Çizelge 53. İstemci yapılandırma dosyasında hangi özneliğin bulunduğunu içeren (devamı var)

Stanza	Öznelik
İstemci yapılandırma dosyasının TCP stanzası	Bağlantı_Zamanaşımı
İstemci yapılandırma dosyasının TCP stanzası	KeepAlive

IBM MQ MQI client yapılandırmasıyla ilgili daha fazla bilgi için bkz. [Yapılandırma dosyası kullanarak istemci yapılandırılması.](#)

### İlgili görevler

[Tracing IBM MQ classes for Java applications](#)



*Java Standard Environment Trace stanza*

IBM MQ classes for Java izleme olanağını yapılandırmak için Java Standard Environment Trace Settings stanza olanağını kullanın.

#### **com.ibm.msg.client.commonservices.trace.outputName = traceOutputAd**

*traceOutputName* , izleme çıkışının gönderileceği dizin ve dosya adıdır.

Varsayılan olarak izleme bilgileri, uygulamanın yürürlükteki çalışma dizinindeki bir izleme dosyasına yazılır. İzleme dosyasının adı, uygulamanın çalışmakta olduğu ortama bağlıdır:

- 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`.
- From IBM MQ 9.0.0 Fix Pack 2, if the application has loaded the IBM MQ classes for Java from the JAR file `com.ibm.mq.jar`, trace is written to a file called `mqjava_%PID%.trc`.
- From IBM MQ 9.0.0 Fix Pack 2, if the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.trc`.
-  From IBM MQ 9.1.5 ve IBM MQ 9.1.0 Fix Pack 5, if the application has loaded the IBM MQ classes for Java from the JAR file `com.ibm.mq.jar`, trace is written to a file called `mqjava_%PID%.c1%u.trc`.
-  From IBM MQ 9.1.5 ve IBM MQ 9.1.0 Fix Pack 5, if the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.c1%u.trc`.

Burada `%PID%` , izlenmekte olan uygulamanın işlem tanıtıcısıdır ve `%u` , farklı Java sınıf yükleyicileri altında izleme çalıştıran iş parçacıkları arasında dosyaları ayırt etmek için kullanılan benzersiz bir sayıdır.

If a process ID is unavailable, a random number is generated and prefixed with the letter `f`. Süreç tanıtıcısını belirttiğiniz bir dosya adına eklemek için `%PID%dizgisini` kullanın.

Başka bir izin belirtirseniz, bu izin var olmalıdır ve bu izin için yazma izniniz olmalıdır. Yazma izniniz yoksa, izleme çıkışı `System.err` olarak yazılır.

#### **com.ibm.msg.client.commonservices.trace.include = includeList**

*includeList* , izlenecek paketlerin ve sınıfların bir listesidir ya da özel değerler ALL ya da NONE.

Paketi ya da sınıf adlarını noktalı virgül ( ; ) ile ayırın. *includeList* varsayılan olarak ALL değerine ayarlanır ve IBM MQ classes for Java içindeki tüm paketleri ve sınıfları izler.

**Not:** Bir paket dahil edebilir, ancak o paketin alt paketlerini dışlayabilirsiniz. For example, if you include package `a.b` and exclude package `a.b.x`, the trace includes everything in `a.b.y` and `a.b.z`, but not `a.b.x` or `a.b.x.1`.

#### **com.ibm.msg.client.commonservices.trace.exclude = excludeList**

*excludeList* , izlenmeyen paketler ve sınıfların bir listesidir ya da özel değerler ALL ya da NONE.

Paketi ya da sınıf adlarını noktalı virgöl ( ; ) ile ayırın. *excludeList* varsayılan olarak NONEdeğerine ayarlanır ve bu nedenle, IBM MQ classes for JMS içindeki hiçbir paketi ve sınıfı izlemekten dışlanmaz.

**Not:** Bir paketi dışlayabilir, ancak o paketin alt paketlerini de içerebilirsiniz. For example, if you exclude package a . b and include package a . b . x, the trace includes everything in a . b . x and a . b . x . 1, but not a . b . y or a . b . z.

Hem içerilen hem de dışlanan paket ya da sınıf, aynı düzeyde belirtilmiş ve dışlanmış olarak dahil edilir.

**com.ibm.msg.client.commonservices.trace.maxBytes = *maxArrayByte***

*maxArrayBytes* , herhangi bir bayt dizisinden izlenen bayt sayısı üst sınışıdır.

*maxArrayBytes* pozitif bir tamsayıya ayarlandıysa, izleme dosyasına yazılacak bayt dizideki bayt sayısını sınırlar. *maxArrayBytes* yazıldıktan sonra bayt dizisini keser. *maxArrayBytes* ayarı sonucunda elde edilen izleme dosyası büyüklüğü azaltılıyor ve izleme işlemi, uygulamanın performansında etkisini azaltır.

Bu özellik için 0 değeri, izleme dosyasına herhangi bir byte dizisinin içeriğinin hiçbirinin gönderilmemesinin anlamına gelir.

Varsayılan değer -1değeridir. Bu değer, izleme dosyasına gönderilen bayt dizisindeki bayt sayısındaki herhangi bir sınırı kaldırır.

**com.ibm.msg.client.commonservices.trace.limit = *maxTraceByte***

*maxTraceBytes* , bir izleme çıkış dosyasına yazılan bayt sayısı üst sınışıdır.

*maxTraceBytes* , *traceCycles* ile çalışır. Yazılan izleme baytlarının sayısı sınıra yakınsa, dosya kapatılır ve yeni bir izleme çıkış dosyası başlatılır.

0 değeri, bir izleme çıkış dosyasının sıfır uzunluğuna sahip olduğu anlamına gelir. Varsayılan değer -1değeridir. Bu, bir izleme çıkış dosyasına yazılacak veri miktarının sınırsız olduğu anlamına gelir.

**com.ibm.msg.client.commonservices.trace.count = *traceCycles***

*traceCycles* , çevrim yoluyla çevrilecek izleme çıkış dosyalarının sayısıdır.

Yürürlükteki izleme çıkış dosyası *maxTraceBytes* ile belirtilen sınıra ulaşırsa, dosya kapatılır. Ek izleme çıkışı, sonraki izleme çıkış dosyasına sırayla yazılır. Her izleme çıkış dosyası, dosya adının sonuna eklenen sayısal bir sonek tarafından ayırt edilir. Yürürlükteki ya da en son izleme çıkış dosyası mqjms . trc . 0, sonraki en son izleme çıkış dosyası mqjms . trc . 1. Daha eski izleme dosyaları, sınırlamaya kadar aynı numaralandırma örüntülerini izler.

*traceCycles* varsayılan değeri 1 'dir. *traceCycles* 1 ise, yürürlükteki izleme çıkış dosyası büyüklük üst sınırına ulaştığında, dosya kapatılır ve silinir. Aynı adı taşıyan yeni bir izleme çıkış dosyası başlatılmış. Bu nedenle, bir kerede tek bir izleme çıkış dosyası vardır.

**com.ibm.msg.client.commonservices.trace.parameter = *traceParameters***

*traceParameters* yöntem değıştirgelerinin ve dönüş değerlerinin izlemenin içerilip içerilmeyeceğini denetler.

*traceParameters* varsayılan olarak TRUEdeğerine ayarlanır. *traceParameters* FALSEolarak ayarlandıysa, yalnızca yöntem imzaları izlenir.

**com.ibm.msg.client.commonservices.trace.startup = *başlangıç***

Kaynakların ayrıldığı, IBM MQ classes for Java ' un kullanıma hazırlama aşaması vardır. Ana izleme olanağı, kaynak ayırma aşaması sırasında kullanıma hazırlanır.

*startup* TRUEolarak ayarlandıysa, başlatma izlemesi kullanılır. İzleme bilgileri hemen üretilir ve izleme olanağı da dahil olmak üzere tüm bileşenlerin kuruluşunu içerir. Başlatma izleme bilgileri, yapılandırma sorunlarını tanılamak için kullanılabilir. Başlatma izleme bilgileri her zaman System . err' a yazılır.

*startup* varsayılan olarak FALSEdeğerine ayarlanır.

*startup* , kullanıma hazırlama işlemi tamamlanmadan önce denetlenir. Bu nedenle, yalnızca komut satırındaki özelliği bir Java sistem özelliği olarak belirtin. Bunu IBM MQ classes for Java yapılandırma dosyasında belirtmeyin.



**com.ibm.msg.client.commonservices.trace.compress = compressedTrace**

İzleme çıkışı sıkıştırmak için *compressedTrace* , TRUE olarak ayarlayın.

*compressedTrace* varsayılan değeri FALSEdeğeridir.

*compressedTrace* TRUEolarak ayarlandıysa, izleme çıkışı sıkıştırılır. Varsayılan izleme çıkışı dosyası adı, *.truzantısına* sahiptir. If compression is set to FALSE, the default value, the file has the extension *.trc* to indicate it is uncompressed. Ancak, izleme çıkışının dosya adı *traceOutputName* içinde belirtilmiş olsa da, bu ad kullanılırsa, dosyaya sonek uygulanmaz.

Sıkıştırılmış izleme çıkışı, sıkıştırılmamış boyuttan küçük. Daha az G/Ç olduğundan, sıkıştırılmamış izlemenin daha hızlı bir şekilde yazılabiliyor. Sıkıştırılmış izleme, IBM MQ classes for Java ' un performansı, sıkıştırılmamış izlemenin dışında daha az etkiye sahiptir.

*maxTraceBytes* ve *traceCycles* ayarlandıysa, birden çok düz dosya yerine birden çok sıkıştırılmış izleme dosyası oluşturulur.

IBM MQ classes for Java denetimli olarak sona erdirilirse, sıkıştırılmış izleme dosyası geçerli olmayabilir. Bu nedenle, izleme sıkıştırması yalnızca IBM MQ classes for Java denetimli bir şekilde kapatıldığında kullanılmalıdır. Yalnızca araştırılmakta olan sorunlar JVM ' nin beklenmedik bir şekilde durmasına neden değilse, izleme sıkıştırmayı kullanın. *System.Halt()* kapatma ya da olağandışı olmayan, denetimsiz JVM sonlandırıcılarıyla sonuçlanabilen 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 alt düzeylerin tümünü içerir. Örneğin, izleme düzeyi TRACE\_INFOolarak ayarlandıysa, izleme düzeyi TRACE\_EXCEPTION, TRACE\_WARNINGya da TRACE\_INFO tanımlı düzeyli herhangi bir izleme noktası yazılır. Diğer tüm izleme noktaları dışlanır.

**com.ibm.msg.client.commonservices.trace.standalone = standaloneTrace**

*standaloneTrace* controls whether the IBM MQ classes for Java client tracing service is used in a WebSphere Application Server environment.

*standaloneTrace* TRUEolarak ayarlandıysa, izleme yapılandırmasını saptamak için IBM MQ classes for Java istemci izleme özellikleri kullanılır.

*standaloneTrace* FALSEolarak ayarlandıysa 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 bağlı olarak değişir.

*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 dağıtımına 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 8.0'içinden, `com.ibm.mq.allclient.jar` JAR dosyasının havuza kurulması gerekir.

IBM MQ 9.0' dan Bouncy Castle güvenlik sağlayıcısı ve CMS destek JAR dosyaları gereklidir. Daha fazla bilgi için bkz. [“IBM MQ classes for Java yeniden yerleşme JAR dosyaları” sayfa 327](#) ve [IBM dışı JRE ' ler için destek](#).

### **IBM MQ classes for Java uygulamaları için kuruluş sonrası ayarları**

IBM MQ classes for Java kurulduktan sonra, kendi uygulamalarınızı çalıştırmak için kuruluşunuzu yapılandırabilirsiniz.

Remember to check the IBM MQ product readme file for the latest information, or for more specific information about your environment. Ürün Beni Oku dosyasının en son sürümü [IBM MQ, WebSphere MQ, ve MQSeries ürün readmes web sayfasında](#) bulunur.

Before attempting to run an IBM MQ classes for Java application in bindings mode, make sure that you have configured IBM MQ as described in [Yapılandırılıyor](#).

*IBM MQ classes for Java' den istemci bağlantılarını kabul etmek için kuyruk yöneticinizin yapılandırılması*  
Kuyruk yöneticinizi istemcilerden gelen bağlantı isteklerini kabul edecek şekilde yapılandırmak için, bir sunucu bağlantı kanalının kullanımını tanımlayın ve bu kanaldan izin verin ve bir dinleyici programı başlatın.

Ayrıntılar için bkz. [“Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102](#).

#### *Running IBM MQ classes for Java applications under the Java security manager*

IBM MQ classes for Java , Java security manager etkin ile çalışabilir. Uygulamaları Java security manager etkinleştirilmiş olarak başarıyla çalıştırmak için, Java virtual machine (JVM) olanağını uygun bir ilke tanımlama dosyası ile yapılandırmalısınız.

Uygun bir ilke tanımlama dosyası oluşturmanın en kolay yolu, Java runtime environment (JRE) ile sağlanan ilke dosyasını değiştirmemektir. Çoğu sistemde bu dosya, JRE dizininize göre `path lib/security/java.policy` dizininde saklanır. İlke dosyalarını, tercih ettiğiniz düzenleyiciyi kullanarak ya da JRE ' nize birlikte verilen ilke aracı programını kullanarak düzenleyebilirsiniz.

You must give authority to the `com.ibm.mq.jmqi.jar` file so that it can:

- Yuva yarat (istemci kipinde)
- Yerli kitaplığı yükle (bağ tanımları kipinde)
- Ortamdaki çeşitli özellikleri okuyun

The system property **os.name** must be available to the IBM MQ classes for Java when running under the Java security manager.

Java uygulamanız Java security manager kullanıyorsa, uygulama tarafından kullanılan `java.security.policy` dosyasına aşağıdaki izni eklemeniz gerekir; tersi durumda, uygulamaya kural dışı durumlar yayınlanacaktır:

```
permission java.lang.RuntimePermission "modifyThread";
```

Bu RuntimePermission (RuntimePermission), istemci tarafından, kuyruk yöneticilerine TCP/IP bağlantıları üzerinden çoklayıcı etkileşimlerin atanmasını ve kapatılıp kapatılabilmemesinin bir parçası olarak gereklidir.

## Örnek ilke dosyası girişi

Aşağıda, IBM MQ classes for Java ' un varsayılan güvenlik yöneticisi altında başarıyla çalıştırılmasını sağlayan bir ilke dosyası girişi örneği yer alıyor. Replace the string `MQ_INSTALLATION_PATH` in this example with the location where IBM MQ classes for Java are installed on your system.

```
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";
};
```


This example of a policy file enables the IBM MQ classes for Java to work correctly under the security manager, but you might still need to enable your own code to run correctly before your applications work.

IBM MQ classes for Java ile birlikte verilen örnek kod, güvenlik yöneticisiyle birlikte kullanılmak üzere özel olarak etkinleştirilmedi; ancak IVT sınamaları bu ilke dosyasıyla ve varsayılan güvenlik yöneticisi ile çalıştırılıyor.

### *IBM MQ classes for Java uygulamaları CICS Transaction Server altında çalıştırılıyor*

Bir IBM MQ classes for Java uygulaması, CICS Transaction Server altında bir hareket olarak çalıştırılabilir.

Bir IBM MQ classes for Java uygulamasını z/OS için CICS Transaction Server altında bir hareket olarak çalıştırmak için aşağıdaki adımları gerçekleştirin:

1. Sağlanan CEDA işlemini kullanarak uygulamayı ve işlemi CICS ' e tanımlayın.
2. Ensure that the IBM MQ CICS adapter is installed in your CICS system.  (Ayrıntılar için [IBM MQ ile CICS komutunu kullanma konusuna bakın.](#))
3. CICS içinde belirtilen JVM ortamının uygun CLASSPATH ve LIBPATH girişlerini içermesine dikkat edin.
4. Normal süreçlerinizin herhangi birini kullanarak işlemi başlatın.

For more information on running CICS Java transactions, refer to your CICS system documentation.

### **IBM MQ classes for Java kuruluşunun doğrulanması**

Bir kuruluş doğrulama programı (MQIVP), IBM MQ classes for Java ile birlikte sağlanır. Bu programı, IBM MQ classes for Java ile ilgili tüm bağlantı kiplerini sınamak için kullanabilirsiniz.

Program, doğrulamak istediğiniz bağlantı kipini belirlemek için bir dizi seçenek ve diğer veri bilgi istemlerini içerir. Kuruluşunuzu doğrulamak için aşağıdaki yordamı kullanın:

1. Programı istemci kipinde çalıştırabiliyorsanız, kuyruk yöneticinizi “Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102 içinde açıklandığı gibi yapılandırın. Kullanılacak kuyruk SYSTEM.DEFAULT.LOCAL.QUEUE.
2. Programı istemci kipinde çalıştırabiliyorsanız, bkz. “kullanma IBM MQ classes for Java” sayfa 321.  
Bu yordamın geri kalan adımlarını, programı çalıştırabilmekte olduğunuz sistemde gerçekleştirin.
3. CLASSPATH ortam değişkeninizi, “IBM MQ classes for Java ile ilgili ortam değişkenleri” sayfa 330 içindeki yönergelerle 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 istemine şunu girin:

```
java -Djava.library.path= library_path MQIVP
```

Burada `library_path`, IBM MQ classes for Java kitaplıklarının yoludur (bkz. “IBM MQ classes for Java Kitaplıklar” sayfa 331).

(1) imlenmiş bilgi isteminde aşağıdaki bilgiler yer aldı:

- TCP/IP bağlantısı kullanmak için bir IBM MQ sunucusu anasistem adı girin.
- Yerel bağlantıyı kullanmak için (bağ tanımları kipi) alanı boş bırakın (ad girmeyin).

Program aşağıdakileri yapmaya çalışır:

- 1. Kuyruk yöneticisine bağlan
- 2. SYSTEM.DEFAULT.LOCAL.QUEUE kuyruğunu açın, kuyruğa bir ileti yazın, kuyruktan bir ileti alın ve kuyruğu kapatın.



- 3. Kuyruk yöneticisinden bağlantı kesme
- 4. İşlemler başarılı olursa bir ileti döndür

Burada, görebileceğiniz bilgi istemlerinin ve yanıtların bir örneği yer alır. 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' ta bilgi isteminde <sup>(1)</sup>imi işaretlenerek alanı boş bırakın.
2. Sunucu bağlantısını seçerseniz, <sup>(2)</sup>ile işaretlenen bilgi istemlerini görmeyin.
3.  On IBM i, you can only issue the `java MQIVP` command from QShell. Diğer bir seçenek olarak, uygulamayı `RUNJAVA 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, kuruluşunuzu ve ileti sistemi sunucunuzu ayarlamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

### **Bu görev hakkında**

Kendi uygulamalarınızı yaratmak için yardıma gerek 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 nesneyi (MQQueueManager, MQConstants, MQMessage, MQPutMessageSeçenekleri ve MQDestination) yaratmak ve uygulamanızın nasıl çalışmasını istediğinizi belirlemek için gereken belirli özellikleri tanımlamak için. Daha fazla bilgi için bkz “IBM MQ classes for Java uygulamaları yazılıyor” sayfa 344. The samples might be subject to change in future releases of IBM MQ Java.

Çizelge 54 sayfa 341 , her altyapıda IBM MQ classes for Java örnek uygulamalarının kurulu olduğu yeri gösterir:






Platform		Dizin
 UNIX	UNIX	MQ_INSTALLATION_PATH/samp/wmqjava/samples
 Linux	Linux	
 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 55 sayfa 342 , IBM MQ classes for Java ile birlikte verilen örnek uygulamaların kümelerini gösterir.

Çizelge 55. IBM MQ classes for Java örnek uygulamalar	
Örneklerin adı	Tanım
IMSBridgeSample.java	IMS köprüsünü IBM MQ classes for Java ile birlikte kullanarak göstermek için basit bir program.
MQIVP.java	IBM MQ Java kuruluş doğrulama programı.
MQMessagePropertiesSample.java	Demonstrates the use of the Message Properties API introduced into IBM WebSphere MQ 7.0.
MQPubSubApiSample.java	IBM WebSphere MQ 7.0'ta tanımlanan yayınlama/abone olma API' yi gösterir.
MQSample.java	Bir kuyruktan ileti koymayı ve almayı göstermek için basit bir program.
MQSampleMessageManager.java	IBM MQ temel Java örneklerinde ileti işleme yardımcı programı sınıfı.
mqjcvp.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 kullanılacak runjms adlı bir komut kütüğü sağlar. This script sets up the IBM MQ environment to allow you to run the IBM MQ classes for Java sample applications.

Çizelge 56 sayfa 342 , her altyapıda komut dosyasının yerini gösterir:

Çizelge 56. runjms komut dosyasının konumu	
Platform	Dizin
 UNIX  Linux	MQ_INSTALLATION_PATH/java/bin/runjms
 Windows	MQ_INSTALLATION_PATH\java\bin\runjms.bat
 IBM i	/qibm/proddata/mqm/java/bin/runjms ya da /qibm/proddata/mqm/java/bin/runjms64
 z/OS	MQ_INSTALLATION_PATHjava/bin/runjms

Örnek bir uygulamayı çağırmak için runjms komut dosyasını kullanmak için aşağıdaki adımları tamamlayın:

## 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 listesini görüntüler.

3. Örneği bu değiştirgelerle çalıştırmak için aşağıdaki komutu girin:

Path to the runjms script/runjms sample\_application\_name parameters

## Örnek

Linux

For example, to run the MQIVP sample on Linux, enter the following commands:

```
cd /opt/mqm/samp/wmqjava/samples
/opt/mqm/java/bin/runjms MQIVP
```

## İlgili kavramlar

[“IBM MQ classes for JMS için kurulu olan” sayfa 81](#)

IBM MQ classes for JMS' u kurduğunuzda, bir dizi dosya ve dizin oluşturulur. Windows' ta, bazı yapılandırma otomatik olarak ortam değişkenleri ayarlanarak kuruluş sırasında gerçekleştirilir. On other platforms, and in certain Windows environments, you must set environment variables before you can run IBM MQ classes for JMS applications.

## IBM MQ classes for Java sorunlarının çözülmesi

Başlangıçta kuruluş doğrulama programını çalıştırın. Ayrıca izleme tesisini de kullanmak zorunda kalabilirsiniz.

Bir uygulama başarıyla tamamlanmazsa, kuruluş doğrulama programını çalıştırın ve tanılama iletilerinde verilen öneriyi izleyin. Kuruluş doğrulama programı [“IBM MQ classes for Java kuruluşunun doğrulanması” sayfa 340](#) içinde açıklanmıştır.

Sorunlar devam ederse ve IBM hizmet ekibiyle bağlantı kurmanız gerekiyorsa, izleme olanağını açmanız istenebilir. Bu işlemi aşağıdaki örnekte gösterildiği gibi yapın.

MQIVP programını izlemek için:

- Bir `com.ibm.mq.commonservices` özellikler dosyası oluşturun (bkz. [com.ibm.mq.commonservices](#) komutunu kullanma).
- 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 (properties) dosyasına giden yoldur (dosya adı da içinde olmak üzere).
- `library_path` , IBM MQ classes for Java kitaplıklarının yoludur (bkz. [“IBM MQ classes for Java Kitaplıklar” sayfa 331](#)).

İzlemenin nasıl kullanılacağı hakkında daha fazla bilgi için [IBM MQ classes for Java uygulamalarını izleme](#) başlıklı konuya bakın.

## z/OS V 9.1.0 MQ Adv. VUE Java üzerinde çalışan toplu iş uygulamalarına istemci bağlantılığı z/OS

Bir istemci bağlantısı kullanarak, z/OS üzerindeki IBM MQ classes for Java uygulaması, **ADVCAP** (ENABLED) özneliği olan z/OS üzerindeki bir kuyruk yöneticisine bağlanabilir. İstemci bağlantısının kullanılması IBM MQ topolojilerini basitleştirebilir.

**ADVCAP** (ENABLED) değeri yalnızca, IBM MQ Advanced for z/OS Value Unit Edition (bkz. [IBM MQ ürün tanıtıcıları ve dışa aktarma bilgileri](#)) olarak lisanslanan ve **QMGRPROD** ayarı ZAVANTAJLI olan z/OS kuyruk yöneticisi için geçerlidir.

**QMGRPROD** ile ilgili daha fazla bilgi için bkz. [DISPLAY QMGR ADVCAP](#) ve [START QMGR](#) .

z/OS üzerindeki bir IBM MQ classes for Java uygulaması, z/OS üzerinde çalışmayan bir kuyruk yöneticisine ya da **ADVCAP** (ENABLED) seçeneği ayarlanmamış bir kuyruk yöneticisine bağlanmak için istemci kipi bağlantısını kullanamaz.

z/OS üzerinde bir IBM MQ classes for Java uygulaması istemci kipini kullanarak bağlanmaya çalışırsa ve buna izin verilmezse, MQRC\_ENVIRONMENT\_ERROR döndürülür.

## Advanced Message Security (AMS) desteği

V 9.1.0

IBM MQ 9.1' den IBM MQ classes for Java istemci uygulamaları, uzak z/OS sistemlerinde IBM MQ Advanced for z/OS Value Unit Edition kuyruk yöneticilerine bağlanırken AMS komutunu kullanabilir.

`jceracfs` anahtar deposu tipi yalnızca z/OS üzerinde `keystore.conf` içinde desteklenir; burada:

- Özellik adı önceki `jceracfs` ve bu ad önceki büyük ve küçük harfe duyarlı değildir.
- Anahtar deposu bir RACF anahtarlığı.
- Parolalar gerekli değildir ve yoksayıdır. Bunun nedeni, RACF anahtarlarının parola kullanmaması olabilir.
- Sağlayıcıyı belirtirseniz, sağlayıcı IBMJCE olmalıdır.

`jceracfs` komutunu AMS ile 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:

```
jceracfs.keystore=safkeyring://JOHNDOE/drq.ams.keyring
```

## IBM MQ classes for Java uygulamaları yazılıyor

This collection of topics provides information to assist with writing Java applications to interact with IBM MQ systems.

To use IBM MQ classes for Java to access IBM MQ queues, you write Java applications that contain calls that put messages onto, and get messages from, IBM MQ queues. Tek tek sınıflara ilişkin ayrıntılar için bkz. [IBM MQ classes for Java](#).

**Not:** Otomatik istemci yeniden bağlantısı IBM MQ classes for Java tarafından desteklenmiyor.

## IBM MQ classes for Java arabirimi

Yordamsal IBM MQ uygulama programlama arabirimi, nesnelere üzerinde işlem yapan filleri kullanır. Java programlama arabirimi, yöntemleri çağırarak üzerinde işlem yapmak için nesnelere kullanır.

Yordamsal IBM MQ uygulama programlama arabirimi, aşağıdaki gibi fillere çevresinde oluşturulur:

```
MQBACK, MQBEGIN, MQCLOSE, MQCONN, MQDISC,  
MQGET, MQINQ, MQOPEN, MQPUT, MQSET, MQSUB
```

Bu fillerin tümü, bir parametre olarak, üzerinde çalışacakları IBM MQ nesnesinin bir tanıtıcısı olarak alır. Programınız, bu nesnelere ilgili yöntemler çağırarak işlem yapmak istediğiniz IBM MQ nesnelere oluşur.

Yordamsal arabirimi kullanırken, `MQDISC` çağrısını (`Hconn`, `CompCode`, `Reason`) kullanarak bir kuyruk yöneticisinden bağlantınızı kesilir; burada `Hconn`, kuyruk yöneticisinin tanıtıcısıdır.



Java arabiriminde, kuyruk yöneticisi, MQQueueManagersınıfı bir nesle temsil edilir. Bu sınıftaki bağlantı kesme () yöntemini çağırarak, kuyruk yöneticisinden bağlantıyı kesmenizi sağlar.

```
// 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 işletim yönteminde, kullanmak istediğiniz bağlantı kiplerine bazı bağımlılıklar var.

İstemci bağlantıları kullanıyorsanız, IBM MQ MQI client ' den bir dizi fark vardır; ancak kavramsal olarak benzerdir. Bağ tanımları kipini kullanıyorsanız, fastpath bağ tanımlarını kullanabilir ve MQBEGIN komutunu verebilirsiniz. MQEnvironment sınıfındaki değişkenleri ayarlayarak hangi kipin kullanılacağını belirtmenizi sağlar.

#### *IBM MQ classes for Java istemci bağlantıları*

When IBM MQ classes for Java is used as a client, it is like the IBM MQ MQI client, but has a number of differences.

IBM MQ classes for Java için istemci olarak kullanılmak üzere programlıyorsanız, aşağıdaki farklardan haberdar olun:

- Yalnızca TCP/IP ' yi destekler.
- Başlatma sırasında hiçbir IBM MQ ortam değişkenini okumaz.
- Bir kanal tanımlamasında ve ortam değişkenlerinde saklanacak bilgiler, Ortam adlı bir sınıfa saklanabilir. Diğer bir seçenek olarak, bu bilgiler, bağlantı yapıldığında parametre olarak geçirilebilir.
- Hata ve kural dışı durum koşulları, MQException sınıfında belirtilen günlüğe yazılır. Varsayılan hata hedefi, Java konsolidir.
- Bir IBM MQ istemcisi yapılandırma dosyasında yalnızca aşağıdaki öznitelikler IBM MQ classes for Java ile ilgilidir. Diğer öznitelikleri belirtirseniz, bunlar geçersiz olur.

<b>Stanza</b>	<b>Öznitelik</b>
<u>ClientExit</u> İstemci yapılandırma dosyasının yol gösterişi	ExitsDefaultYolu
<u>ClientExit</u> İstemci yapılandırma dosyasının yol gösterişi	ExitsDefaultPath64
<u>ClientExit</u> İstemci yapılandırma dosyasının yol gösterişi	JavaExitsClasspath
İstemci yapılandırma dosyasınınMessageBuffer kısmı	MaximumSize
İstemci yapılandırma dosyasınınMessageBuffer kısmı	PurgeTime
İstemci yapılandırma dosyasınınMessageBuffer kısmı	UpdatePercentage
İstemci yapılandırma dosyasının TCP stanzası	ClntRcvBuffSize
İstemci yapılandırma dosyasının TCP stanzası	ClntSndBuffSize
İstemci yapılandırma dosyasının TCP stanzası	Bağlantı_Zamanaşımı
İstemci yapılandırma dosyasının TCP stanzası	KeepAlive

- If connecting to a queue manager that requires character data to be converted, then the V7 Java client is now capable of doing the conversion if queue manager is unable to do so. İstemci JVM 'nin, istemcinin CCSID 'si ile kuyruk yöneticisinin arasındaki dönüşümü desteklemesi gerekir.
- Otomatik istemci yeniden bağlanması IBM MQ classes for Javatarafından desteklenmez.

İstemci kipinde kullanıldığında, *IBM MQ classes for Java* MQBEGIN çağrısını desteklemez.

#### *IBM MQ classes for Java bağ tanımları kipi*

IBM MQ classes for Java bağ tanımları kipi, istemci kipinden üç ana yol kadar farklılık gösterir.

When used in bindings mode, IBM MQ classes for Java uses the Java Native Interface (JNI) to call directly into the existing queue manager API, rather than communicating through a network.

Varsayılan olarak, bağ tanımları kipinde IBM MQ classes for Java ögesini kullanan uygulamalar, *ConnectOption*, MQCNO\_STANDARD\_BAĞLAMALARI kullanılarak bir kuyruk yöneticisine bağlanır.

IBM MQ classes for Java , aşağıdaki *ConnectOptions*' i destekler:

- MQCNO\_FASTPATH\_BINDING
- MQCNO\_STANDARD\_BINDING
- MQCNO\_SHARED\_BINDING
- MQCNO\_ISOLATED\_BINDING

*ConnectOptions* ile ilgili daha fazla bilgi için bkz. [“MQCONNX çağrısını kullanarak kuyruk yöneticisine bağlanma” sayfa 775.](#)

Bindings mode supports the MQBEGIN call to initiate global units of work that are coordinated by the queue manager, on all platforms apart from IBM MQ for IBM i and IBM MQ for z/OS.

MQEnvironment sınıfı tarafından sağlanan parametrelerin çoğu bağ tanımları kipiyle ilgili değildir ve yoksayıdır.

#### *Kullanılacak IBM MQ classes for Java bağlantısının tanımlanması*

Kullanılacak bağlantı tipi, MQEnvironment sınıfındaki değişkenlerin ayarlarıyla belirlenir.

İki değişken kullanılır:

#### **MQEnvironment.properties**

Bağlantı tipi, CMQC . TRANSPORT\_PROPERTY anahtar adıyla ilişkili değer tarafından belirlenir. Olası değerler aşağıdaki gibidir:

#### **CMQC.TRANSPORT\_MQSERIES\_BINDINGS**

Bağ tanımları moduna 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 ana makine adına ayarlayın.
- Bağ tanımları kipi için bu değişkeni ayarlamayın ya da boş değere ayarlayın

### ***Kuyruk yöneticilerine ilişkin işlemler***

This collection of topics describes how to connect to, and disconnect from, a queue manager using IBM MQ classes for Java.

#### *IBM MQ classes for Java için IBM MQ ortamını ayarlama*

Bir uygulamanın istemci kipinde bir kuyruk yöneticisine bağlanmasını sağlamak için, uygulamanın kanal adını, anasistem adını ve kapı numarasını belirtmesi gerekir.

**Not:** Bu konudaki bilgiler, uygulamanızın istemci kipindeki bir kuyruk yöneticisine bağlanması durumunda geçerlidir. Bağ tanımları moduna bağlanıyorsa, bu durum uygun değildir. Bkz. [“IBM MQ classes for JMS için bağlantı kipleri” sayfa 99](#)

Kanal adını, anasistem adını ve kapı numarasını, MQEnvironment sınıfındaki alanlar olarak ya da MQQueueManager nesnesinin özellikleri olarak belirleyebilirsiniz.

MQEnvironment sınıfındaki alanları ayarluyorsanız, bunlar, bir özellikler HASH çizelgesi tarafından geçersiz kılınan durumlar dışında, tüm uygulamanıza uygulanır. MQEnvironment 'da kanal adını ve anasistem adını belirtmek için aşağıdaki kodu kullanın:

```
MQEnvironment.hostname = "host.domain.com";
MQEnvironment.channel = "java.client.channel";
```

Bu, bir **MQSERVER** ortam değişkeni ayarlamaya eşdeğerdir:

```
"java.client.channel/TCP/host.domain.com".
```

By default, the Java clients attempt to connect to an IBM MQ listener at port 1414. Farklı bir kapı belirtmek için aşağıdaki kodu kullanın:

```
MQEnvironment.port = nnnn;
```

burada nnnn gerekli kapı numarasıdır

Özellikleri, yaratılışındaki bir kuyruk yöneticisi nesnesine geçerseniz, bu nesne yalnızca o kuyruk yöneticisine uygulanır. Create entries in a Hashtable object with keys of **hostname**, **channel**, and, optionally, **port**, and with appropriate values. Varsayılan kapıyı kullanmak için 1414, **port** girdisini atlayabilirsiniz. Özellikler karma çizelgesini kabul eden bir oluşturucu kullanarak MQQueueManager nesnesini yaratın.

## Bir uygulama adı belirleyerek kuyruk yöneticisiyle bağlantı tanımlanması

Bir uygulama, kuyruk yöneticisiyle olan bağlantısını tanıtan bir ad ayarlayabilir. Bu uygulama adı **DISPLAY CONN MQSC/PCF** komutu tarafından gösterilir (alanın adı **APPLTAG** olan yerdir) ya da IBM MQ Explorer **Application Connections** (Uygulama Bağlantıları) görüntüsünde (alanın **App name** olarak adlandırıldığı yerde).

Uygulama adları 28 karakterle sınırlıdır, bu nedenle daha uzun adlar kısaltılır. Bir uygulama adı belirtilmediyse, varsayılan değer olarak bir varsayılan değer sağlanır. Varsayılan ad, çağırana (ana) sınıfa dayalıdır; ancak, bu bilgi yoksa, Java için IBM MQ Client metni kullanılır.

Çağırana sınıfın adı kullanılırsa, gerekiyorsa, önde gelen paket adları kaldırılarak sığaca ayarlanır. Örneğin, çağırılan sınıf `com.example.MainAppise`, tam ad kullanılır, ancak çağırana sınıf `com.example.dictionaryAndThesaurus.multilingual.mainAppise`, bu ad, sınıf adının en uzun birleşimi ve kullanılabilir uzunluğa uyan en sağdaki paket adının en uzun birleşimi olduğu için `multilingual.mainApp` kullanılır.

Sınıf adının kendisi 28 karakterden uzunsa, sığaca kesilir. Örneğin, `com.example.mainApplicationForSecondTestCase`, `mainApplicationForSecondTestolur`.

MQEnvironment sınıfında bir uygulama adı belirlemek için, adı aşağıdaki kodu kullanarak **MQConstants.APPNAME\_PROPERTY** anahtarıyla MQEnvironment.properties hash çizelgesine ekleyin:

```
MQEnvironment.properties.put(MQConstants.APPNAME_PROPERTY, "my_application_name");
```

To set an application name in the properties hash table that is passed to the MQQueueManager constructor, add the name to the properties hash table with a key of **MQConstants.APPNAME\_PROPERTY**.

## Bir IBM MQ istemcisi yapılandırma dosyasında belirtilen özelliklerin geçersiz kılınması

Bir IBM MQ istemcisi yapılandırma dosyası, IBM MQ classes for Java' u yapılandırmak için kullanılan özellikleri de belirtebilir. Ancak, bir IBM MQ MQI client yapılandırma dosyasında belirtilen özellikler yalnızca, bir uygulama istemci kipinde bir kuyruk yöneticisine bağlandığında geçerlidir.

Gerekliyse, IBM MQ yapılandırma kütüğündeki herhangi bir özneliği aşağıdaki yollardan herhangi bir şekilde geçersiz kılabilirsiniz. Seçenekler öncelik sırasına göre gösterilir.

- Yapılandırma özelliği için bir Java sistem özelliği ayarlayın.
- Set the property in the MQEnvironment.properties map.
- Java5 ve sonraki yayın düzeylerinde, bir sistem ortam değişkeni ayarlayın.

Bir IBM MQ istemcisi yapılandırma dosyasında yalnızca aşağıdaki öznelikler IBM MQ classes for Java ile ilgilidir. Diğer öznelikleri belirtmiş ya da geçersiz kıyorsanız, bu herhangi bir etkisi olmaz.

Stanza	Öznelik
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	ExitsDefaultYolu
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	ExitsDefaultPath64
<a href="#">ClientExitİstemci yapılandırma dosyasının yol gösterişi</a>	JavaExitsClasspath
<a href="#">İstemci yapılandırma dosyasınınMessageBuffer kısmı</a>	MaximumSize
<a href="#">İstemci yapılandırma dosyasınınMessageBuffer kısmı</a>	PurgeTime
<a href="#">İstemci yapılandırma dosyasınınMessageBuffer kısmı</a>	UpdatePercentage
<a href="#">İstemci yapılandırma dosyasının TCP stanzası</a>	ClntRcvBufSize
<a href="#">İstemci yapılandırma dosyasının TCP stanzası</a>	ClntSndBufSize
<a href="#">İstemci yapılandırma dosyasının TCP stanzası</a>	Bağlantı_Zamanaşımı
<a href="#">İstemci yapılandırma dosyasının TCP stanzası</a>	KeepAlive

*IBM MQ classes for Java içinde bir kuyruk yöneticisine bağlanma*

MQQueueManager sınıfının yeni bir örneğini oluşturarak bir kuyruk yöneticisine bağlanın. Bağlantı kesme () yöntemini çağırarak, kuyruk yöneticisinden bağlantıyı kesin.

Artık MQQueueManager sınıfının yeni bir örneğini oluşturarak bir kuyruk yöneticisine bağlanmaya hazırsınız:

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Kuyruk yöneticisinden bağlantıyı kesmek için, kuyruk yöneticisinde bağlantı kesme () yöntemini çağırın:

```
queueManager.disconnect();
```

Bağlantı kesme yöntemini çağırırsanız, o kuyruk yöneticisi aracılığıyla eriştiğiniz tüm açık kuyruklar ve işlemler kapatılır. Ancak, bu kaynakları kullanmayı bitirdiğinizde, bu kaynakları açık bir şekilde kapatmak için iyi bir programlama uygulamasıdır. Bunu yapmak için, ilgili nesnelere ilgili close () yöntemini kullanın.

Kuyruk yöneticisiyle ilgili kesinleştirme () ve backout () yöntemleri, yordamsal arabirimle birlikte kullanılan MQCMIT ve MQBACK çağrılarına eşdeğerdir.

### *Using a client channel definition table with IBM MQ classes for Java*

Bir IBM MQ classes for Java istemcisi uygulaması, istemci kanal tanımlama çizelgesinde (CCDT) saklanan istemci bağlantı kanalı tanımlarını kullanabilir.

As an alternative to creating a client connection channel definition by setting certain fields and environment properties in the `MQEnvironment` class or passing them to an `MQQueueManager` in a properties hash table, an IBM MQ classes for Java client application can use client connection channel definitions that are stored in a client channel definition table. Bu tanımlar, IBM MQ Script (MQSC) komutları ya da IBM MQ Programmable Command Format (PCF) komutları ya da IBM MQ Explorer kullanılarak yaratılır.

Uygulama bir `MQQueueManager` nesnesi yarattığında, IBM MQ classes for Java istemcisi uygun bir istemci bağlantısı kanal tanımlaması için istemci kanal tanımlama çizelgesini arar ve bir MQI kanalını başlatmak için kanal tanımlamasını kullanır. İstemci kanal tanımlama tabloları ve nasıl oluşturulacağı hakkında daha fazla bilgi için bkz. [İstemci kanal tanımlama tablosu](#).

Bir istemci kanal tanımlama çizelgesi kullanmak için, uygulamanın önce bir URL nesnesi yaratması gerekir. URL nesnesi, istemci kanal tanımlama çizelgesini içeren dosyanın adını ve yerini tanımlayan ve dosyanın nasıl erişilebileceğini belirten bir URL adresi (uniform resource locator; URL) yerleştirir.

Örneğin, `ccdt1.tab` dosyası bir istemci kanal tanımlama çizelgesi içeriyorsa ve uygulamanın çalıştığı sistemde saklandıysa, 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 çalışmakta olduğu sistemden farklı bir sistemde saklandığı varsayıldığını varsayalım. Dosyaya FTP iletişim kuralı kullanılarak erişilebildiyse, uygulama aşağıdaki şekilde bir URL nesnesi yaratabilir:

```
java.net.URL chanTab2 = new URL("ftp://ftp.server/admdata/ccdt2.tab");
```

Uygulama bir URL nesnesi yarattıktan sonra, uygulama, bir URL nesnesini parametre olarak alan oluşturuculardan birini kullanarak bir `MQQueueManager` nesnesi yaratabilir. Örnek:

```
MQQueueManager mars = new MQQueueManager("MARS", chanTab2);
```

Bu deyim, IBM MQ classes for Java istemcisinin `chanTab2URL` nesnesi tarafından tanımlanan istemci kanal tanımlama çizelgesine erişmesini sağlar, uygun bir istemci bağlantısı kanalı tanımlaması için çizelgeyi arar ve sonra MARS adlı kuyruk yöneticisine bir MQI kanalı başlatmak için kanal tanımlamasını kullanır.

Bir uygulama istemci kanal tanımlama çizelgesi kullanıyorsa, aşağıdaki noktalara dikkat edin:

- Uygulama, parametre olarak bir URL nesnesi alan bir oluşturucu kullanan bir `MQQueueManager` nesnesi yarattığında, `MQEnvironment` sınıfında bir alan olarak ya da bir ortam özelliği olarak herhangi bir kanal adı ayarlanmamalıdır. Bir kanal adı ayarlandıysa, IBM MQ classes for Java istemcisi bir `MQException`atar. Kanal adını belirten alan ya da ortam özelliği, değeri boş değerli, boş bir dizgi ya da tüm boşluk karakterleri içeren bir dizgi ise, bu alan ya da ortam özelliğinin ayarlanabileceği varsayılır.
- `MQQueueManager` oluşturucudaki **`queueManagerName`** parametresi aşağıdaki değerlerden birine sahip olabilir:
  - Kuyruk yöneticisinin adı
  - Bir yıldız işareti (\*) ve ardından bir kuyruk yöneticisi grubu adı gelir.
  - Yıldız işareti (\*)
  - Boş değerli, boş bir dizgi ya da tüm boşluk karakterlerini içeren bir dizgi

Bu deęerler, Message Queue Interface (MQI) kullanan bir istemci uygulaması tarafından yayınlanan bir MQCONN çağrısındaki **QMgrName** parametresi için kullanılacak deęerlerdir. Bu deęerlerin anlamı hakkında daha fazla bilgi için bkz. "[Message Queue Interface-Genel Bakış](#)" sayfa 760.

Uygulamanız bağlantı havuzlama kullanıyorsa, bkz. "[IBM MQ classes for Java'deki varsayılan bağlantı havuzunu denetleme](#)" sayfa 369.

- IBM MQ classes for Java istemcisi, istemci kanal tanımlama çizelgesinde uygun bir istemci bağlantısı kanalı tanımlaması bulunduęunda, bu kanal tanımlamasından alınan bilgileri yalnızca bir MQI kanalı başlatmak için kullanır. Uygulamanın MQEnvironment sınıfında ayarlanmış olabileceęi kanallarla ilgili alanlar ya da ortam özellikleri yoksayılr.

Transport Layer Security (TLS) kullanıyorsanız, özellikle ařaęıdaki noktalara dikkat edin:

- Bir MQI kanalı TLS 'yi yalnızca istemci kanal tanımlama çizelgesinden alınan kanal tanımlaması, IBM MQ classes for Java istemcisi tarafından desteklenen bir CipherSpec ' in adını belirtiyorsa kullanır.
- İstemci kanalı tanımlama çizelgesi, sertifika iptal listelerini (CRL ' ler) bulunduran LDAP (Lightweight Directory Access Protocol; Temel Dizin Eriřimi Protokolü) sunucularının konularıyla ilgili bilgileri de içerir. IBM MQ classes for Java istemcisi, CRL ' leri tutan LDAP sunucularına eriřmek için yalnızca bu bilgileri kullanır.
- İstemci kanalı tanımlama çizelgesi bir OCSP yanıtlayıcıya iliřkin yeri de içerebilir. IBM MQ classes for Java , bir istemci kanal tanımlama çizelgesi dosyasında OCSP bilgilerini kullanamaz. Ancak, OCSP ' yi [Using Online Certificate Protocol](#)(Çevrimiçi Sertifika İletişim Kuralı) bölümünde açıklandığı gibi yapılandırabilirsiniz.

TLS 'yi bir istemci kanal tanımlama çizelgesiyle kullanmaya iliřkin ek bilgi için [Bir MQI kanalının TLS' yi kullandığını belirtme](#) başlıklı konuya bakın.

Kanal çıkıřlarını kullanıyorsanız ařaęıdaki noktalara da dikkat edin:

- Bir MQI kanalı, kanal çıkıřlarını ve dięer yöntemler kullanılarak belirlenen verileri kullanarak, istemci kanalı tanımlama çizelgesinden alınan kanal tanımlaması tarafından belirlenen kanal çıkıřlarını ve iliřkili kullanıcı verilerini kullanır.
- İstemci kanalı tanımlama çizelgesinden alınan bir kanal tanımlaması, Java, C ya da C + + içinde yazılan kanal çıkıřlarını belirtebilir. Java ' ta kanal çıkıřı nasıl yazılacağı hakkında daha fazla bilgi için bkz. "[IBM MQ classes for Java' ta bir kanal çıkıřı oluřturma](#)" sayfa 362. Bir kanal çıkıřının dięer dillerde nasıl yazılabileceęi hakkında daha fazla bilgi için bkz. "[Using channel exits not written in Java with IBM MQ classes for Java](#)" sayfa 366.

*IBM MQ classes for Java istemci bağlantıları için bir kapı aralıęı belirtme*

Bir uygulamanın iki yönlü olarak bağlanabileceęi bir kapı ya da kapı aralıęı belirtebilirsiniz.

Bir IBM MQ classes for Java uygulaması istemci kipinde bir IBM MQ kuyruk yöneticisine bağlanmayı denedięinde, bir güvenlik duvarı yalnızca belirtilen kapılardan ya da kapı aralıęından kaynaklanan bağlantılara izin verebilir. Bu durumda, uygulamanın bağlanabileceęi bir kapı ya da kapı aralıęı belirtebilirsiniz. Kapı (lar) ı ařaęıdaki řekillerde belirtebilirsiniz:

- MQEnvironment sınıfındaki localAddressSetting 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 oluřturabildięinizde, "192.0.2.0(2000,3000)" deęerine sahip LOCAL\_ADDRESS\_PROPERTY içeren bir özellikler hashtable öęesini geçirebilirsiniz.

Bu örneklerin her birinde, uygulama daha sonra bir kuyruk yöneticisine bağlandığında, uygulama, 192.0.2.0(2000)- 192.0.2.0(3000) aralıęındaki yerel bir IP adresine ve kapı numarasına bağlanır.

Birden çok ağ arabirimine sahip bir sistemde, `localAddress`(yerel adres) ayar alanını ya da `CMQC.LOCAL_ADDRESS_PROPERTY`, bir bağlantı için hangi ağ arabiriminin kullanılması gerektiğini belirtmek için.

Kapı aralığını sınırladığınızda bağlantı hataları ortaya çıkabilir. Bir hata ortaya çıkarsa, IBM MQ neden kodu `MQRC_Q_MGR_NOT_AVAILABLE` ve şu 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 erişme

Kuyruklara, konulara ve süreçlere erişmek için `MQQueueManager` sınıfının yöntemlerini kullanın. `MQOD` (nesne tanımlayıcı yapısı), bu yöntemlerin parametrelerine daraltılır.

## **Kuyruklar**

Bir kuyruğu açmak için, `MQQueueManager` sınıfının `accessQueue` yöntemini kullanabilirsiniz. Örneğin, `queueManager` adlı bir kuyruk yöneticisinde aşağıdaki kodu kullanın:

```
MQQueue queue = queueManager.accessQueue("qName", CMQC.MQOO_OUTPUT);
```

`accessQueue` yöntemi, `MQQueue` sınıfını yeni bir nesne döndürür.

Kuyruğu kullanmayı bitirdiğinizde, aşağıdaki örnekteki gibi kapatmak için `close()` yöntemini kullanın:

```
queue.close();
```

Ayrıca, `MQQueue` oluşturucusunu kullanarak bir kuyruk da yaratabilirsiniz. Parametreler, bir kuyruk yöneticisi değiştirmesinin eklenmesiyle tam olarak `accessQueue` yöntemi ile aynıdır. Örneğin:

```
MQQueue queue = new MQQueue(queueManager,
                             "qName",
                             CMQC.MQOO_OUTPUT,
                             "qMgrName",
                             "dynamicQName",
                             "altUserID");
```

Kuyruklar yaratırken bir dizi seçenek belirleyebilirsiniz. Bunlara ilişkin ayrıntılar için `Class.com.ibm.mq.MQQueue` başlıklı konuya bakın. Bu şekilde bir kuyruk nesnesi oluşturulması, kendi `MQQueue` alt sınıflarınızı yazmanızı sağlar.

## **Konular**

Benzer şekilde, `MQQueueManager` sınıfının `accessTopic` yöntemini kullanarak bir konuyu açabilirsiniz. Örneğin, `queueManager` adlı bir kuyruk yöneticisinde, bir abone ve yayıncı 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, kapatmak için `close()` yöntemini kullanın.

Ayrıca, MQTopic oluşturucusunu kullanarak bir konu da yaratabilirsiniz. Parametreler, bir kuyruk yöneticisi değiştirgesinin eklenmesiyle tam olarak accessTopic yöntemiyle aynıdır. Örneğin:

```
MQTopic subscriber = new
    MQTopic(queueManager, "TOPICSTRING", "TOPICNAME",
    CMQC.MQTOPIC_OPEN_AS_SUBSCRIPTION, CMQC.MQSO_CREATE);
```

Konular oluştururken bir dizi seçenek belirtebilirsiniz. Bunlara ilişkin ayrıntılar için [Class com.ibm.mq.MQTopic](#) başlıklı konuya bakın. Bu şekilde bir konu nesnesi oluşturmak, kendi MQkonusuyla ilgili alt sınıflarınızı yazmanızı sağlar.

Bir konunun yayınlanması için ya da abonelik için açılması gerekir. MQQueueManager sınıfının sekiz accessTopic yöntemi vardır ve Konu sınıfı sekiz oluşturucuda bulunur. Her durumda, bunlardan dördünün bir **destination** parametresi ve dört tanesi de **subscriptionName** parametresine sahiptir (ikisi de dahil olmak üzere). Bunlar yalnızca abonelikler için konuyu açmak için kullanılabilir. Kalan iki yöntemde bir **openAs** parametresi vardır ve konu, **openAs** parametresinin değerine bağlı olarak yayınlanmak ya da abonelik için açılabilir.

Kalıcı bir abone olarak bir konu oluşturmak için, MQQueueManager sınıfının bir accessTopic yöntemini ya da abonelik adını kabul eden bir MQTopic oluşturucusu kullanın ve her iki durumda da CMQC.MQSO\_DURABLE seçeneğini ayarlayın.

## Süreçler

Bir sürece erişmek için, MQQueueManager'ın accessProcess yöntemini kullanın. Örneğin, queueManageradlı bir kuyruk yöneticisinde, bir 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 sınıfı yeni bir nesne döndürür.

İşlem nesnesini kullanmayı bitirdiğinizde, aşağıdaki örnekteki gibi kapatmak için close () yöntemini kullanın:

```
process.close();
```

Ayrıca, MQProcess oluşturucusunu kullanarak bir süreç de yaratabilirsiniz. Parametreler, bir kuyruk yöneticisi değiştirgesinin eklenmesiyle tam olarak accessProcess yöntemiyle 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 sınıfı tarafından temsil edilir. İletiler, MQQueue ve MQTopic alt sınıfları olan MQDestination sınıfı yöntemlerini kullanarak yerleştirir alır.

MQDestination sınıfının put () yöntemini kullanarak iletileri kuyruklara ya da konulara yerleştirin. MQDestination sınıfının get () yöntemini kullanarak, kuyruklardan ya da konulardan ileti alır. Yordamsal arabirimden farklı olarak, burada MQPUT ve MQGET, Java programlama dili MQPUT ve MQMessage 'ın bayt dizilerini alır ve bu MQMessage sınıfının somut örneklerini alır. MQMessage sınıfı, gerçek ileti verilerini içeren veri arabelleğini, tüm MQMD (ileti tanımlayıcı) parametrelerini ve bu iletiyi açıklayan ileti özelliklerini içerir.



Yeni bir ileti oluşturmak için, MQMessage sınıfının yeni bir örneğini oluşturun ve ileti arabelleğindeki verileri yerleştirmek için writeXXX yöntemlerini kullanın.

Yeni ileti eşgörünümü yaratıldığında, tüm MQMD parametreleri otomatik olarak varsayılan değerlerine ayarlanır ( MQMD için ilk değerler ve dil bildirimleri' da tanımlandığı gibi). MQDestination 'ın put () yöntemi, değiştirge olarak MQPutMessageSeçenekleri sınıfının bir örneğini de alır. Bu sınıf MQPMO yapısını temsil eder. Aşağıdaki örnek bir ileti yaratır ve bunu 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 örneği döndürür. Ayrıca, parametre olarak MQGetMessageOptions sınıfının bir eşgörünümünü alır. Bu sınıf MQGMO yapısını temsil eder.

get () yöntemi, gelen iletiye sığması için iç arabelleğindeki büyüklüğü otomatik olarak ayarlandığından, ileti boyutu üst sınırı belirtmeniz gerekmez. Döndürülen iletteki verilere erişmek için MQMessage sınıfının readXXX yöntemlerini kullanın.

Aşağıdaki örnekte, kuyruktan iletinin nasıl alacacağı gösterilmektedir:

```
// 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, *encoding* üye değişkenini ayarlayarak değiştirebilirsiniz.

*characterSet* adlı üye değişkenini ayarlayarak dizgileri okumak ve yazmak için kullanılacak karakter kümesini değiştirebilirsiniz.

Ek bilgi için [“MQMessage sınıfı” sayfa 677](#) başlıklı konuya bakın.

**Not:** MQMessage 'ın writeUTF() yöntemi, içerdiği Unicode baytların yanı sıra dizginin uzunluğunu da otomatik olarak kodlar. İletiniz başka bir Java programı tarafından okunacağı zaman ( readUTF() kullanılarak), dizilim bilgileri göndermenin en kolay yoludur.

*IBM MQ classes for Java' ta kalıcı olmayan iletilerin performansının geliştirilmesi*

İletilere göz atılırken ya da bir istemci uygulamasından kalıcı olmayan iletileri tüketirken performansı artırmak için *devamını okuseçeneğini* kullanabilirsiniz. MQGET ya da zamanuyumsuz tüketimi kullanan istemci uygulamaları, iletiler göz atılırken ya da kalıcı olmayan iletiler tüketirken performans iyileştirmelerinden yararlanacaktır.

İleriye okuma tesisine ilişkin genel bilgiler için bkz. İlgili konu.

IBM MQ classes for Java'ta, CMQC.MQSO\_READ\_AHEAD ve CMQC.MQSO\_NO\_READ\_AHEAD (MQQueue ya da MQTopic nesnesi), ileti tüketicilerinin ve kuyruk tarayıcılarının o nesne üzerinde okuma yazma izni olup olmadığını saptamak için kullanılır.

*IBM MQ classes for Javakomutunu kullanarak iletileri zamanuyumsuz olarak koyma*  
Bir iletiyi zamanuyumsuz olarak koymak için MQPMO\_ASYNC\_RESPONSE ayarlayın.

MQDestination sınıfının put () yöntemini kullanarak iletileri kuyruklara ya da konulara yerleştirdiniz. Bir iletiyi zamanuyumsuz olarak yerleştirmek için, kuyruk yöneticisinden yanıt beklemeden işlemin tamamlanmasına izin vermek için, MQPutMessageSeçenekleri 'nin seçenekler alanında MQPMO\_ASYNC\_RESPONSE ayarlayabilirsiniz. Zamanuyumsuz yerleştirmenin başarısını ya da başarısızlığı saptamak için MQQueueManager.getAsyncDurum çağrısını kullanın.

### **IBM MQ classes for Java içinde yayınla/abone ol**

IBM MQ classes for Java' ta, konu MQTopic sınıfı tarafından gösterilir ve MQTopic.put() yöntemlerini kullanarak bu konuyu yayınlayabilirsiniz.

IBM MQ yayınlama/abone olma hakkında genel bilgi edinmek için bkz. [Yayınlama/abone olma ileti alışverişi](#).

### **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 nesnelere, 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.Iterator gibi 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.List gibi çalışır. MQHeaderIterator gibi, üstbilgi içeriğini ayrıştırır, ancak belirli üstbilgileri aramanızı, yeni üstbilgiler eklemenizi, var olan üstbilgileri kaldırmanızı, üstbilgi alanlarını güncellenizi ve üstbilgi içeriğini bir iletiye geri yazmanızı sağlar. Diğer bir seçenek olarak, boş bir MQHeaderList yaratabilir, 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 Javakullanarak bir iletide tüm üstbilgileri yazdırma*

Bu örnekte, MQHeaderIterator yönetim ortamı, üstbilgileri bir kuyruktan alınan bir MQMessage 'a ayrıştırır. The MQHeader objects returned from the nextHeader() method display their structure and contents when their toString method is invoked.

```
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 Javakullanarak bir iletide üstbilgilerin üzerinden atlanması*

Bu örnekte, MQHeaderIterator ' in skipHeaders() yöntemi, son üstbilgiden hemen sonra ileti okuma imlecini konumlandırır.

```
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 ();
```

#### *Finding the reason code in a dead-letter message using IBM MQ classes for Java*

Bu örnekte, okuma yöntemi, iletiden okuyarak MQDLH nesnesini doldurur. Okuma işleminden sonra, ileti okuma imleci, MQDLH üstbilgi içeriğinden hemen sonra konumlanır.

Kuyruk yöneticisinin ölü harf kuyruğunda, önekli iletiler (MQDLH) öneki vardır. Bu iletilerin nasıl işleneceğine karar vermek için-örneğin, bunları yeniden denemek ya da atmak için-bir ölü-mektup işleme uygulamasının MQDLH içinde 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, doğrudan tek bir adımda doğrudan iletiden kendilerini 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 ());
```

### *Reading and removing the header from a dead-letter message using IBM MQ classes for Java*

Bu örnekte, üstbilgiyi bir ölü-mektup iletisinden kaldırmak için MQDLH kullanılır.

Bir ölü-mektup işleme uygulaması genellikle, neden kodlarının geçici bir hata olduğunu belirtmesi durumunda reddedilen iletileri yeniden gönderecektir. İletiyi yeniden göndermeden önce, MQDLH üstbilgisini kaldırmalıdır.

Bu örnek, aşağıdaki adımları gerçekleştirir (örneğin, örnek koddaki açıklamalara bakın):

1. MQHeaderList , tüm iletiyi okur ve iletiyle karşılaşılan her üstbilgi, listedeki bir öge olur.
2. Ölü-harfli iletiler, ilk üstbilgisi olarak bir MQDLH içerir; bu nedenle, bu ileti üstbilgi listesinin ilk ögesinde bulunabilir. MQHeaderList yapılırsa, MQDLH zaten iletiden doldurulmuştur; bu nedenle okuma yöntemini çağılmaya gerek yoktur.
3. Neden kodu, MQDLH sınıfı tarafından sağlanan getReason() yöntemi kullanılarak ayıklanır.
4. Neden kodu incelendi ve iletiyi yeniden göndermenin uygun olduğunu gösterir. MQDLH, MQHeaderList remove () yöntemi kullanılarak kaldırılır.
5. MQHeaderList , kalan içeriğini yeni bir ileti nesnesine yazar. Yeni ileti, MQDLH dışında özgün iletide her şeyi içeriyor ve bir kuyruğa yazılabilir. Oluşturucuya ve yazma yöntemine **true** bağımsız değişkeni, ileti gövdesinin MQHeaderList içinde tutulacağı ve yeniden dışarı yazıldığını gösterir.
6. Şimdi yeni iletinin ileti tanımlayıcısındaki biçim alanı, MQDLH biçim alanında daha önce bulunan değeri içeriyor. İleti verileri, ileti tanımlayıcısındaki sayısal kodlama ve CCSID ayarlarıyla 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 Javakullanılarak iletinin içeriğinin yazdırılması*

Bu örnekte, üstbilgileri de dahil olmak üzere bir iletinin içeriğini yazdırmak için MQHeaderList kullanılır.

Çıktı, iletinin gövdesinin yanı sıra tüm üstbilgi içeriklerinin bir görünümünü de içerir. MQHeaderList sınıfı, bir kerede tüm üstbilgileri kodu çözer; MQHeaderIterator sınıfı, uygulama denetimi altında her defasında birheadersadımına yol gösterirken de bunları çözer. WebSphere MQ uygulamaları yazılırken 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 örnekte, MQMD sınıfı kullanılarak ileti tanımlayıcı alanları da yazdırılıyor. com.ibm.mq.headers.MQMD sınıfının copyFrom() yöntemi, ileti gövdesinin okunmasından ziyade, MQMessage 'in ileti tanımlayıcı alanlarından üstbilgi nesnesini 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 kullanarak bir iletide belirli bir üstbilgi tipini bulma*

Bu örnek, varsa bir iletide MQRFH2 üstbilgisi bulmak için MQHeaderList ' un indexOf(String) yöntemini kullanmaktadır.

```
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 bir MQRFH2 üstbilgisinin çözümleniyor*

Bu örnek, MQRFH2 sınıfını kullanarak, adlandırılan bir klasörde bilinen bir alan değerine nasıl erişileceğini gösterir.

MQRFH2 sınıfı, yalnızca yapının sabit bölümündeki alanları değil, aynı zamanda NameValueVeri alanı içinde taşınan XML ile kodlanmış klasör içeriklerine de erişmek için bir dizi yol sağlar. Bu örnek, bir MQ JMS iletisinde yanıt kuyruğu adını temsil eden jms klasöründeki Rto alanında bilinen bir alan değerine bu örnekte nasıl erişileceğini gösterir.

```
MQRFH2 rfh = ...  
String value = rfh.getStringFieldValue ("jms", "Rto");
```

Bir MQRFH2 dosyasının içeriğini keşfetmek için (belirli alanları doğrudan istemek yerine), getFolders listesini, alanlar ve diğer klasörler içerebilecek bir klasörün yapısını temsil eden MQRFH2.ElementListesini döndürebilirsiniz. Bir alanın ya da klasörün boş değere ayarlanması, alanı MQRFH2' den kaldırır. NameValueVeri klasörü içeriğini bu şekilde değiştirdiğinizde, StrucLength alanı buna göre otomatik olarak güncellenir.

#### *Reading and writing byte streams other than MQMessage objects using IBM MQ classes for Java*

Bu örnekler, veri kaynağı bir MQMessage nesnesi olmadığı zaman IBM MQ üstbilgi içeriğini ayrıştırmak ve değiştirmek için üstbilgi sınıflarını kullanır.

Veri kaynağı bir MQMessage nesnesinden başka bir şey olsa bile, üstbilgi sınıflarını IBM MQ üstbilgi içeriğini ayrıştırmak ve değiştirmek için 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 kullanabilmeniz anlamına gelir. Bu, farklı kodlamalarda üstbilgi zincirini içeren iletiler için kullanışlıdır.

Ayrıca, diğer veri akışlarından DataInput ve DataOutput nesnelerini, örneğin dosya ya da yuva akışları ya da JMS iletilerinde taşınan bayt dizileri elde edebilirsiniz. The java.io.DataInputStream classes implement DataInput and the java.io.DataOutputStream classes implement DataOutput. 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);
```

The line starting MQHeaderIterator could be replaced with

```
MQDLH dlh = new MQDLH (in, CMQC.MQENC_NATIVE, CMQC.MQCCSI_DEFAULT);
// or any other header type
```

Bu örnek, DataOutputAkımı kullanılarak 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 çalıştığınızda, kodlama ve characterSet bağımsız değişkenleri için doğru değerleri kullanmaya dikkat edin. Üstbilgileri okurken, byte içeriğinin başlangıçta yazıldığı kodlamayı ve CCSID ' yi belirtin. Üstbilgi yazarken, oluşturmak istediğiniz kodlamayı ve CCSID ' yi belirtin. Veri dönüştürme otomatik olarak üstbilgi sınıflarına göre gerçekleştirilir.

#### *IBM MQ classes for Java kullanarak yeni üstbilgi tipleri için sınıflar yaratılması*

IBM MQ classes for Java ile birlikte sağlanmamış üstbilgi tipleri için Java sınıfları yaratabilirsiniz.

To add a Java class representing a new header type that you can use in the same way as any header class supplied with IBM MQ classes for Java, you create a class that implements the MQHeader interface. En basit yaklaşım, com.ibm.mq.headers.impl.Header sınıfını genişletmeniz olmalıdır. Bu örnek, MQTM üstbilgi yapısını gösteren tam işlevli bir sınıf üretir. Her alan için ayrı alıcı ve ayarlayıcı (setter) yöntemleri eklemeniz gerekmez; ancak, üstbilgi sınıfı kullanıcıları için yararlı bir kolaylık sağlar. Alan adı için bir dizgi alan soysal getValue ve setValue yöntemleri, üstbilgi tipinde tanımlanan tüm alanlar için çalışır. Edinilmiş okuma, yazma ve büyüklük yöntemleri, yeni üstbilgi tipinin eşgörünümlerinin okunmasını ve yazılacağını ve üstbilgi büyüklüğünü, alan tanımlamasına dayalı olarak doğru olarak hesaplayacaklarını sağlar. Tip tanımlaması yalnızca bir kez yaratılır, ancak bu üstbilgi sınıfının birçok eşgörünümlü yaratılır. To make the new header definition available for decoding using the MQHeaderIterator or MQHeaderList classes, you would register it using the MQHeaderRegistry. Ancak, MQTM üstbilgi sınıfının zaten bu pakette yer aldığına ve varsayılan kayıt dosyasına kaydedildiğine dikkat edin.

```
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.
}

```

### **PCF iletilerinin IBM MQ classes for Java ile işlenmesi**

Java sınıfları PCF tarafından yapılandırılmış iletiler yaratmak ve ayrıştırmak ve PCF isteklerinin gönderilmesini kolaylaştırmak ve PCF yanıtlarını toplamak için sağlar.

PCFMessage & MQCFGR sınıfları, PCF parametre yapılarının dizilerini temsil eder. Bunlar, PCF parametreleri eklemek ve almak için kolaylık sağlayan yöntemler sağlar.

PCF değiştirge yapıları, MQCFH, MQCFIN, MQCFIN64, MQCFST, MQCFBS, MQCFIL, MQCFIL64 MQCFSL ve MQCFGR sınıflarıyla gösterilir. Bu paylaşımına ilişkin temel işletim arabirimleri şunlardır:

- İleti içeriğini okuma ve yazma yöntemleri: read (), write () ve boyut ()
- Değiştirgelemlerin işlenmesine ilişkin yöntemler: getValue (), setValue (), getParameter () ve diğerleri
- Enumerator yöntemi.nextParameter (), bir MQMessage 'da PCF içeriğini ayrıştırıyor

Bir süzgeç işlevi sağlamak için, sorgu komutlarında PCF süzgeç değiştirgesi kullanılır. Aşağıdaki sınıflarda kapsüllenmiştir:

- MQCFIF-tamsayı süzgeci
- MQCFSF-dizgi süzgeci
- MQCFBF-byte süzgeci

İki aracı sınıfı, PCFAgent ve PCFMessageAgent , bir Kuyruk Yöneticisine, komut sunucusu kuyruğuna ve ilişkili bir yanıt kuyruğuna bağlantıyı yönetmek için sağlar. PCFMessageAgent , PCFAgent 'ı genişletir ve olağan durumda tercihte kullanılır. PCFMessageAgent sınıfı, alınan MQMessages ögesini dönüştürür ve bunları, bir PCFMessage dizisi olarak çağırıcıya geri çevirir. PCFAgent, kullanmadan önce ayrıştırmak zorunda olduğunuz bir MQMessages dizisini döndürür.

### **IBM MQ classes for Java'daki ileti özelliklerinin işlenmesi**

Function calls to process message handles have no equivalent in IBM MQ classes for Java. İleti tanıtıcısı özelliklerini ayarlamak, geri döndürmek ya da silmek için, MQMessage sınıfının yöntemlerini kullanın.

İleti özelliklerine ilişkin genel bilgi için bkz. [“Özellik adları” sayfa 26.](#)

İletilere IBM MQ classes for Java erişimi, MQMessage sınıfından geçer. Bu nedenle Java ortamında ileti tanıtıcıları sağlanmaz ve MQCRTMH, MQDLTMH, MQMHBUF ve MQBUFMH çağrılarının IBM MQ işleviyle eşdeğer değildir.

Yordamsal arabirimde ileti tanıtıcısı özelliklerini ayarlamak için, MQSETMP çağrısını kullanıyorsunuz. IBM MQ classes for Java' ta, MQMessage sınıfının uygun yöntemini kullanın:

- setBooleanözelligi
- setByteözelligi
- setBytesÖzelligi

- setShortözelliği
- setIntözelliği
- setInt2Property
- setInt4Property
- setInt8Property
- setLongözelliği
- setFloatÖzelliği
- setDoubleözelliği
- setStringözelliği
- setObjectözelliği

Bunlar bazen toplu olarak *set\*property* yöntemleri olarak adlandırılır.

Yordamsal arabirimde ileti tanıtıcısı özelliklerinin değerini döndürmek için, MQINQMP çağrısını kullanıyorsunuz. IBM MQ classes for Java' ta, MQMessage sınıfının uygun yöntemini kullanın:

- getBooleanÖzelliği
- getByteÖzelliği
- getBytesÖzelliği
- getShortÖzelliği
- getIntÖzelliği
- getInt2Property
- getInt4Property
- getInt8Property
- getLongÖzelliği
- getFloatÖzelliği
- getDoubleÖzelliği
- getStringÖzelliği
- getObjectÖzelliği

Bunlar bazen toplu olarak *get\*property* yöntemleri olarak adlandırılır.

Yordamsal arabirimde ileti tanıtıcısı özelliklerinin değerini silmek için, MQDLTMP çağrısını kullanıyorsunuz. IBM MQ classes for Java' ta, MQMessage sınıfının deleteProperty yöntemini kullanın.

### **Handling errors in IBM MQ classes for Java**

Handle errors arising from IBM MQ classes for Java using Java try and catch blocks.

Java arabirimindeki yöntemler bir tamamlanma kodu ve neden kodu döndürmez. Bunun yerine, bir IBM MQ çağrısından kaynaklanan tamamlanma kodu ve neden kodu her iki sıfır olmadığında bir kural dışı durum yayınlarlar. This simplifies the program logic so that you do not have to check the return codes after each call to IBM MQ. Programınızın hangi noktalarda hata olma olasılığına karşı karar vereceğine karar verebilirsiniz. Bu noktalarda kodunuzu try ve catch blokları ile çevrebilirsiniz. Örneğin:

```
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 arama neden kodları, [API tamamlama ve neden kodları](#) içinde belgelenir.

Bir IBM MQ classes for Java uygulaması çalışırken yayınlanan kural dışı durumlar da günlüğe yazılır. Ancak bir uygulama, belirli bir neden koduyla ilişkili özel durumları günlüğe kaydedilmesini önlemek için MQException.logExclude() yöntemini çağırabilir. Belirli bir neden koduyla ilişkili birçok kural dışı durumun ortaya atılacağı ve günlüğünün bu kural dışı durumlarla doldurulmamasını beklediğiniz durumlarda bunu yapmak isteyebilirsiniz. Örneğin, uygulamanız bir döngü etrafında her yinelendiğinde bir kuyruktan ileti almayı denerse ve bu girişimlerin çoğu için, kuyrukta uygun bir ileti olmadığını bekliyorsanız, MQRC\_NO\_MSG\_AVAILABLE 'nin günlüğe kaydedilmesinin neden koduyla ilişkili kural dışı durumları önlemek isteyebilirsiniz. If an application has previously prevented exceptions associated with a specific reason code from being logged, it can allow these exceptions to be logged again by calling the method MQException.logInclude().

Bazen neden kodu, hatayla ilişkili tüm ayrıntıları iletmez. Yayınlanan her kural dışı durum için, bir uygulamanın bağlantılı kural dışı durumu denetmesi gerekir. Bağlantılı kural dışı durumun kendisi başka bir bağlantılı kural dışı duruma sahip olabilir ve bu nedenle bağlantılı kural dışı durumlar, özgün temel soruna geri giden bir zincir oluşturur. java.lang.Throwable sınıfının zincirleme kural dışı durum mekanizması kullanılarak bağlantılı bir kural dışı durum uygulandı ve bir uygulama, Throwable.getCause() yöntemini çağırarak bağlantılı bir kural dışı durumu alır. Bir MQException örneği olan bir kural dışı durumdan MQException.getCause(), temeldeki com.ibm.mq.jmqi.JmqiException ve getCause örneğini alır; bu kural dışı durum, hataya neden olan temel java.lang.Exception değerini alır.

### **Getting and setting attribute values in IBM MQ classes for Java**

Birçok ortak öznitelik için getXXX() ve setXXX() yöntemleri sağlanmıştır. Diğerlerine soysal sorma () ve set () yöntemleri kullanılarak erişilebilir.

For many of the common attributes, the classes MQManagedObject, MQDestination, MQQueue, MQTopic, MQProcess, and MQQueueManager contain getXXX() and setXXX() methods. Bu yöntemler, öznitelik değerlerini almanıza ve ayarlamanıza olanak sağlar. MQDestination, MQQueue ve MQTopic için, yöntemlerin yalnızca, nesneyi açtığınızda uygun sorgu ve küme işaretlerini belirttiğinizde işe yaradığını unutmayın.

Daha az ortak öznitelikler için, MQQueueManager, MQDestination, MQQueue, MQTopic, ve MQProcess sınıfları tüm devralırları MQManagedObject adlı bir sınıftan devralır. Bu sınıf, sorgula () ve set () arabirimlerini tanımlar.

Yeni işlecini kullanarak yeni bir kuyruk yöneticisi nesnesi yarattığınızda, bu nesne otomatik olarak sorgulanmak üzere açılır. Bir süreç nesnesine erişmek için accessProcess() yöntemini kullandığınızda, o nesne sorgulamak için otomatik olarak açılır. Bir kuyruk nesnesine erişmek için accessQueue() yöntemini kullandığınızda, bu nesne sorgulamak ya da ayarlama işlemleri için otomatik olarak açılmaz. Bunun nedeni, bu seçeneklerin otomatik olarak eklenmesi bazı uzak kuyruklar tipleriyle sorunlara neden olabilir. Bir kuyrukta sorgula, set, getXXX ve setXXX yöntemlerini kullanmak için, accessQueue() yönteminin openOptions deęiřtirgesinde uygun olarak sorgu ve ayar işaretlerini belirlemeniz gerekir. Aynı durum hedef ve konu nesnelere için de geçerlidir.

Sorgu 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 CharAttrLength deęiřtirgelerine gerek yoktur. Ařađıdaki örnek, bir kuyruğun nasıl bir kuyruğun üzerinde nasıl yapılır gösterileceğini 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' ta çok iş parçacıklı programlar**

Java yürütme ortamı, doğal olarak çok iş parçacıklıdır. IBM MQ classes for Java , bir kuyruk yöneticisi nesnesinin birden çok iş parçacığı tarafından paylaşılmasına izin verir, ancak hedef kuyruk yöneticisine tüm erişimin uyumlulaştırılmasını sağlar.

Çok iş parçacıklı programlar Java' ta kaçınmak için zordur. Bir kuyruk yöneticisine bağlanan ve başlatma sırasında kuyruk açan basit bir programı düşünün. Program ekranda tek bir düğme görüntüler. Bir kullanıcı bu düğmeyi tıklattığında, program kuyruktan bir ileti alır.

Java yürütme ortamı, doğal olarak çok iş parçacıklıdır. Bu nedenle, uygulamanızın kullanıma hazırlanması bir iş parçacığıda gerçekleşir ve düğmenin düğmesine yanıt olarak yürütülen kod ayrı bir iş parçacığıda (kullanıcı arabirimi iş parçacığı) yürütülür.

C tabanlı IBM MQ MQI clientile birlikte, birden çok iş parçacığının çekme noktalarının paylaşımına ilişkin sınırlamalar olduğu için bu sorun bir soruna neden olur. IBM MQ classes for Java , bir kuyruk yöneticisi nesnesinin (ve ilişkili kuyruk, konu ve süreç nesnelerinin) birden çok iş parçacığının paylaşılmasına olanak tanıyarak bu kısıtı yeniden aktarır.

IBM MQ classes for Java uygulaması, belirli bir bağlantı (MQQueueManager nesne eşgörünümü) için, hedef IBM MQ kuyruk yöneticisine tüm erişim erişiminin uyumlulaştırılmasını sağlar. Bir kuyruk yöneticisine çağrı yapmak isteyen bir iş parçacığı, ilgili bağlantı için devam eden diğer tüm çağrılar tamamlanıncaya kadar engellenir. Programınızdaki birden çok iş parçacığının aynı kuyruk yöneticisine eşzamanlı olarak erişmeniz gerekiyorsa, eşzamanlı erişim gerektiren her iş parçacığı için yeni bir MQQueueManager nesnesi yaratın. (Bu, her iş parçacığı için ayrı bir MQCONN çağrısı yayınlamaya eşdeğerdir.)

**Not:** Aynı anda ileti isteyen iş parçacıkları arasında com.ibm.mq.MQGetMessageOptions sınıfı eşgörünümleri paylaşılmamalıdır. Bu sınıfın eşgörünümleri, ilgili MQGET isteği sırasında verilerle güncellenir ve birden çok iş parçacığı nesnenin aynı eşgörünümünde koştuzamanlı olarak çalışırken beklenmeyen sonuçlarla sonuçlanabilir.

## **Using channel exits in IBM MQ classes for Java**

An overview of how to use channel exits in an application using the IBM MQ classes for Java.

The following topics describe how to write a channel exit in Java, how to assign it, and how to pass data to it. Daha sonra, C içinde yazılan kanal çıkışlarının nasıl kullanılacağını ve kanal çıkışlarının sırasını nasıl kullanacaklarını açıklar.

Uygulamanızın kanal çıkış sınıfını yüklemek için doğru güvenlik iznine sahip olması gerekir.

### *IBM MQ classes for Java' ta bir kanal çıkışı oluşturma*

Uygun bir arabirimi gerçekleştiren bir Java sınıfı tanımlayarak kendi kanal çıkışlarınızı sağlayabilirsiniz.

Bir çıkışı gerçekleştirmek için, uygun arabirimi gerçekleştiren yeni bir Java sınıfı tanımlarsınız. com.ibm.mq.exits paketindeki üç çıkış arabirimi tanımlanır:

- WMQSendExit
- WMQReceiveExit
- WMQSecurityExit

**Not:** Kanal çıkışları yalnızca istemci bağlantıları için desteklenir; bağ tanımları bağlantıları için desteklenmez. Örneğin, C içinde yazılmış bir istemci uygulaması kullanıyorsanız, IBM MQ classes for Javadaşında bir Java kanalı çıkışı kullanamazsınız.

Bir bağlantı için tanımlanan TLS şifrelemesi, *bundan sonra* gönderme ve güvenlik çıkışları çağrılmış olarak gerçekleştirilir. Benzer şekilde, şifre çözme işlemi *önce* alma ve güvenlik çıkışları çağrılmadan gerçekleştirilir.

Aşağıdaki örnekte, üç arabirimi gerçekleştiren bir sınıf tanımlanmaktadır:

```
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 çıkışta bir MQCXP nesnesi ve bir MQCD nesnesi geçirilir. Bu nesnelere, yordamsal arabirimde tanımlanan MQCXP ve MQCD yapılarını gösterir.

Yazdığınız herhangi bir çıkış sınıfının bir oluşturucusu olmalıdır. Bu, varsayılan oluşturucu ya da bir dizgi bağımsız değişkeni alan bir oluşturucu olabilir. Bir dizgi sürerse, kullanıcı verileri yaratıldığında çıkış sınıfına aktarılır. Çıkış sınıfı hem varsayılan bir oluşturucu, hem de tek bir bağımsız değişken oluşturucusu içeriyorsa, tek bağımsız değişken oluşturucusunun önceliği vardır.

Gönderme ve güvenlik çıkışlarında, çıkış kodunuz sunucuya göndermek istediğiniz verileri döndürmelidir. Bir alma çıkışı için çıkış kodunuzun, IBM MQ ' un yorumlayacağı değiştirilmiş verileri döndürmesi gerekir.

Olabilecek en basit çıkış gövdesi şunlardır:

```
{ return agentBuffer; }
```

Kuyruk yöneticisini kanal çıkışı içinden kapatmayın.

## Varolan kanal çıkış sınıflarının kullanılması

7.0 sürümünden önceki IBM MQ sürümlerinde, aşağıdaki örnekte olduğu gibi MQSendExit, MQReceiveExit ve MQSecurityExit arabirimlerini kullanarak bu çıkışları uyguladınız. Bu yöntem geçerli olmaya devam eder, ancak geliştirilmiş işlev ve başarımlar için yeni yöntem 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
}
}

```

### Assigning a channel exit in IBM MQ classes for Java

IBM MQ classes for Javakomutunu kullanarak bir kanal çıkışı atayabilirsiniz.

There is no direct equivalent to the IBM MQ channel in IBM MQ classes for Java. Kanal çıkışları bir MQQueueManager' a atanılır. Örneğin, WMQSecurityExit arabirimini gerçekleştiren bir sınıfı tanımlamış olmak için, bir uygulama güvenlik çıkışını şu dört yoldan biriyle kullanabilir:

- Bir MQQueueManager nesnesi yaratmadan önce, sınıfın bir eşgörünümünü MQEnvironment.channelSecurityExit alanına atayarak
- By setting the MQEnvironment.channelSecurityExit field to a string representing the security exit class before creating an MQQueueManager object
- By creating a key/value pair in the properties hashtable passed to MQQueueManager with a key of CMQC.SECURITY\_EXIT\_PROPERTY
- İstemci kanal tanımlama çizelgesi (CCDT) kullanılması

MQEnvironment.channelSecurityExit alanını bir dizgiye ayarlayarak, özellikler hashtable ya da CCDT kullanarak bir anahtar/değer çifti oluşturarak atanacak herhangi bir çıkış, varsayılan bir oluşturucuya yazılmalıdır. Bir sınıfın somut örneği olarak atanan bir çıkışa, uygulamaya bağlı olarak varsayılan bir oluşturucuya gerek yoktur.

Bir uygulama, benzer şekilde bir gönderme ya da alma çıkışını kullanabilir. Örneğin, aşağıdaki kod parçası, daha önce MQEnvironment kullanarak tanımlanan MyMQExitssınıfında uygulanan güvenlik, gönderme ve alma çıkışlarını nasıl kullanacağını gösterir.

```

MyMQExits myexits = new MyMQExits();
MQEnvironment.channelSecurityExit = myexits;
MQEnvironment.channelSendExit = myexits;
MQEnvironment.channelReceiveExit = myexits;
:
MQQueueManager jupiter = new MQQueueManager("JUPITER");

```

Kanal çıkışı atamak için birden çok yöntem kullanılırsa, öncelik sırası aşağıdaki gibidir:

1. Bir CCDT 'nin URL adresi MQQueueManager' a iletilirse, CCDT ' nin içeriği kullanılacak kanal çıkışlarını ve MQEnvironment 'daki çıkış tanımlarını ya da özellik gruplama çizelgesini (hashtable) yoksayılır.
2. CCDT URL iletilmezse, MQEnvironment 'dan çıkış tanımlamaları ve gruplama çizelgesi birleştirilir
  - Aynı çıkış tipi hem MQEnvironment, hem de hashtable içinde tanımlandıysa, hashtable içinde tanım kullanılır.
  - Eşdeğer eski ve yeni çıkış tipleri belirtilirse (örneğin, IBM WebSphere MQ 7.0' dan önceki sürümlerde kullanılan tip çıkış tipi için kullanılabilen sendExit alanı ve herhangi bir gönderme çıkışı için kullanılabilen channelSendÇıkış alanı, eski çıkış yerine yeni çıkış (channelSendExit) kullanılır.




Bir kanal çıkışı dizgi olarak bildirdiyseniz, kanal çıkış programının yerini belirlemek için IBM MQ ' i etkinleştirmeniz gerekir. Uygulamanın çalışmakta olduğu ortama ve kanal çıkış programlarının nasıl paketleneyeceği üzerine bağlı olarak çeşitli şekillerde yapabilirsiniz.

- Bir uygulama sunucusunda çalışan bir uygulama için, dosyaları Çizelge 57 sayfa 365 dizininde gösterilen ya da **exitClasspath** tarafından başvuru JAR dosyalarında paketlenmiş bir dizinde saklamanız gerekir.
- Uygulama sunucusunda çalışmayan bir uygulama için aşağıdaki kurallar geçerli olur:
  - Kanal çıkış sınıflarınız ayrı JAR dosyalarında paketleniyse, bu JAR dosyaları **exitClasspath'** te yer almalıdır.
  - Kanal çıkış sınıflarınız JAR dosyalarında paketlenmediyse, sınıf dosyaları Çizelge 57 sayfa 365 içinde gösterilen dizinde ya da JVM sistem sınıfı yolundaki herhangi bir dizinde ya da **exitClasspath** dizininde saklanabilir.

**exitClasspath** özelliği dört şekilde belirtilebilir. Öncelik sırasına göre, bu yollar aşağıdaki gibidir:

1. The system property com.ibm.mq.exitClasspath (defined on the command line using the -D option)
2. mqclient.ini dosyasının exitPath kısmı
3. A hashtable entry with the key CMQC.EXIT\_CLASSPATH\_PROPERTY
4. MQEnvironment değişkeni **exitClasspath**

Birden çok yolu java.io.File.pathSeparator karakterini kullanarak ayırın.

Çizelge 57. Kanal çıkış programlarına ilişkin dizin	
Platform	Dizin
 AIX	/var/mqm/exits (32 bit kanal çıkış programları)
 Linux	/var/mqm/exits64 (64-bit kanal çıkış programları)
 Solaris	
Windows	kuruluş_data_dzn\çıkışlar

**Not:** *install\_data\_dir* , kuruluş sırasında IBM MQ veri dosyaları için seçtiğiniz dizindir. Varsayılan dizin C:\ProgramData\IBM\MQdizindir.

*IBM MQ classes for Java' taki kanal çıkışlarına veri aktarma*

Kanal çıkışlarına veri aktarabilir ve kanal çıkışlarından uygulamanıza veri döndürebilirsiniz.

## agentBuffer parametresi

Gönderme çıkışı için, *agentBuffer* parametresi, gönderilecek verileri içerir. Bir alma çıkışı ya da güvenlik çıkışı için, *agentBuffer* parametresi az önce alınmış verileri içerir. *agentBuffer.limit ()* ifadesi dizinin uzunluğunu gösterdiğinden, uzunluk parametresine gerek yoktur.

Gönderme ve güvenlik çıkışlarında, çıkış kodunuz sunucuya göndermek istediğiniz verileri döndürmelidir. Bir alma çıkışı için çıkış kodunuzun, IBM MQ ' un yorumlayacağı değiştirilmiş verileri döndürmesi gerekir.

Olabilecek en basit çıkış gövdesi şunlardır:

```
{ return agentBuffer; }
```

Kanal çıkışları, arka diziyeye sahip bir arabellekle çağrılır. En iyi başarıyı elde etmek için, çıkışta yedek diziyeye sahip bir arabellek döndürülmelidir.

## Kullanıcı verileri

If an application connects to a queue manager by setting `channelSecurityExit`, `channelSendExit`, or `channelReceiveExit`, 32 bytes of user data can be passed to the appropriate channel exit class when it is called, using the `channelSecurityExitUserData`, `channelSendExitUserData`, or `channelReceiveExitUserData` fields. Bu kullanıcı verileri kanal çıkış sınıfı için kullanılabilir, ancak çıkışta her çağrıldığında yenilenir. Bu nedenle, kanal çıkışındaki kullanıcı verilerinde yapılan değişiklikler kaybedilir. Bir kanal çıkışındaki verilerde kalıcı değişiklikler yapmak istiyorsanız, MQCXP exitUseralanını kullanın. Çıkışa yönelik çağrılar arasında bu alandaki veriler saklanır.

Uygulama `securityExit`, `sendExit` ya da `receiveExit` ayarlarsa, bu kanal çıkış sınıflarına kullanıcı verisi aktarılabilir.

Bir uygulama, bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, bir istemci bağlantı kanalı tanımlamasında belirtilen kullanıcı verileri, çağrıldığında kanal çıkış sınıflarına geçirilir. İstemci kanal tanımlama çizelgesini kullanma hakkında daha fazla bilgi için bkz. [“Using a client channel definition table with IBM MQ classes for Java” sayfa 349.](#)

*Using channel exits not written in Java with IBM MQ classes for Java*

Bir Java uygulamasından C içinde yazılan kanal çıkış programlarını kullanma.

In IBM WebSphere MQ 7.0, you can specify the name of a channel exit program written in C as a String passed to the `channelSecurityExit`, `channelSendExit`, or `channelReceiveExit` fields in the `MQEnvironment` object or properties Hashtable. Ancak, başka bir dilde yazılmış bir uygulamadaki Java ' ta yazılmış bir kanal çıkışı kullanamazsınız.

Specify the exit program name in the format `library(function)` and ensure that the location of the exit program is specified as described in [Çıkışa giden yol](#).

C içinde bir kanal çıkışı nasıl yazılabilmeye ilişkin bilgi için bkz. [“İleti alışverişi kanallarına ilişkin kanal çıkışı programları” sayfa 993.](#)

## Dış çıkış sınıflarının kullanılması

IBM WebSphere MQ 7.0' dan önceki sürümlerde, Javadişındaki dillerde yazılmış kanal çıkışlarını kullanabilmenize olanak sağlamak için üç sınıf sağlanmıştı:

- MQSecurityExit arabirimini gerçekleştirenMQExternalSecurityExit
- MQSendExit arabirimini gerçekleştirenMQExternalSendExit
- MQReceiveExit arabirimini gerçekleştirenMQExternalReceiveExit

Bu sınıfların kullanımı geçerli olmaya devam eder, ancak yeni yöntem tercih edilir.

Java içinde yazılmamış bir güvenlik çıkışı kullanmak için, önce bir uygulama MQExternalSecurityçıkış nesnesi yaratmış olmalıdır. Belirtilen uygulama, MQExternalSecurityçıkış oluşturucusuna parametre olarak, güvenlik çıkışını içeren kitaplığın adı, güvenlik çıkışa ilişkin giriş noktasının adı ve çağrıldığında güvenlik çıkışa geçirecek kullanıcı verileri. Java içinde yazılmamış olan kanal çıkış programları, [Çizelge 57 sayfa 365](#) içinde gösterilen dizinde depolanır.

*IBM MQ classes for Java' ta bir kanal gönderme ya da alma çıkış dizisi kullanma*

Bir IBM MQ classes for Java uygulaması, art arda çalıştırılan bir kanal gönderme ya da alma çıkışlarını kullanabilir.

Bir uygulama çıkış dizisini kullanmak için, bir uygulama gönderme çıkışlarını içeren bir Liste ya da Dizgi yaratabilir. Bir Liste kullanılırsa, Liste 'nin her ögesi aşağıdakilerden biri olabilir:

- WMQSendExit arabirimini gerçekleştiren kullanıcı tanımlı bir sınıfın somut örneği
- MQSendExit arabirimini gerçekleştiren kullanıcı tanımlı bir sınıfın somut örneği ( Java ' ta yazılan bir gönderme çıkışı için)
- MQExternalSendçıkış sınıfının somut örneği ( Java ' ta bir gönderme çıkışı için yazılmamış)
- MQSendExitZincir sınıfının somut örneği

- Dizgi sınıfının somut örneği

Bir Liste başka bir Liste içeremez.

Uygulama, benzer bir şekilde bir alma işlemi dizisi kullanabilir.

Bir Dizgi kullanılırsa, bu dizgi bir ya da daha çok virgülle ayrılmış çıkış tanımlamasından oluşmalıdır; her biri bir Java sınıfının adı ya da `library (function)` biçiminde bir C programı olabilir.

Daha sonra uygulama, bir `MQQueueManager` nesnesi yaratmadan önce, Liste ya da Dizgi nesnesini `MQEnvironment.channelSendExit` alanına atar.

Çıkışlara aktarılan bilgilerin bağlamı yalnızca çıkışların etki alanı içinde yer alıyor. Örneğin, bir Java çıkışı ve bir C çıkışı zincirlenirse, C çıkışı üzerinde Java çıkışının etkisi olmaz.

## Çıkış zinciri sınıflarının kullanılması

IBM WebSphere MQ 7.0' dan önceki sürümlerde, çıkış sıralarına izin vermek için iki sınıf sağlanmıştır:

- `MQSendExit` arabirimini gerçekleştiren `MQSendExitZinciri`
- `MQReceiveExit` arabirimini gerçekleştiren `MQReceiveExitZinciri`

Bu sınıfların kullanımı geçerli olmaya devam eder, ancak yeni yöntem tercih edilir. Using the IBM MQ Classes for Java interfaces means that your application still has a dependency on `com.ibm.mq.jar`. If the new set of interfaces in the `com.ibm.mq.exits` package are used there is no dependency on `com.ibm.mq.jar`.

Bir uygulama çıkış dizisi kullanmak için, bir uygulama, nesnelere listesini (burada her nesnenin aşağıdakilerden biri olduğu bir liste) oluşturdu:

- `MQSendExit` arabirimini gerçekleştiren kullanıcı tanımlı bir sınıfın somut örneği ( Java ' ta yazılan bir gönderme çıkışı için)
- `MQExternalSendExit` sınıfının somut örneği ( Java ' ta bir gönderme çıkışı için yazılmamış)
- `MQSendExitZincir` sınıfının somut örneği

Uygulama, bu nesne listesini oluşturucuda değiştirge olarak geçirerek bir `MQSendExitChain` nesnesi yarattı. Daha sonra, uygulama `MQQueueManager` nesnesi yaratmadan önce `MQSendExitChain` nesnesini `MQEnvironment.sendExit` alanına atayacaktı.

## IBM MQ classes for Java içinde kanal sıkıştırması

Bir kanalda akan verilerin sıkıştırılıp açılması, kanalın performansını artırabilir ve ağ trafiğini azaltabilir. IBM MQ classes for Java , IBM MQ için yerleşik sıkıştırma işlevini kullanır.

IBM MQ ile birlikte verilen 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, bir kanalda veri sıkıştırılmadı. Kanal sıkıştırması ( IBM MQ içinde nasıl uygulanırsa da dahil olmak üzere) tam açıklaması için bkz. [Veri sıkıştırması \(COMMSG\)](#) ve [Üstbilgi sıkıştırması \(COMMPHDR\)](#).

IBM MQ classes for Java uygulaması, bir `java.util.Collection` nesnesi yaratarak bir istemci bağlantısında üstbilgi ya da ileti verileri sıkıştırılması için kullanılacak teknikleri belirtir. Her sıkıştırma tekniği, kaynak grubundaki bir Tamsayı nesnesidir ve uygulamanın kaynak grubuna sıkıştırma tekniklerini eklediği sıra, istemci bağlantısı başlatıldığında, sıkıştırma tekniklerinin kuyruk yöneticisiyle kararlaştırıldığı sıradır. Daha sonra uygulama, veri toplama nesnesini `hdrCompListe` alanına, üstbilgi verileri için ya da `msgCompListe` alanını, ileti verileri için `MQEnvironment` sınıfındaki atayabilir. Uygulama hazır olduğunda, bir `MQQueueManager` nesnesi yaratarak istemci bağlantısını başlatabilir.

Aşağıdaki kod parçaları açıklanan yaklaşımı gösterir. İlk kod parçası, üstbilgi veri sıkıştırmasının nasıl gerçekleştirileceğini gösterir:

```
Collection headerComp = new Vector();
headerComp.add(new Integer(CMQXC.MQCOMPRESS_SYSTEM));
:
MQEnvironment.hdrCompList = headerComp;
```

```
:  
MQQueueManager qMgr = new MQQueueManager(QM);
```

İkinci kod parçası, ileti veri sıkıştırmasının nasıl gerçekleştirileceğini 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 ZLIBGHH ' de (SLE) kararlaştırılır. Seçilen sıkıştırma tekniği, MQQueueManager nesnesinin geçerlik süresi boyunca değiştirilemez.

Bir istemci bağlantısında hem istemci, hem de kuyruk yöneticisi tarafından desteklenen üstbilgi ve ileti verilerine ilişkin sıkıştırma teknikleri, bir MQChannelDefinition nesnesinin hdrCompListesi ve msgCompListesi alanlarındaki kaynak grupları olarak bir kanal çıkışa geçirilir. Bir istemci bağlantısında üstbilgi ve ileti verileri sıkıştırılması için kullanılmakta olan gerçek teknikler, MQChannelExit nesnesinin CurHdrSıkıştırması ve CurMsgSıkıştırma alanlarında bir kanal çıkışa geçirilir.

Bir istemci bağlantısında sıkıştırma kullanılıyorsa, veriler, kanal gönderme çıkışları işlendikten sonra, kanal alma çıkışları işlenmeden ve çıkarılmadan önce sıkıştırılır. Gönderme ve alma çıkışlarına aktarılan veriler, bu nedenle sıkıştırılmış durumda.

Sıkıştırma tekniklerini belirtme ve hangi sıkıştırma tekniklerinin kullanılabilir olduğuna ilişkin ek bilgi için [Class com.ibm.mq.MQEnvironment](#) ve [Interface com.ibm.mq.MQCbaşlıklı](#) konuya bakın.

### **IBM MQ classes for Java içinde bir TCP/IP bağlantısının paylaşılması**

Tek bir TCP/IP bağlantısını paylaşmak için bir MQI kanalının birden çok eşgörünümü yapılabilir.

IBM MQ classes for Java' ta, tek bir TCP/IP bağlantısını paylaşabilen etkileşimlerin sayısını denetlemek için MQEnvironment.sharingConversations değişkenini kullanıyorsunuz.

SHARECNV özneliği, bağlantı paylaşımına yönelik en iyi bir çalışma yaklaşımıdır. Therefore when a SHARECNV value greater than 0 is used with the IBM MQ classes for Java it is not guaranteed that a new connection request will always share an already established connection.

### **IBM MQ classes for Java içinde bağlantı havuzu oluşturma**

IBM MQ classes for Java , yedek bağlantıların yeniden kullanılmak üzere havuza gönderilmesine izin verir.

IBM MQ classes for Java , IBM MQ kuyruk yöneticileriyle birden çok bağlantıyla uğraşan uygulamalar için ek destek sağlar. Bir bağlantı artık gerekmediği zaman, onu yok etmek yerine, havuza gönderilebilir ve daha sonra yeniden kullanılabilir. Bu, rasgele kuyruk yöneticilerine dizisel olarak bağlanan uygulamalar ve ara katman yazılımları için önemli bir başarımlar geliştirmesi sağlayabilir.

IBM MQ , varsayılan bağlantı havuzu sağlar. Uygulamalar, bu bağlantı havuzunu MQEnvironment sınıfından belirteçler kaydederek ve kayıttan kaldırılarak etkinleştirebilir ya da devre dışı bırakabilir. pool, IBM MQ classes for Java bir MQQueueManager nesnesi oluşturduğunda etkin ise, bu varsayılan havuzda arama yapar ve uygun bağlantıyı yeniden kullanır. Bir MQQueueManager.disconnect () çağrısı gerçekleşirse, temeldeki bağlantı havuza geri döndürülür.

Diğer bir seçenek olarak, uygulamalar belirli bir kullanım için bir MQSimpleConnectionManager bağlantı havuzu oluşturabilirler. Daha sonra, uygulama bir MQQueueManager nesnesinin yapımı sırasında havuzu belirtebilir ya da varsayılan bağlantı havuzu olarak kullanılmak üzere bu havuzu MQEnvironment 'a geçirebilir.

Bağlantıların çok fazla kaynak kullanmasını önlemek için, bir MQSimpleConnectionYöneticisi nesnesinin işleyebileceği toplam bağlantı sayısını sınırlayabilir ve bağlantı havuzunun büyüklüğünü sınırlayabilirsiniz. Bir JVM içindeki bağlantılar için çakışan talepler varsa, bu sınırların ayarlanması yararlı olur.

Varsayılan olarak, getMaxConnections () yöntemi sıfır değerini döndürür; bu da, MQSimpleConnectionManager nesnesinin işleyebileceği bağlantı sayısı için bir sınır olmadığı anlamına gelir. setMaxConnections () yöntemini kullanarak bir sınır ayarlayabilirsiniz. Bir sınır ayarlayıp sınıra



ulaşılırsa, daha fazla bağlantı için bir istek, MQRC\_MAX\_CONNS\_IMlim\_REACHED neden koduyla bir MQException yayınlanmasına neden olabilir.

*IBM MQ classes for Java* içindeki varsayılan bağlantı havuzunu denetleme  
Bu örnek, varsayılan bağlantı havuzunun nasıl kullanılacağını gösterir.

Aşağıdaki örnek uygulamayı göz önünde bulundurun: MQApp1:

```
import com.ibm.mq.*;
public class MQApp1
{
    public static void main(String[] args) throws MQException
    {
        for (int i=0; i<args.length; i++) {
            MQQueueManager qmgr=new MQQueueManager(args[i]);
            :
            : (do something with qmgr)
            :
            qmgr.disconnect();
        }
    }
}
```

MQApp1 , komut satırından yerel kuyruk yöneticilerinin listesini alır, her bir sırayla bağlanır ve bazı işlemler gerçekleştirir. Ancak, komut satırı aynı kuyruk yöneticisini birçok kez listelediğinde, bu bağlantıyı yalnızca bir kez bağlamak 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() ögesini kullanın.

Aşağıdaki örnek uygulama ( MQApp2), işlevsel olarak MQApp1 ile aynı, 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, MQEnvironment içeren bir MQPoolToken nesnesini kaydettirerek varsayılan bağlantı havuzunu etkinleştirir.

MQQueueManager oluşturucusu, bu havuzu uygun bir bağlantı için arar ve var olan bir bağlantı bulamazsa kuyruk yöneticisiyle bağlantı yaratır. qmgr.disconnect() çağırısı, daha sonra yeniden kullanılmak üzere havuzla bağlantıyı döndürür. Bu API çağrıları örnek uygulamayla ( MQApp1) aynıdır.

Vurgulu görüntülenen ikinci satır, havuzda saklanan kuyruk yöneticisi bağlantılarını yok eden varsayılan bağlantı havuzunu devre dışı bırakır. Bunun nedeni, uygulamanın havuzdaki canlı kuyruk yöneticisi bağlantılarıyla sona erdirileceği için önemlidir. Bu durum, kuyruk yöneticisi günlüklerinde görüntülenecek hatalara neden olabilir.

Bir uygulama bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, MQQueueManager oluşturucusu, önce uygun bir istemci bağlantı kanalı tanımlaması için çizelgeyi arar. Bir bağlantı bulunursa, oluşturucu, kanal için kullanılacak bir bağlantı için varsayılan

bağlantı havuzunu arar. Oluşturucu havuzda uygun bir bağlantı bulamazsa, sonraki uygun istemci bağlantı kanalı tanımlaması için istemci kanal tanımlama çizelgesinde arama yapar ve daha önce açıklandığı gibi devam eder. Oluşturucu, istemci kanal tanımlama çizelgesini aramasını tamamlarsa ve havuzda uygun bir bağlantı bulamazsa, oluşturucu, çizelgenin ikinci bir aramasını başlatır. Bu arama sırasında oluşturucu, her bir uygun istemci bağlantı kanalı tanımlaması için yeni bir bağlantı yaratmayı dener ve bu bağlantı, yaratmayı başardığı ilk bağlantıyı kullanır.

Varsayılan bağlantı havuzu, en çok on kullanılmamış bağlantıyı saklar ve kullanılmayan bağlantıları en fazla beş dakika etkin tutar. Uygulama bunu değiştirebilir (ayrıntılar için [“IBM MQ classes for Java’inde farklı bir bağlantı havuzu sağlama”](#) sayfa 371 konusuna bakın).

Bir MQPoolToken sağlamak için MQEnvironment 'ı kullanmak yerine, uygulama kendi yapısını oluşturabilir:

```
MQPoolToken token=new MQPoolToken();
MQEnvironment.addConnectionPoolToken(token);
```

Some applications or middleware vendors provide subclasses of MQPoolToken in order to pass information to a custom connection pool. Bağlantı havuzuna ek bilgi aktarılabilmesi için, bunlar yaratılabilir ve bu şekilde addConnectionPoolToken() yoluna geçebilirler.

*IBM MQ classes for Java’inde varsayılan bağlantı havuzu ve birden çok bileşen*

This example shows how to add or remove MQPoolTokens from a static set of registered MQPoolToken objects.

MQEnvironment, kayıtlı bir MQPoolToken nesnesi kümesini içerir. Bu kümeden MQPoolTokens eklemek ya da kaldırmak için aşağıdaki yöntemleri kullanın:

- MQEnvironment.addConnectionPoolToken()
- MQEnvironment.removeConnectionPoolToken()

Bir uygulama, bağımsız olarak var olan birçok bileşenden oluşabilir ve kuyruk yöneticisi kullanarak iş gerçekleştirir. Böyle bir uygulamada, her bileşen, yaşam süresi için MQEnvironment kümesine bir MQPoolToken eklemelidir.

Örneğin, örnek uygulama MQApp3 on iş parçacığı yaratır ve her birini başlatır. Her iş parçacığı kendi MQPoolToken'ı kaydeder, süre uzunluğunu bekler ve kuyruk yöneticisine bağlanır. İş parçacığı bağlantısını kestikten sonra, kendi MQPoolToken'ı kaldırır.

Varsayılan bağlantı havuzu, MQPoolToken kümesinde en az bir simge olduğunda etkin kalır; bu nedenle, bu uygulama bu uygulamanın süresi boyunca etkin kalır. Uygulamanın, iş parçacıklarının genel denetiminde ana nesne 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 sağlama

Bu örnekte, farklı bir bağlantı havuzu sağlamak için **com.ibm.mq.MQSimpleConnectionManager** sınıfının nasıl kullanılacağı gösterilmektedir.

Bu sınıf, bağlantı havuzlama için temel olanaklar sağlar ve uygulamalar, havuzun davranışını uyarlamak için bu sınıfı kullanabilir.

Bir örnek oluşturulduktan sonra, MQQueueManager oluşturucuda bir MQSimpleConnectionManager belirtilebilir. Daha sonra, oluşturulan MQQueueManager' in temelini oluşturan bağlantıyı yöneten MQSimpleConnectionManager. MQSimpleConnectionManager uygun bir havuza yollanmış bağlantı içeriyorsa, o 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. The connection is then reused during the second call to the MQQueueManager constructor.

İkinci satır MQSimpleConnectionManager 'ı etkinleştirir. The last line disables MQSimpleConnectionManager, destroying any connections held in the pool. Bir MQSimpleConnectionManager, bu bölümde daha sonra açıklanan MODE\_AUTO ' da varsayılan değer olarak.

MQSimpleConnectionYöneticisi, en son kullanılan bir temelde bağlantı ayırır ve bağlantıları en az kullanılan en az kullanılan bir temelde yok eder. Varsayılan değer olarak, beş dakika kullanılmadıysa ya da havuzda kullanılmayan on 'dan fazla bağlantı varsa, bağlantı yok edilir. Bu değerleri değiştirmek için MQSimpleConnectionManager.setTimeout() ögesini çağırarak değiştirebilirsiniz.

MQQueueManager oluşturucuda herhangi bir Bağlantı Yöneticisi sağlanmadığında kullanılacak varsayılan bağlantı havuzu olarak kullanılacak bir MQSimpleConnectionManager 'ı da ayarlayabilirsiniz.

Aşağıdaki uygulama bunu göstermektedir:

```

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.setDefaultConnectionFactory(myConnMan);
MQApp3.main(args);
}
}

```

Kalın çizgiler bir MQSimpleConnectionFactory nesnesi yaratabilir ve yapılandırabilir. Yapılandırma şunları yapar:

- Bir saat kullanılmayan bağlantıları sona erdirir
- myConnMan tarafından yönetilen bağlantı sayısını 75 olarak sınırlar.
- Havuzdaki kullanılmayan bağlantı sayısını 50 ile sınırlar
- MODE\_AUTO ' yı belirler; varsayılan değer bu. Bu, havuzun yalnızca varsayılan bağlantı yöneticisi olması durumunda etkin olduğu ve MQEnvironment tarafından tutulan MQPoolTokens kümesinde en az bir simge olması anlamına gelir.

Yeni MQSimpleConnectionFactory, varsayılan bağlantı yöneticisi olarak ayarlanır.

In the last line, the application calls MQApp3.main(). This runs a number of threads, where each thread uses IBM MQ independently. Bu iş parçacıkları, bağlantı kurdukları sırada myConn(MyConn) olanağını kullanır.

### **IBM MQ classes for Javakullanarak JTA/JDBC eşgüdümü**

IBM MQ classes for Java , MQQueueManager.begin () yöntemini destekler; bu yöntem, IBM MQ ' in JDBC tip 2 ya da JDBC tip 4 uyumlu sürücü sağlayan bir veritabanı için eşgüdümcü olarak işlev görmesini sağlar.

Bu destek tüm altyapılarda kullanılamaz. Hangi platformların JDBC koordinasyonunu desteklediğini denetlemek için bkz. [IBM MQ](#).

XA-JTA desteğini kullanmak için özel JTA anahtar kitaplığını kullanmanız gerekir. Bu kitaplığın kullanılmasına ilişkin yöntem, Windows ya da diğer altyapılardan birini kullanmanıza bağlı olarak değişiklik gösterir.

#### *Windowsüzerinde JTA/JDBC eşgüdümü yapılandırılıyor*

XA kitaplığı, jdbcxxx.dll biçiminde bir DLL adı olarak sağlanır.

The supplied jdbcora12.dll provides compatibility with Oracle 12C, for an IBM MQ for Windows server installation.

Windows sistemlerinde XA kitaplığı tam bir DLL olarak sağlanır. Bu DLL ' in adı jdbcxxx.dll olur; burada xxx , anahtar kitaplığının derlendiği veritabanını gösterir. Bu kitaplık, IBM MQ classes for Java kurulumunuzun java\lib\jdbc ya da java\lib64\jdbc dizininde yer alan bir kitaptır. Anahtar yükleme dosyası olarak da tanımlanan XA kitaplığını kuyruk yöneticisine bildirmeniz gerekir. IBM MQ Explorer' yi kullanın. XA kaynak yöneticisi altında, kuyruk yöneticisi özellikleri panosunda anahtar yükleme dosyasının ayrıntılarını belirtin. Kitaplığın adını yalnızca siz vermelisiniz. Örneğin:

Bir Db2 veritabanı için, SwitchFile alanını şu şekilde ayarlayın: dbcdb2

Bir Oracle veritabanı için, SwitchFile alanını şu şekilde ayarlayın: jdbcora

#### **Notlar:**

1. Oracle 12C is supported by the IBM MQ classes for Java, only on IBM MQ for Windows.
2. The supported version of Oracle 12C is 12.1.0.1.0 Enterprise Edition and future fix packs.
3. 64 bit Windows üzerindeki Oracle 64 bit veritabanları 32 bit Oracle istemcisini gerektirir.
4. Using the IBM MQ classes for Java, IBM MQ can act as a transaction coordinator. Ancak bir JTA stili hareketine katılmanız mümkün değildir.

#### *Windowsdışındaki platformlarda JTA/JDBC koordinasyonunun yapılandırılması*

Nesne dosyaları sağlanır. Sağlanan makefile ' ı kullanarak uygun olanını bağlayın ve yapılandırma dosyasını kullanarak kuyruk yöneticisine bildirin.

Her veritabanı yönetim sistemi için IBM MQ iki nesne dosyası sağlar. 32 bit anahtar kitaplığı yaratmak için bir nesne dosyasını bağlamanız ve 64 bit anahtar kitaplığı yaratmak üzere diğer nesne dosyasını bağlamanız gerekir. Db2 için, her nesne dosyasının adı jdbcd2.o 'dır ve Oracle için her bir nesne dosyasının adı jdbcora.o' dur.

Her nesne dosyasını, IBM MQ ile birlikte verilen uygun makefile kullanarak bağlamanız gerekir. Anahtar kitaplığı, farklı sistemlerde farklı konumlarda saklanabilen diğer kitaplıkları gerektirir. Ancak, anahtar kitaplığı bir setuid ortamında çalışan kuyruk yöneticisi tarafından yüklendiğinden, bir anahtar kitaplığı bu kitaplıkları bulmak için kitaplık yolu ortam değişkenini kullanamaz. Bu nedenle, sağlanan makefile, bir anahtar kitaplığının bu kitaplıkların tam olarak nitelenmiş yol adlarını içermesini sağlar.

Bir anahtar kitaplığı oluşturmak için, aşağıdaki biçimi kullanarak bir **make** komutu girin. 32 bit anahtar kitaplığı yaratmak için, komutu IBM MQ kurulumunuzun /java/lib/jdbc dizinine girin. 64 bit anahtar kitaplığı oluşturmak için, komutu /java/lib64/jdbc dizinine girin.

```
make DBMS
```

Burada *DBMS* , anahtar kitaplığını oluşturduğunuz veritabanı yönetim sistemidir. Geçerli değerler, Db2 için db2 ve Oracle için oracle değerleridir.

Aşağıda bir **make** komutu örneği yer almaktadır:

```
make db2
```

Aşağıdaki noktalara dikkat edin:

- 32 bit uygulamaları çalıştırmak için, kullanmakta olduğunuz her bir veritabanı yönetim sistemi için hem 32 bit hem de 64 bit anahtar kitaplığı yaratmalısınız. 64 bit uygulamaları çalıştırmak için, yalnızca 64 bit anahtar kitaplığı yaratmanız gerekir. Db2 için, her anahtar kitaplığının adı jdbcd2 ' dir ve Oracle için, her anahtar kitaplığının adı jdbcora 'dır. Makefiles, 32 bit ve 64 bit anahtar kitaplıklarının farklı IBM MQ dizinlerinde depolandığından emin olur. 32 bit anahtar kitaplığı /java/lib/jdbc dizininde saklanır ve bir 64 bit anahtar kitaplığı /java/lib64/jdbc dizininde saklanır.
- Oracle ' ı bir sistemin herhangi bir yerinde kurabileceğiniz için, makefiles, Oracle ' in kurulu olduğu yeri bulmak için ORACLE\_HOME ortam değişkenini kullanır.

After you have created the switch libraries for Db2, Oracle, or both, you must declare them to your queue manager. Kuyruk yöneticisi yapılandırma dosyası (qm.ini), Db2 ya da Oracle veritabanlarına ilişkin XAResourceManager stanzaları içeriyorsa, her bir Stanza içindeki SwitchFile girişini aşağıdakilerden biri ile değiştirmeniz gerekir:

### Db2 veritabanı için

```
SwitchFile=jdbcd2
```

### Oracle veritabanı için

```
SwitchFile=jdbcora
```

32 bit ya da 64 bit anahtar kitaplığının tam olarak nitelenmiş yol adını belirtmeyin. Yalnızca kitaplığın adını belirtin.

If the queue manager configuration file does not already contain XAResourceManager stanzas for Db2 or Oracle databases, or if you want to add additional XAResourceManager stanzas, see [Administering](#) for information about how to construct an XAResourceManager stanza. Ancak, yeni bir XAResourceManager stanza içindeki her SwitchFile girişi, tam olarak Db2 ya da Oracle veritabanı için açıklandığı gibi olmalıdır. You must also include the entry ThreadOfControl=PROCESS.

Kuyruk yöneticisi yapılandırma kütüğünü güncelledikten ve tüm uygun veritabanı ortam değişkenlerinin ayarlandığından emin olduktan sonra, kuyruk yöneticisini yeniden başlatabilirsiniz.

## JTA/JDBC koordinasyonunun kullanılması

API çağrılarınızı sağlayan örnekte olduğu gibi kodlayın.

Bir kullanıcı uygulaması için API çağrılarının temel sırası şöyledir:

```
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 çağrısındaki xads , bağlanmak için veritabanının ayrıntılarını tanımlayan XADatSource arabiriminin veritabanına özgü bir somutlamasını içerir. getJDBCConnection' a geçmek üzere uygun bir XADatSource nesnesinin nasıl yaratılacağı hakkında bilgi almak için veritabanınıza ilişkin belgelere bakın.

You must also update your class path with the appropriate database-specific jar files for performing JDBC work.

Birden çok veritabanına bağlanmanız gerekiyorsa, birçok farklı bağlantıda işlemi gerçekleştirmek için getJDBCConnection çağrısını birkaç kez aramalısınız.

There are two forms of the getJDBCConnection, reflecting the two forms of XADatSource.getXConnection:

```
public java.sql.Connection getJDBCConnection(javax.sql.XADatSource xads)
    throws MQException, SQLException, Exception

public java.sql.Connection getJDBCConnection(XADatSource dataSource,
    String userid, String password)
    throws MQException, SQLException, Exception
```

Bu yöntemler, JTA işlevlerini kullanmayan müşteriler için JVM doğrulayıcısındaki sorunları önlemek için throws yantımlerinde kural dışı durum bildirmektedir. Atılan gerçek kural dışı durum javax.transaction.xa.XAException , daha önce gerektirmeyen programlar için sınıf yoluna jta.jar dosyasının eklenmesini gerektirir.

JTA/JDBC desteğini kullanmak için, uygulamanıza aşağıdaki deyiimi eklemelisiniz:

```
MQEnvironment.properties.put(CMQC.THREAD_AFFINITY_PROPERTY, new Boolean(true));
```

## JTA/JDBC eşgüdümü ile ilgili bilinen sorunlar ve sınırlamalar

JTA/JDBC desteğinin bazı sorunları ve sınırlamaları, kullanımda olan veritabanı yönetim sistemine bağlıdır; örneğin, test edilmiş JDBC sürücüleri, uygulama çalışırken veritabanı kapatıldığında farklı davranır. Bir uygulamanın kullandığı veritabanıyla bağlantı kesilirse, uygulamanın kuyruk yöneticisi ve veritabanı ile yeni bir bağlantı kurmak için gerçekleştirebileceği adımlar vardır; böylece, bu yeni bağlantıları kullanarak işlemsel işi gerçekleştirmek için bu yeni bağlantıları kullanabilirsiniz.

JTA/JDBC desteği, JDBC sürücülerini çağırdığı için, bu JDBC sürücülerinin somutlaması, sistem işleyişi üzerinde önemli bir etkiye sahip olabilir. Özellikle, test edilen JDBC sürücüleri, bir uygulama çalışırken veritabanı kapatıldığında farklı davranır.

**Önemli:** Bir veritabanını, açık bağlantıları tutan uygulamalar varken, her zaman bir veritabanını aniden kapatmaktan kaçınınız.

**Not:** Bir IBM MQ classes for Java uygulaması, veritabanı eşgüdümcüsü olarak IBM MQ işlemi yapmak için bağ tanımları kipini kullanarak bağlanmalıdır.

## Birden çok XAResourceManager stanzası

Bir kuyruk yöneticisi yapılanış kütüğünde ( qm . ini ) birden çok XAResourceManager stanza kullanımı desteklenmez. Birinciden başka herhangi bir XAResourceManager kısmı yok sayılır.

## Db2

Bazen Db2 , bir SQL0805N hatası döndürür. Bu sorun şu CLP komutlarıyla çözülebilir:

```
DB2 bind @db2cli.lst blocking all grant public
```

Ek bilgi için Db2 belgelerine bakın.

XAResourceManager stanza, ThreadOfControl=PROCESS kullanacak şekilde yapılandırılmalıdır. Db2 8.1 ve üstü için bu, Db2 için denetim ayarının varsayılan iş parçacığı ile eşleşmez; dolayısıyla, XA Open String içinde toc=p belirtilmelidir. JTA/JDBC eşgüdümü ile Db2 için bir XAResourceManager stanza örneği aşağıdaki gibidir:

```
XAResourceManager:  
Name=jdbcdb2  
SwitchFile=jdbcdb2  
XAOpenString=uid=userid,db=dbalias,pwd=password,toc=p  
ThreadOfControl=PROCESS
```

Bu, JTA/JDBC eşgüdümü kullanan Java uygulamalarının birden çok iş parçacıklı olmasını engellemektedir.

## Oracle

MQQueueManager.disconnect () işleminden sonra JDBC Connection.close() yöntemi bir SQL kural dışı durumu (SQL kural dışı durumu) yaratır. Either call Connection.close() before MQQueueManager.disconnect(), or omit the call to Connection.close().

## Veri tabanı bağlantılarıyla ilgili sorunları ele alma

Bir IBM MQ classes for Java uygulaması, IBM MQ tarafından sağlanan JTA/JDBC desteğini kullandığında, genellikle aşağıdaki adımları gerçekleştirir:

1. İşlem yöneticisi olarak işlev göreceği, kuyruk yöneticisine yönelik bir bağlantıyı göstermek için yeni bir MQQueueManager nesnesi yaratır.
2. İşlemde listeleneceği veritabanına nasıl bağlanacağına ilişkin ayrıntıları içeren bir XADatasource nesnesi oluşturur.
3. Calls the method MQQueueManager.getJDBCConnection(XADatasource) passing in the XADatasource that was created previously. 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 işini gerçekleştirir.
6. Gerekli tüm işler tamamlandıktan sonra, MQQueueManager.commit () yöntemini çağırır. Bu, 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 yöneticisinden bağlantıyı kesmek için MQQueueManager.disconnect () yöntemini çağırmanız gerekir.

IBM MQ classes for Java , bir uygulama MQQueueManager.getJDBCConnection(XADatasource) çağırılırken yaratılmış olan tüm veritabanı bağlantılarının bir iç listesini korur. Bir kuyruk yöneticisinin XA hareketinin işlenmesi sırasında veritabanıyla iletişim kurması gerekiyorsa, aşağıdaki işlem gerçekleşir:

1. Kuyruk yöneticisi, veritabanına geçirilmesi gereken XA çağrısının ayrıntılarına geçerek, IBM MQ classes for Java' e çağrıda bulunmalarını sağlar.
2. IBM MQ classes for Java daha sonra listede uygun bağlantıyı arar ve daha sonra, XA çağrısını veritabanına akış için bu bağlantıyı kullanır.

Veritabanı bağlantısı, bu işlem sırasında herhangi bir noktada kaybolursa, uygulama şöyle olmalıdır:

1. MQQueueManager.backout () yöntemini çağırarak, hareket altında yapılan var olan tüm işleri yedekle.

2. Veritabanı bağlantısını kapatın. Bunun nedeni, IBM MQ classes for Java ' un bozuk veritabanı bağlantısının ayrıntılarını iç listesinde kaldırmasına neden olmalıdır.
3. Disconnect from the queue manager, by calling the method MQQueueManager.disconnect().
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 yöneticisine ve veritabanına yeni bir bağlantı yeniden kurmasını ve daha sonra bu bağlantıları kullanarak işlemsel çalışmayı gerçekleştirmek için kullanılmasını sağlar.

### ***Transport Layer Security (TLS) support in IBM MQ classes for Java***

IBM MQ classes for Java istemci uygulamaları TLS şifrelemesini destekler. TLS şifrelemesini kullanmak için bir JSSE sağlayıcısına gereksinim duyarsınız.

TRANSPORT (CLIENT) kullanan IBM MQ classes for Java istemci uygulamaları TLS şifrelemesini destekler. TLS, iletişim şifreleme, kimlik doğrulama ve ileti bütünlüğü sağlar. Bu, genellikle İnternet üzerinde ya da bir intranet içinde herhangi iki eş arasında iletişim sağlamak için kullanılır.

IBM MQ classes for Java , TLS şifrelemesini işlemek için Java Secure Socket Extension (JSSE) olanağını kullanır ve JSSE sağlayıcısını gerektirir. JSE v1.4 JVM ' ler yerleşik bir JSSE sağlayıcısına sahip. Sertifikaların nasıl yönetileceği ve saklanabileceği ile ilgili ayrıntılar sağlayıcıdan sağlayıcıya göre değişebilir. Bu konuda bilgi almak için, JSSE sağlayıcısının belgelerine bakın.

Bu bölümde, JSSE sağlayıcınızın doğru bir şekilde kurulmuş ve yapılandırılmış olduğu ve JSSE sağlayıcınız için uygun sertifikaların kurulmuş ve kullanılabilir durumda olduğu varsayılır.

IBM MQ classes for Java istemci uygulamanız bir kuyruk yöneticisine bağlanmak için bir istemci kanal tanımlama çizelgesi (CCDT) kullanıyorsa, [“Using a client channel definition table with IBM MQ classes for Java” sayfa 349](#) başlıklı konuya bakın.

#### *Enabling TLS in IBM MQ classes for Java*

TLS ' yi etkinleştirmek için bir CipherSuite belirtmenizi sağlar. Bir CipherSuite belirtmenin iki yolu vardır.

TLS yalnızca istemci bağlantıları için desteklenir. TLS ' yi etkinleştirmek için, kuyruk yöneticisiyle iletişim kurarken kullanılacak CipherSuite değerini belirlemeniz gerekir; bu CipherSuite , hedef kanaldaki CipherSpec ayarına uygun olmalıdır. Buna ek olarak, JSSE sağlayıcınız tarafından desteklenen CipherSuite ' in de desteklenmesi gerekir. Ancak, CipherSuites , CipherSpecs ' dan farklıdır ve farklı adlara sahiptir. [“TLS CipherSpecs and CipherSuites in IBM MQ classes for Java” sayfa 380](#) contains a table mapping the CipherSpecs supported by IBM MQ to their equivalent CipherSuites as known to JSSE.

TLS 'yi etkinleştirmek için, MQEnvironment 'in sslCipher Suite statik üye değişkenini kullanarak CipherSuite ' i belirtin. Aşağıdaki örnek, TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256: ' un CipherSpec ile TLS gerektirecek şekilde ayarlanmış bir SVRCONN kanalına SECURE.SVRCONN.CHANNEL adlı bir kanala 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");
```

Kanalda TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 için bir CipherSpec sahip olmasına rağmen, Java uygulaması SSL\_RSA\_WITH\_AES\_128\_CBC\_SHA256 için bir CipherSuite belirtmelidir. See [“TLS CipherSpecs and CipherSuites in IBM MQ classes for Java” sayfa 380](#) for a list of mappings between CipherSpecs and CipherSuites.

Bir uygulama ayrıca CMQC.SSL\_CIPHER\_SUITE\_PROPERTY ortam özelliğini ayarlayarak bir CipherSuite de belirtebilir.

Diğer bir seçenek olarak, İstemci Kanal Tanımlama Çizelgesi 'ni (CCDT) kullanın. Daha fazla bilgi için bkz. [“Using a client channel definition table with IBM MQ classes for Java” sayfa 349](#)



IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) tarafından desteklenen bir CipherSuite kullanmak için istemci bağlantısı gerekiyorsa, bir uygulama MQEnvironment sınıfındaki sslFipsRequired alanını true olarak ayarlayabilir. Diğer bir seçenek olarak, uygulama CMQC.SSL\_FIPS\_REQUIRED\_PROPERTY ortam özelliğini ayarlayabilir. Varsayılan değer false'dir. Bu, bir istemci bağlantısının IBM MQ tarafından desteklenen herhangi bir CipherSuite 'i kullanabileceği anlamına gelir.

Bir uygulama birden çok istemci bağlantısı kullanıyorsa, uygulama ilk istemci bağlantısını yarattığında kullanılan sslFipsGerekli alanının değeri, uygulama sonraki istemci bağlantısı yarattığında kullanılan değeri belirler. Bu nedenle, uygulama sonraki bir istemci bağlantısı yarattığında, sslFipsRequired (Gerekli) alanının değeri yoksayılır. sslFipsZorunlu alanı için farklı bir değer kullanmak istiyorsanız uygulamayı yeniden başlatmalısınız.

TLS 'yi başarıyla kullanarak bağlanmak için, JSSE güvenilirlik deposunun, kuyruk yöneticisi tarafından sunulan sertifikana doğrulanabilmesi için sertifika yetkilisi kök sertifikalarıyla ayarlanması gerekir. Benzer şekilde, SVRCONN kanalında SSLClientAuth MQSSL\_CLIENT\_AUTH\_REQUIRED olarak ayarlandıysa, JSSE anahtar deposu, kuyruk yöneticisi tarafından güvenilen bir tanıtıcı sertifika içermelidir.

## İlgili başvurular

[Federal Information Processing Standards \(FIPS\) for UNIX, Linux, and Windows](#)

*Using the distinguished name of the queue manager in IBM MQ classes for Java*

Kuyruk yöneticisi kendisini ayırt edici bir ad (DN) içeren bir TLS sertifikasını kullanarak tanımlar. Bir IBM MQ classes for Java istemci uygulaması, doğru kuyruk yöneticisiyle iletişim kurduğundan emin olmak için bu DN 'yi kullanabilir.

MQEnvironment 'in sslPeerAd değişkeni kullanılarak bir DN örneği belirtildi. Örneğin:

```
MQEnvironment.sslPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPHERE";
```

Ancak kuyruk yöneticisi, QMGR. ' un başında bir Common Name değeri olan bir sertifika sunarsa, bağlantının başarılı olmasına izin verir. ve en az iki Kuruluş Birimi adı, bunlardan ilki IBM ve ikinci WebSphere olmalıdır.

sslPeerAdı ayarlandıysa, bağlantılar yalnızca geçerli bir örüntü olarak ayarlandıysa ve kuyruk yöneticisi eşleşen bir sertifika sunarsa başarılı olur.

Bir uygulama ayrıca CMQC.SSL\_PEER\_NAME\_PROPERTY ortam özelliğini ayarlayarak, kuyruk yöneticisinin ayırt edici adını da belirtebilir. Ayırt edici adlara ilişkin ek bilgi için [Ayırt edici adlar](#) başlıklı konuya bakın.

*Using certificate revocation lists in IBM MQ classes for Java*

java.security.cert.CertStore sınıfı aracılığıyla kullanılacak sertifika iptal listelerini belirtin. IBM MQ classes for Java daha sonra, sertifikaları belirtilen CRL 'ye göre denetler.

Sertifika iptal listesi (CRL), sertifika yetkilisi ya da yerel kuruluş tarafından iptal edilen bir sertifikalar kümesidir. CRL 'ler genellikle LDAP sunucularında barındırılır. Java 2 v1.4 ile, bağlantı sırasında bir CRL sunucusu belirtilebilir ve kuyruk yöneticisi tarafından sunulan sertifika, bağlantıya izin verilmeden önce CRL 'ye yönelik olarak denetlenir. Sertifika iptal listeleri ve IBM MQ ile ilgili daha fazla bilgi için bkz. [Sertifika İptal Listeleri ve Yetki İptali Listeleriyle Çalışma ve IBM MQ classes for Java ve IBM MQ classes for JMS ile CRL 'lere ve ARL 'lere erişilmesi](#).

**Not:** Bir LDAP sunucusunda barındırılan bir CRL ile başarılı bir şekilde CertStore kullanmak için, Java Software Development Kit (SDK) ürününüzün CRL ile uyumlu olduğundan emin olun. Bazı SDK 'lar, CRL 'nin, LDAP v2 için bir şema tanımlayan RFC 2587 'ye uymasını gerektirir. Çoğu LDAP v3 sunucusu, bunun yerine RFC 2256 'yı kullanır.

Kullanılacak CRL 'ler java.security.cert.CertStore sınıfı aracılığıyla belirtilir. CertStore yönetim ortamlarının nasıl edinileceği ile ilgili ayrıntılı bilgi için bu sınıftaki belgelere bakın. LDAP sunucusuna dayalı olarak bir CertStore yaratmak için, önce bir LDAPCertStoreDeğiştiricileri eşgörünümü yaratın; kullanılacak sunucu ve kapı ayarlarıyla kullanıma hazırlandı. Örneğin:

```
import java.security.cert.*;
CertStoreParameters csp = new LDAPCertStoreParameters("crl_server", 389);
```

Bir CertStoreDeğiştiricileri eşgörünümü yarattıktan sonra, LDAP tipinde bir CertStore yaratmak için CertStore ' da durağan oluşturucuyu kullanın:

```
CertStore cs = CertStore.getInstance("LDAP", csp);
```

Diğer CertStore tipleri (örneğin, Derlem) de desteklenir. Genellikle, yedeklilik sağlamak için aynı CRL bilgileriyle birlikte ayarlanan birden çok CRL sunucusu vardır. Bu CRL sunucularının her biri için bir CertStore nesnesi bulunduğu, bunları uygun bir Toplamaya yerleştirin. Aşağıdaki örnekte, ArrayList(ArrayList) içine yerleştirilen CertStore nesnelere gösterilmektedir:

```
import java.util.ArrayList;
Collection crls = new ArrayList();
crls.add(cs);
```

CRL denetimini etkinleştirmek için bağlanmadan önce, bu toplama MQEnvironment statik değişkenine, sslCertStores 'a ayarlanabilir:

```
MQEnvironment.sslCertStores = crls;
```

Bir bağlantı ayarlandığında, kuyruk yöneticisi tarafından sunulan sertifika aşağıdaki gibi doğrulanır:

1. Kaynak grubundaki ilk CertStore nesnesi, bir CRL sunucusunu tanımlamak için kullanılır. sslCertdepolat tarafından tanımlanır.
2. CRL sunucusuyla bağlantı kurma girişiminde bulunmanız gerekir.
3. Girişim başarılı olursa, sunucu, sertifikana için bir eşleşme için arama yapılır.
  - a. Sertifikana geri alınacak bir sertifika bulunursa, arama işlemi sona erer ve bağlantı isteği, MQRC\_SSL\_CERTIFICATE\_FESHIVE neden kodlarıyla başarısız olur.
  - b. Sertifika bulunamazsa, arama işlemi sona ermiş ve bağlantının devam etmesine izin verilir.
4. Sunucuyla iletişim kurma girişimi başarısız olursa, bir sonraki CertStore nesnesi bir CRL sunucusunu tanımlamak için kullanılır ve süreç 2. adımdan yinelenir.

Bu, derlemdeki son CertStore ise ya da Kaynak Grubu hiçbir CertStore nesnesi içermiyorsa, arama işlemi başarısız oldu ve bağlantı isteği, MQRC\_SSL\_CERT\_STORE\_ERROR neden koduyla başarısız olur.

Veri Toplama nesnesi, CertStores ' un kullanıldığı sırayı belirler.

CertStores adlı kaynak grubu da CMQC.SSL\_CERT\_STORE\_PROPERTYkullanılarak da ayarlanabilir. Kolaylık olması nedeniyle, bu özellik, tek bir CertStore ' un bir Kaynak Grubu üyesi olmadan belirtilmesine de olanak tanır.

sslCertStores boş değere ayarlıysa, CRL denetimi gerçekleştirilmez. sslCipherSuite ayarlanmadıysa bu özellik yok sayılır.

#### *IBM MQ classes for Java içindeki gizli anahtarları yeniden ilişki kurma*

Bir IBM MQ classes for Java istemci uygulaması, bir istemci bağlantısında şifreleme için kullanılan gizli anahtarın, gönderilen ve alınan toplam bayt sayısı bakımından yeniden ilişki kuracağını denetleyebilir.

Uygulama bunu aşağıdaki yollardan biriyle yapabilir: Uygulama bu yollardan birinden fazlasını kullanıyorsa, olağan öncelik kuralları geçerli olur.

- MQEnvironment sınıfındaki sslResetSayı alanını ayarlayarak.
- Bir Hashtable nesnesinde MQC.SSL\_RESET\_COUNT\_PROPERTY ortam özelliği ayarlanarak. Uygulama daha sonra, hashtable ' ı MQEnvironment sınıfındaki properties alanına atar ya da hashtable ' ı oluşturucudaki bir MQQueueManager nesnesine geçirir.

sslResetSayı alanı ya da ortam özelliğinin değeri MQC.SSL\_RESET\_COUNT\_PROPERTY , 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 çözme işleminden sonra gelen sayıdır. Bayt sayısı, IBM MQ classes for Java istemcisi tarafından gönderilen ve alınan denetim bilgilerini de içerir.

İlk duruma getirme sayısı sıfırsa, varsayılan değer olan gizli anahtar hiçbir zaman yeniden anlaşılacaktır. CipherSuite belirtilmediyse, ilk duruma getirme sayısı dikkate alınmaz.

#### *IBM MQ classes for Java' ta özelleştirilmiş bir SSLSocketFactory sağlama*

Özelleştirilmiş bir JSSE Yuva Üreticisi kullanıyorsanız, MQEnvironment.sslSocketFactory ' yi özelleştirilmiş fabrika nesnesine ayarlayın. Ayrıntılar, farklı JSSE somutlamaları arasında farklılık gösterir.

Farklı JSSE uygulamaları farklı özellikler sağlayabilir. Örneğin, özelleştirilmiş bir JSSE somutlaması, belirli bir şifreleme donanımı modelinin yapılandırılmasına izin verebilir. Buna ek olarak, bazı JSSE sağlayıcıları, anahtar depolarının ve güvenilirlik depolarının programa göre özelleştirilmesine ya da anahtar deposundan değiştirilmek üzere kimlik sertifikası seçmesine izin verir. JSSE ' de, tüm bu ayarlamalar bir üretici sınıfına ( javax.net.ssl.SSLSocketFactory) soyutlanır.

Özelleştirilmiş bir SSLSocketFactory uygulamasının nasıl oluşturulabilmesiyle ilgili ayrıntılar için JSSE belgelerinize bakın. Ayrıntılar sağlayıcıdan sağlayıcıya göre değişir, ancak aşağıdaki tipik adımlar dizisi aşağıdaki gibi olabilir:

1. SSLContext üzerinde durağan bir yöntem kullanarak bir SSLContext nesnesi yaratır
2. Uygun KeyManager ve TrustManager uygulamalarıyla (kendi fabrika sınıflarından yaratılan) bu SSLContext ' i başlatın.
3. SSLContext içinden bir SSLSocketFactory yarat

When you have an SSLSocketFactory object, set the MQEnvironment.sslSocketFactory to the customized factory object. Örneğin:

```
javax.net.ssl.SSLSocketFactory sf = sslContext.getSocketFactory();
MQEnvironment.sslSocketFactory = sf;
```

IBM MQ classes for Java , IBM MQ kuyruk yöneticisine bağlanmak için bu SSLSocketFactory ' yi kullanın. Bu özellik, CMQC.SSL\_SOCKET\_FACTORY\_PROPERTYkullanılarak da ayarlanabilmektedir. sslSocketFactory boş değer olarak ayarlandıysa, JVM ' nin varsayılan SSLSocketFactory değeri kullanılır. sslCipherSuite ayarlanmadıysa bu özellik yok sayılır.

Özel SSLSocketFactories' i kullandığınızda, TCP/IP bağlantı paylaşımının etkisini göz önünde bulundurun. If connection sharing is possible then a new socket is not requested of the SSLSocketFactory supplied, even if the socket produced would be different in some way in the context of a subsequent connection request. Örneğin, sonraki bir bağlantıda farklı bir istemci sertifikası sunulacaksa, bağlantı paylaşımına izin verilmemesi gerekir.

*IBM MQ classes for Java' da JSSE anahtar deposunda ya da güvenilirlik deposunda değişiklik yapılması* JSSE anahtar deposunu ya da güvenilir deposunu değiştirdiğinizde, değişikliklerin yürürlüğe girmesi için bazı işlemleri gerçekleştirmeniz gerekir.

JSSE anahtar deposu ya da güvenilirlik deposunun içeriğini değiştirirseniz ya da anahtar deposu ya da güvenilirlik deposu dosyasının konumunu değiştirdiğinizde, o sırada çalışan IBM MQ classes for Java uygulamaları otomatik olarak değişiklikleri almaz. Değişikliklerin yürürlüğe girmesi için aşağıdaki işlemler gerçekleştirilmelidir:

- Uygulamalar tüm bağlantılarını kapatmalı ve bağlantı havuzlarındaki kullanılmayan bağlantıları yok etmelidir.
- JSSE sağlayıcınız, bilgileri anahtar deposundan ve güvenilir depodan önbelleğe aldıysa, bu bilgiler yenilenmelidir.

Bu işlemler gerçekleştirildikten sonra, uygulamalar bağlantılarını yeniden yaratabilirler.

Uygulamalarınızı nasıl tasarladığınıza ve JSSE sağlayıcınız tarafından sağlanan işlemlere bağlı olarak, uygulamalarınızı durdurup yeniden başlatmaksızın bu işlemleri gerçekleştirmeniz de mümkün olabilir. Ancak, uygulamaların durdurulması ve yeniden başlatılması en basit çözüm olabilir.

#### *Error handling when using TLS with IBM MQ classes for Java*

Bir kuyruk yöneticisine TLS kullanılarak bağlanılırken IBM MQ classes for Java tarafından bir dizi neden kodu yayınlanabilir.

Bu bilgiler aşağıdaki listede açıklanmıştır:

#### **MQRC\_SSL\_NOT\_ALLOWED**

sslCipherSuite özelliği ayarlıydı, ancak bağ tanımları bağlantısı kullanıldı. Yalnızca istemci bağlantısı TLS 'yi destekler.

#### **MQRC\_JSSE\_ERROR**

JSSE sağlayıcısı, IBM MQ tarafından işlenemeyen bir hata bildirdi. Bunun nedeni, JSSE ile bir yapılandırma sorunu olabilir ya da kuyruk yöneticisi tarafından sunulan sertifikana doğrulanamadı. JSSE tarafından üretilen kural dışı durum, MQException 'daki getCause() yöntemi kullanılarak alınabilir.

#### **MQRC\_SSL\_INITIALIZATION\_ERROR**

TLS yapılandırma seçenekleri belirtilen 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**

sslPeerAd özelliğinde belirtilen DN kalıbı geçerli değil.

#### **MQRC\_UNSUPPORTED\_CIPHER\_SUITE**

sslCipher Suite içindeki CipherSuite takımı JSSE sağlayıcısı tarafından tanınmadı. A full list of CipherSuites supported by the JSSE provider can be obtained by a program using the SSLSocketFactory.getSupportedCipherSuites() method. IBM MQ ile iletişim kurmak için kullanılacak CipherSuites listesi, [“TLS CipherSpecs and CipherSuites in IBM MQ classes for Java”](#) sayfa 380 içinde bulunabilir.

#### **MQRC\_SSL\_CERTIFICATE\_FRESHED**

Kuyruk yöneticisi tarafından sunulan sertifika, sslCertStores özelliği ile belirtilen bir CRL ' de bulundu. Güvenilir sertifikalar kullanmak için kuyruk yöneticisini güncelleyin.

#### **MQRC\_SSL\_CERT\_STORE\_ERROR**

Kuyruk yöneticisi tarafından sunulan sertifika için, belirtilen CertStores ' in hiçbiri aranmadı. MQException.getCause() yöntemi, denenen ilk CertStore aranırken oluşan hatayı döndürür. Nedensel kural dışı durum NoSuchElementException, ClassCastKural Dışı Durumu ya da NullPointerException Durumu ise, sslCertMağazalarında belirtilen Derlemin en az bir geçerli CertStore nesnesi içerdiğini doğrulayın.

#### *TLS CipherSpecs and CipherSuites in IBM MQ classes for Java*

IBM MQ classes for Java uygulamalarının bir kuyruk yöneticisiyle bağlantı kurabilme yeteneği, MQI kanalının sunucu ucunda belirtilen CipherSpec değerine ve istemci ucunda belirtilen CipherSuite ' e bağlıdır.

Aşağıdaki çizelgede IBM MQ tarafından desteklenen CipherSpecs ve eşdeğeri olan CipherSuites listesi listelenmektedir.

Aşağıdaki çizelgede listelenen CipherSpecs'in IBM MQ tarafından kullanımdan kaldırılmış olup olmadığını ve bu durumda CipherSpec ' in kullanımdan kaldırılıp kaldırılmadığını görmek için [Kullanımdan Kaldırılan CipherSpecs](#) konusunu gözden geçirmelisiniz.

**Önemli:** Listelenen CipherSuites , IBM MQ ile sağlanan IBM Java Runtime Environment (JRE) tarafından desteklenen sürümlerdir. Listelenen CipherSuites , Oracle Java JRE tarafından desteklenenler içerir. Uygulamanızın bir Oracle Java JRE kullanacak şekilde yapılandırılmasına ilişkin ek bilgi edinmek için bkz. [“Uygulamanızın IBM Java ya da Oracle Java CipherSuite eşlemelerini kullanacak şekilde yapılandırılması”](#) sayfa 409.

Tablo ayrıca, iletişim için kullanılan protokolü ve CipherSuite ' in FIPS 140-2 standardına uygun olup olmadığını da belirtir.

Uygulama, FIPS 140-2 uyumluluğunu zorunlu kılacak şekilde yapılandırılmamışsa, FIPS 140-2 uyumluluğu yapılandırılmamışsa, ancak uygulama için FIPS 140-2 uyumluluğu yapılandırıldıysa (yapılandırma için aşağıdaki notlara bakın), yalnızca FIPS 140-2 uyumlu olarak işaretlenmiş olan CipherSuites ' lar yapılandırılabilir; başka CipherSuites sonuçlarını bir hatayla kullanma girişiminde bulunmanız gerekir.

**Not:** Her JRE birden çok şifreleme güvenlik sağlayıcısına sahip olabilir; bunların her biri aynı CipherSuiteürünün uygulanmasına katkıda bulunabilir. Ancak, tüm güvenlik sağlayıcıları FIPS 140-2 sertifikalı değildir. If FIPS 140-2 compliance is not enforced for an application then it is possible that an uncertified implementation of the CipherSuite might be used. Onaylanmamış uygulamalar FIPS 140-2 ile uyumlu çalışmayabilir; CipherSuite teorik olarak, standart için gerekli olan en düşük güvenlik düzeyini karşılayabilir. See the following notes for more information about configuring FIPS 140-2 enforcement in IBM MQ Java applications.

CipherSpecs ve CipherSuitesi için FIPS 140-2 ve Suite-B uyumluluğu ile ilgili daha fazla bilgi için bkz. [Specifying CipherSpecs](#). Ayrıca, [US Federal Information Processing Standards](#)(Federal Bilgi İşleme Standartları) ile ilgili bilgilerden haberdar olmanız da gerekebilir.

To use the full set of CipherSuites and to operate with certified FIPS 140-2 and/or Suite-B compliance, a suitable JRE is required. IBM Java 7 Service Refresh 4 Düzeltme Paketi 2 ya da daha yüksek bir IBM JRE düzeyi, [Çizelge 58 sayfa 382](#) içinde listelenen TLS 1.2 CipherSuites için uygun desteği sağlar.

**V 9.1.5** TLS v1.3 Ciphers olanağını kullanabilmek için uygulamanızın çalışan JRE 'nin TLS v1.3' ü desteklemesi gerekir.

**Not:** Bazı CipherSuites'i kullanmak için, JRE' de 'kısıtlanmamış' ilke dosyalarının yapılandırılması gerekir. İlke dosyalarının bir SDK ya da JRE ' de nasıl ayarlarıyla ilgili ayrıntılı bilgi için, kullandığınız sürüm için [Security Reference for IBM SDK, Java Technology Edition](#) içindeki [IBM SDK Policy files](#) başlıklı konuya bakın.

Çizelge 58. IBM MQ ve eşdeğer CIPHER Suite tarafından desteklenen Cipher Specs

CipherSpec	Eşdeğer Cipher Suite (IBM JRE)	Eşdeğer Cipher Suite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 58. IBM MQ ve eşdeğer CIPHER Suite tarafından desteklenen Cipher Specs (devamı var)

CipherSpec	Eşdeğer Cipher Suite (IBM JRE)	Eşdeğer Cipher Suite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_ECDSA_NULL_SHA256	SSL_ECDHE_ECDSA_WITH_NULL_SHA	TLS_ECDHE_ECDSA_WITH_NULL_SHA	TLS 1.2	hayır

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CIPHERSPEC	Eşdeğer CIPHERSUITE (IBM JRE)	Eşdeğer CIPHERSUITE (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_ECDSA_RC4_128_SHA256	SSL_ECDHE_ECDSA_WITH_RC4_128_SHA	TLSECDHE_ECDSA_WITH_RC4_128_SHA	TLS 1.2	hayır

Çizelge 58. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
ECDHE_RSA_3DES_EDE_CBC_SHA 256	SSL_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA	TLS 1.2	evet

Çizelge 58. IBM MQ ve eşdeğer CIPHER Suite tarafından desteklenen CIPHER Specs (devamı var)

CIPHER Spec	Eşdeğer CIPHER Suite (IBM JRE)	Eşdeğer CIPHER Suite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 58. IBM MQ ve eşdeğer CIPHER Suite tarafından desteklenen Cipher Specs (devamı var)

CipherSpec	Eşdeğer Cipher Suite (IBM JRE)	Eşdeğer Cipher Suite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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

Çizelge 58. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_RSA_NULL_SHA256	SSL_ECDHE_RSA_WITH_NULL_SHA	TLS_ECDHE_RSA_WITH_NULL_SHA	TLS 1.2	hayır

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
ECDHE_RSA_RC4_128_SHA256	SSL_ECDHE_RSA_WITH_RC4_128_SHA	TLSECDHE_RSA_WITH_RC4_128_SHA	TLS 1.2	hayır

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CIPHERSPEC	Eşdeğer CIPHERSUITE (IBM JRE)	Eşdeğer CIPHERSUITE (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_3DES_EDE_CBC_SHA <a href="#">"1" sayfa 408</a>	SSL_RSA_WITH_3DES_EDE_CBC_SHA	TL S_ R S A - W I T H _ 3 D E S _ E D E _ C B C _ S H A	TLS 1.0	evet

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_128_CBC_SHA	SSL_RSA_WITH_AES_128_CBC_SHA	TL S_ R S A - W I T H - A E S _1 2 8_ C B C_ S H A	TLS 1.0	evet

Çizelge 58. IBM MQ ve eşdeğer CIPHER Suite tarafından desteklenen Cipher Specs (devamı var)

CipherSpec	Eşdeğer Cipher Suite (IBM JRE)	Eşdeğer Cipher Suite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_128_CBC_SHA256	SSL_RSA_WITH_AES_128_CBC_SHA256	TL S_ R S A - W I T H - A E S _1 2 8_ C B C_ S H A 2 5 6	TLS 1.2	evet

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	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	TL S_ R S A - W I T H - A E S _1 2 8_ G C M _S H A 2 5 6	TLS 1.2	evet

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_256_CBC_SHA	SSL_RSA_WITH_AES_256_CBC_SHA	TLS_RSA_WITH_AES_256_CBC_SHA	TLS 1.0	evet



Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	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	TL S_ R S A - W I T H - A E S _2 5 6_ C B C_ S H A 2 5 6	TLS 1.2	evet

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
TLS_RSA_WITH_AES_256_GCM_SHA384	SSL_RSA_WITH_AES_256_GCM_SHA384	TL S_ R S A - W I T H - A E S _2 5 6_ G C M _S H A 3 8 4	TLS 1.2	evet

Çizelge 58. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
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

Çizelge 58. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
TLS_RSA_WITH_RC4_128_SHA256	SSL_RSA_WITH_RC4_128_SHA	SSL_RSA_WITH_RC4_128_SHA	TLS 1.2	hayır
ANY_TLS12	*TLS12	*TLS12	TLS 1.2	evet

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
<b>V9.1.5</b> TLS_AES_128_GCM_SHA256 "2" sayfa 408	TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256	TLS V1.3	hayır
<b>V9.1.5</b> TLS_AES_256_GCM_SHA384 "2" sayfa 408	TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384	TLS V1.3	hayır

Çizelge 58. IBM MQ ve eşdeğer CIPHERSUITE tarafından desteklenen CIPHERSPECS (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumu
<p>► <b>V9.1.5</b></p> <p>TLS_CHACHA20_POLY1305_SHA256 6 "2" sayfa 408</p>	TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256	TLS V1.3	hayır

Çizelge 58. IBM MQ ve eşdeğer CipherSuite tarafından desteklenen CipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
<p>► <b>V9.1.5</b>                      TLS_AES_128_CCM_SHA256                      "2" sayfa 408</p>	<p>TLS_AES_128_CCM_SHA256</p>	<p>TL S_ A E S _1 2 8_ C C M _S H A 2 5 6</p>	<p>TLS V1.3</p>	<p>hayır</p>
<p>► <b>V9.1.5</b>                      TLS_AES_128_CCM_8_SHA256                      "2" sayfa 408</p>	<p>TLS_AES_128_CCM_8_SHA256</p>	<p>TL S_ A E S _1 2 8_ C C M _8 _S H A 2 5 6</p>	<p>TLS V1.3</p>	<p>hayır</p>
<p>► <b>V9.1.5</b> ANY "2" sayfa 408</p>	<p>*ANA</p>	<p>*A N A</p>	<p>Çoklu</p>	<p>hayır</p>

Çizelge 58. IBM MQ ve eşdeğer CipherSuitelarafından desteklenenCipherSpecs (devamı var)

CipherSpec	Eşdeğer CipherSuite (IBM JRE)	Eşdeğer CipherSuite (Oracle JRE)	Protokol	FIPS 140-2 uyumlu
> V9.1.5 ANY_TLS13 "2" sayfa 408	*TLS13	*TLS13	TLS V13	hayır
> V9.1.5 ANY_TLS12_OR_HIGHER "2" sayfa 408	*TLS12ORHIGHER	*TLS12ORHIGHER	TLS V1.2 ve üstü	hayır
> V9.1.5 ANY_TLS13_OR_HIGHER "2" sayfa 408	*TLS13ORHIGHER	*TLS13ORHIGHER	TLS V1.3 ve üstü	hayır

**Notlar:**

1. CipherSpec TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA kullanımdan kaldırıldı. However, it can still be used to transfer up to 32 GB of data before the connection is terminated with error AMQ9288. Bu hatayı önlemek için, üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpec komutunu kullanırken gizli anahtar sıfırlamayı etkinleştirmeniz gerekir.
2. > V9.1.5 TLS v1.3 Ciphers olanağını kullanabilmek için uygulamanızın çalışan JRE 'nin TLS v1.3' ü desteklemesi gerekir.



## IBM MQ classes for Java uygulamasında Ciphersuites ve FIPS uyumluluğunu yapılandırma

- IBM MQ classes for Java 'u kullanan bir uygulama, bir bağlantı için CipherSuite ' i ayarlamak için iki yöntemden birini kullanabilir:
  - MQEnvironment sınıfındaki sslCipherSuite alanını CipherSuite adına ayarlayın.
  - Set the property CMQC.SSL\_CIPHER\_SUITE\_PROPERTY in the properties hashtable passed to the MQQueueManager constructor to the CipherSuite name.
- IBM MQ classes for Java ' u kullanan bir uygulama, FIPS 140-2 uyumluluğunu zorunlu kılacak iki yöntemden birini kullanabilir:
  - MQEnvironment sınıfında sslFipsRequired alanını true değerine ayarlayın.
  - Set the property CMQC.SSL\_FIPS\_REQUIRED\_PROPERTY in the properties hash table passed to the MQQueueManager constructor to true.

## Uygulamanızın IBM Java ya da Oracle Java CipherSuite eşlemelerini kullanacak şekilde yapılandırılması

Uygulamanızın varsayılan IBM Java CipherSuite ögesini IBM MQ CipherSpec eşlemelerine ya da Oracle CipherSuite ' i IBM MQ CipherSpec eşlemelerine kullanıp kullanmayacağını yapılandırabilirsiniz. Bu nedenle, uygulamanızın bir IBM JRE ya da bir Oracle JRE kullanıyorsa TLS CipherSuites ' i kullanabilirsiniz. The Java System Property `com.ibm.mq.cfg.useIBMCipherMappings` controls which mappings are used. Özellik, aşağıdaki değerlerden biri olabilir:

### doğru

IBM Java CipherSuite ' i IBM MQ CipherSpec eşlemelerine kullanın.

Bu değer, varsayılan değerdir.

### yanlış

Oracle CipherSuite ' i IBM MQ CipherSpec eşlemelerine kullanın.

For more information about using IBM MQ Java and TLS Ciphers, see the MQdev blog post [MQ Java, TLS Ciphers, Non-IBM JREs & APAR 'lar IT06775, IV66840, IT09423, IT10837.](#)

## Birlikte çalışabilirlik

Belirli CipherSuites , kullanılan protokole bağlı olarak birden çok IBM MQ CipherSpec ile uyumlu olabilir. Ancak , yalnızca Tablo 1 'de belirtilen TLS sürümünü kullanan CipherSuite/CipherSpec bileşimi desteklenir. Desteklenmeyen CipherSuites ve CipherSpecs birleşimlerini kullanmaya çalışmak uygun bir kural dışı durum ile başarısız olur. Bu CipherSuite/CipherSpec birleşimlerinden herhangi birini kullanan kuruluşlar, desteklenen bir birleşmeye geçmelidir.

Aşağıdaki tabloda, bu sınırlamanın geçerli olduğu CipherSuites gösterilmektedir.

CipherSuite	Desteklenen TLS CipherSpec	Desteklenmeyen SSL CipherSpec
SSL_RSA_WITH_3DES_EDE_CBC_SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA A "1" sayfa 409	TRIPLE_DES_SHA_US
SSL_RSA_WITH_DES_CBC_SHA	TLS_RSA_WITH_DES_CBC_SHA	DESTE_SHA_EXPORT
SSL_RSA_WITH_RC4_128_SHA	TLS_RSA_WITH_RC4_128_SHA256	RC4_SHA_US

### Not:

1. Bu CipherSpec TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA kullanımdan kaldırılmıştır. However, it can still be used to transfer up to 32 GB of data before the connection is terminated with error AMQ9288. Bu

hatayı önlemek için, üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckomutunu kullanırken gizli anahtar sıfırlamayı etkinleştirmeniz gerekir.

### **IBM MQ classes for Java uygulamalarının çalıştırılması**

If you write an application (a class that contains a main() method), using either the client or the bindings mode, run your program using the Java interpreter.

Şu komutu kullanın:

```
java -Djava.library.path= library_path MyClass
```

Burada *library\_path* , IBM MQ classes for Java kitaplıklarının yoludur. Daha fazla bilgi için [“IBM MQ classes for Java Kitaplıklar” sayfa 331](#) başlıklı konuya bakın.

### **İlgili görevler**

[IBM MQ classes for Java uygulamalarının izlenmesi](#)

[IBM MQ Kaynak Bağdaştırıcısı 'nın İzlenmesi](#)

### **IBM MQ classes for Java ortamına bağımlı davranış**

IBM MQ classes for Java , farklı IBM MQ sürümlerine karşı çalışabilecek uygulamalar oluşturmanızı sağlar. Bu konu grubunda, Java sınıflarının bu farklı sürümlere bağımlı davranışı anlatılır.

IBM MQ classes for Java , tüm ortamlarda tutarlı bir işlev ve davranış sağlayan, sınıfların bir temel sınıfını sağlar. Bu çekirdeğin dışındaki özellikler, uygulamanın bağlı olduğu kuyruk yöneticisinin yeteneğine bağlıdır.

Burada belirtilenler dışında, sergilenen davranış, kuyruk yöneticisine uygun [MQI uygulama başvurusu](#) içinde anlatıldığı gibidir.

### **IBM MQ classes for Java içindeki temel sınıflar**

IBM MQ classes for Java , tüm ortamlarda kullanılabilen bir temel sınıf kümesi içerir.

Aşağıdaki sınıf kümesi, temel sınıflar olarak kabul edilir ve tüm ortamlarda yalnızca [“Restrictions and variations for core classes of IBM MQ classes for Java” sayfa 411](#) içinde listelenen küçük çeşitlilikler ile kullanılabilir.

- MQEnvironment
- MQException
- MQGetMessageSeçenekleri

Hariç Tutma:

- MatchOptions
- GroupStatus
- SegmentStatus
- Bölümleme

- MQManagedObject

Hariç Tutma:

- sorgulama ()
- set ()

- MQMessage

Hariç Tutma:

- groupId
- messageFlags

- messageSequenceNumarası
- offset
- originalLength
- MQPoolServices
- MQPoolServicesOlayı
- MQPoolServicesEventListener
- MQPoolToken
- MQPutMessageSeçenekleri

Hariç Tutma:

- knownDestSayısı
- unknownDestSayısı
- invalidDestSayı
- recordFields
- MQProcess
- MQQueue
- MQQueueManager

Hariç Tutma:

- begin ()
- accessDistributionListesi ()
- MQSimpleConnectionYöneticisi
- MQTopic
- MQC

**Not:**

1. Bazı sabitler çekirdeklere dahil değildir (ayrıntılar için bkz. [“Restrictions and variations for core classes of IBM MQ classes for Java” sayfa 411](#)); bunları tamamen taşınabilir programlarda kullanmayın.
2. Bazı platformlar tüm bağlantı kiplerini desteklemez. Bu altyapılarda, yalnızca desteklenen kiplerle ilgili temel sınıfları ve seçenekleri kullanabilirsiniz.

***Restrictions and variations for core classes of IBM MQ classes for Java***

Temel sınıflar genellikle tüm ortamlarda tutarlı bir şekilde davranır, aynı MQI çağrılarını normal olarak ortam farklılıklarına sahip olsa bile. The behavior is as if a Windows, UNIX or Linux IBM MQ queue manager is used, except for the following minor restrictions and variations.

*IBM MQ classes for Java* içindeki MQGMO\_\* değerlerine ilişkin kısıtlamalar

Bazı MQGMO\_\* değerleri, tüm kuyruk yöneticileri tarafından desteklenmez.

Aşağıdaki MQGMO\_\* değerlerinin kullanılması, bir MQQueue.get() içinden bir MQException yayınına neden olabilir:

```
MQGMO_SYNCPOINT_IF_PERSISTENT
MQGMO_MARK_SKIP_BACKUT
MQGMO_BROWSE_MSG_UNDER_CURSOR
MQGMO_LOCK
MQGMO_UNLOCK
MQGMO_LOGICAL_ORDER
MQGMO_COMPLETE_MESSAGE
MQGMO_ALL_MSGS_AVALABILIR
MQGMO_ALL_SEGMENTS_AVALABILIR
MQGMO_UNMARKET_BROWSE_MSG
```

MQGMO\_MARK\_BROWSE\_HANDLE  
MQGMO\_MARK\_BROWSE\_CO\_OP  
MQGMO\_UNMARK\_BROWSE\_HANDLE  
MQGMO\_UNMARK\_BROWSE\_CO\_OP

Bunun yanı sıra, Java' tan kullanıldığında MQGMO\_SET\_SIGNAL desteklenmez.

*IBM MQ classes for Java'daki MQPMRF\_ \* değerlerine ilişkin kısıtlamalar*

Bunlar yalnızca dağıtım listesine ileti yerleştirilirken kullanılır ve yalnızca dağıtım listelerini destekleyen kuyruk yöneticileri tarafından desteklenmektedir. Örneğin, z/OS kuyruk yöneticileri dağıtım listelerini desteklememektedir.

*IBM MQ classes for Java'daki MQPMO\_ \* değerlerine ilişkin kısıtlamalar*

Bazı MQPMO\_ \* değerleri, tüm kuyruk yöneticileri tarafından desteklenmiyor

Aşağıdaki MQPMO\_ \* değerlerinin kullanılması, bir MQQueue.put() ya da MQQueueManager.put () içinden bir MQException yayınına neden olabilir:

MQPMO\_LOGICAL\_ORDER  
MQPMO\_NEW\_CORREL\_ID  
MQPMO\_NEW\_MESSAGE\_ID  
MQPMO\_RESOLVE\_LOCAL\_Q

*IBM MQ classes for Java'daki MQCNO\_ \* değerlerine ilişkin kısıtlamalar ve varyasyonlar*

Bazı MQCNO\_ \* değerleri desteklenmiyor.

- Otomatik istemci yeniden bağlanması IBM MQ classes for Java tarafından desteklenmez. Ayarladığınız MQCNO\_RECONNECT\_ \* değeri her ne olursa olsun, bağlantı MQCNO\_RECONNECT\_DISABLED' u ayarladığınız gibi hareket etmeye devam eder.
- MQCNO\_FASTPATH is ignored on queue managers that do not support MQCNO\_FASTPATH. İstemci bağlantıları tarafından da yoksayılır.

*IBM MQ classes for Java'daki MQRO\_ \* değerlerine ilişkin kısıtlamalar*

Aşağıdaki rapor seçenekleri ayarlanabilir.

MQRO\_EXCEPTION\_WITH\_FULL\_DATA  
MQRO\_EXPIRATION\_WITH\_FULL\_DATA  
MQRO\_COA\_WITHL\_FULL\_DATA  
MQRO\_COD\_WITH\_FULL\_DATA  
MQRO\_DISCARD\_MSG  
MQRO\_PASS\_DISCARD\_AND\_IFADESI

Daha fazla bilgi için bkz. [Rapor](#).

*z/OS ve diğer platformlarda IBM MQ classes for Java arasındaki çeşitli farklar*

IBM MQ for z/OS , bazı alanlardaki diğer platformlarda IBM MQ ' dan farklı şekilde hareket eder.

### **BackoutCount**

Bir z/OS kuyruk yöneticisi, ileti 255 kereden fazla yedeklense bile, en çok 255 BackoutCount değerini döndürür.

### **Varsayılan dinamik kuyruk öneki**

Bağ tanımları bağlantısı kullanılarak bir z/OS kuyruk yöneticisine bağlanıldığında, varsayılan dinamik kuyruk öneki CSQ. \* olur. Ters durumda, varsayılan dinamik kuyruk öneki AMQ. \* olur.

### **MQQueueManager oluşturucusu**

Client connect is not supported on z/OS. İstemci seçenekleri ile bağlantı kurma girişimi, MQCC\_FAILED ve MQRC\_ENVIRONMENT\_ERROR ile bir MQException ile sonuçlanmaya çalışılıyor.

MQQueueManager oluşturucusu, MQRC\_CHAR\_CONVERSION\_ERROR ile de başarısız olabilir ( IBM-1047 ve ISO8859-1 kod sayfaları arasında dönüştürme başlatılamazsa) ya da MQRC\_UCS2\_CONVERSION\_ERROR (kuyruk yöneticisinin kod sayfası ile Unicode arasında

dönüştürme başlatılamazsa). Uygulamanız bu neden kodlarından biriyle başarısız olursa, Dil Ortamı 'nın Ulusal Dil Kaynakları bileşeninin kurulu olduğundan emin olun ve doğru dönüştürme çizelgelerinin kullanılabilir olduğundan emin olun.

Unicode 'a ilişkin dönüştürme çizelgeleri, z/OS C/C++ isteğe bağlı özelliğinin bir parçası olarak kurulur. UCS-2 dönüştürmelerinin etkinleştirilmesiyle ilgili ek bilgi için *z/OS C/C++ Programming Guide*, SC09-4765adlı yayına bakın.

### **IBM MQ classes for Java' in çekirdek sınıflarının dışındaki özellikler**

IBM MQ classes for Java , tüm kuyruk yöneticileri tarafından desteklenmeyen API uzantılarını kullanmak için özel olarak tasarlanmış belirli işlevleri içerir. Bu konu grubunda, bunları desteklemeyen bir kuyruk yöneticisi kullanılırken nasıl davrandıkları anlatılır.

#### *MQQueueManager oluşturucu seçeneğindeki varyasyonlar*

MQQueueManager oluşturucularından bazıları isteğe bağlı bir tamsayı bağımsız değişkeni içerir. Bu bağımsız değişkenin bazı değerleri tüm altyapılarda kabul edilmez.

Bir MQQueueManager oluşturucusunun isteğe bağlı bir tamsayı bağımsız değişkeni içerdiğinde, bu, MQI ' in MQCNO seçenekleri alanına eşlenir ve normal ve hızlı yol bağlantısı arasında geçiş yapmak için kullanılır. Kullanılan tek seçenek MQCNO\_STANDARD\_BINDING ya da MQCNO\_FASTPATH\_BINDING ise, bu oluşturucunun 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, bunu destekleyen bir kuyruk yöneticisine yönelik bağ tanımları bağlantısıyla tanıtılır. Diğer ortamlarda bu değer yoksayılr.

#### *MQQueueManager.begin () yöntemine ilişkin kısıtlamalar*

Bu yöntem, bağ tanımları kipinde yalnızca UNIX, Linuxya da Windows sistemlerinde bir IBM MQ kuyruk yöneticisine karşı kullanılabilir. Ters durumda, MQRC\_ENVIRONMENT\_ERROR ile başarısız olur.

Daha ayrıntılı bilgi için bkz. [“IBM MQ classes for Javakullanarak JTA/JDBC eşgüdümü” sayfa 372 .](#)

#### *MQGetMessageSeçenekleri alanlarındaki çeşitlemeler*

Bazı kuyruk yöneticileri Sürüm 2 MQGMO yapısını desteklemez, bu nedenle bazı alanları varsayılan değerlerine ayarlamalısınız.

Sürüm 2 MQGMO yapısını desteklemeyen bir kuyruk yöneticisi kullanırken, aşağıdaki alanları varsayılan değerlerine ayarlı olarak bırakın:

GroupStatus  
SegmentStatus  
Bölümleme

Ayrıca, MatchOptions alanı yalnızca MQMO\_MATCH\_MSG\_ID ve MQMO\_MATCH\_COREL\_ID ' yi destekler. Bu alanlara desteklenmeyen değerler koyarsanız, sonraki MQDestination.get() işlemi MQRC\_GMO\_ERROR ile başarısız olur. Kuyruk yöneticisi Sürüm 2 MQGMO yapısını desteklemiyorsa, başarılı bir MQDestination.get() işleminden sonra bu alanlar güncellenmez.

#### *IBM MQ classes for Java' içindeki dağıtım listelerindeki kısıtlamalar*

Tüm kuyruk yöneticileri bir MQDistributionListolanağını açmanıza izin vermez.

Dağıtım listeleri yaratmak için aşağıdaki sınıflar kullanılır:

MQDistributionList  
MQDistributionListÖgesi  
MQMessageTracker

You can create and populate MQDistributionLists and MQDistributionListItems in any environment, but not all queue managers allow you to open an MQDistributionList. Özellikle, z/OS kuyruk yöneticileri dağıtım listelerini desteklememektedir. MQRC\_OD\_ERROR içinde bu tür bir kuyruk yöneticisi sonucu kullanılırken bir MQDistributionList açılmaya çalışılıyor.

### *MQPutMessageSeçenekleri alanlarındaki çeşitlendmeler*

Bir kuyruk yöneticisi dağıtım listelerini desteklemiyorsa, bazı MQPMO alanları farklı davranılır.

MQPMO 'daki dört alan, MQPutMessageSeçenekleri sınıfında aşağıdaki üye değişkenleri olarak görsel olarak gerçekleştirilir:

knownDestSayısı  
unknownDestSayısı  
invalidDestSayı  
recordFields

Bu alanlar öncelikli olarak dağıtım listeleriyle kullanılmak üzere tasarlanmıştır. Ancak, dağıtım listelerini destekleyen bir kuyruk yöneticisi, bir MQPUT 'dan sonra tek bir kuyruğa gelen DestCount alanlarını da doldurur. Örneğin, kuyruk bir yerel kuyruğa çözümlerse, knownDestSayı 1 olarak ayarlanır ve diğer iki sayı alanı 0 olarak ayarlanır.

Kuyruk yöneticisi dağıtım listelerini desteklemiyorsa, bu değerler aşağıdaki gibi benzetimli olur:

- Put () başarılı olursa, unknownDestSayısı 1 olarak ayarlanır ve diğerleri 0 değerine ayarlanır.
- Put () başarısız olursa, invalidDestSayı 1 olarak ayarlanır ve diğerleri 0 değerine ayarlanır.

Dağıtım listeleriyle birlikte recordFields değişkeni kullanılır. Ortam ne olursa olsun, herhangi bir zamanda recordFields içine bir değer yazılabilir. MQPutMessageOptions nesnesi, MQDistributionList.put () yerine sonraki bir MQDestination.put() ya da MQQueueManager.put () üzerinde kullanılırsa yoksayılır.

### *Restrictions in MQMD fields with IBM MQ classes for Java*

İleti bölümlenmesiyle ilgili bazı MQMD alanları, segmentasyonu desteklemeyen bir kuyruk yöneticisi kullanılırken varsayılan değerlerinde bırakılmalıdır.

Aşağıdaki MQMD alanları büyük oranda ileti bölümlenmesiyle ilgilendir:

GroupId  
MsgSeqNumarası  
Görelî Konum  
MsgFlags  
OriginalLength

Bir uygulama bu MQMD alanlarından herhangi birini varsayılan değerlerine göre ayarlarsa ve bu alanları desteklemeyen bir kuyruk yöneticisine bir put () ya da get () işlemi yaparsa, put () ya da get (), MQRC\_MD\_ERROR ile MQException ortaya çıkar. Böyle bir kuyruk yöneticisiyle başarılı bir put () ya da get (), her zaman MQMD alanlarını varsayılan değerlerine ayarlanmış olarak bırakır. İleti gruplamasını ve kesimlere ayırma özelliğini desteklemeyen bir kuyruk yöneticisine karşı çalışan bir Java uygulamasına gruplandırılmış ya da bölümlenmiş bir ileti göndermeyin.


Bir Java uygulaması, bu alanları desteklemeyen bir kuyruk yöneticisinden ileti almaya çalışırsa ve alınacak fiziksel ileti bir bölümlenmiş ileti grubunun bir parçasıysa (yani, MQMD alanları için varsayılan olmayan değerlere sahip), hata olmadan alınır. Ancak, MQMessage 'daki MQMD alanları güncellenmez; MQMessage biçimi özelliği MQFMT\_MD\_EXTENSION değerine ayarlıdır ve gerçek ileti verilerinin başında, yeni alanların değerlerini içeren bir MQMDE yapısı bulunur.

### **CICS Transaction Server altında IBM MQ classes for Java ile ilgili kısıtlamalar**

CICS Transaction Server for z/OS ortamında, yalnızca ana (birinci) iş parçacığının CICS ya da IBM MQ çağrılarını yayınına izin verilir.

Note, that IBM MQ JMS classes are not supported for use in a CICS Java application.

Bu nedenle, bu ortamdaki iş parçacıkları arasında MQQueueManager ya da MQQueue nesnelerini paylaşmak ya da alt iş parçacığının üzerinde yeni bir MQQueueManager yaratmak olanaklı değildir.

 [“z/OS ve diğer platformlarda IBM MQ classes for Java arasındaki çeşitli farklar” sayfa 412](#), bir z/OS kuyruk yöneticisine karşı çalışırken IBM MQ classes for Java için geçerli olan bazı kısıtlamaları ve çeşitlendmeleri belirtir. Buna ek olarak, CICS altında çalışırken, MQQueueManager üzerindeki

hareket denetim yöntemleri desteklenmez. MQQueueManager.commit () ya da MQQueueManager.backout () komutunu vermek yerine, uygulamalar JCICS görev eşitleme yöntemlerini ( Task.commit() ve Task.rollback() ) yöntemini kullanır. Görev sınıfı, com.ibm.cics.server paketindeki JCICS tarafından sağlanır.

## IBM MQ kaynak bağıdaştırıcısının kullanılması

Kaynak bağıdaştırıcısı, bir uygulama sunucusunda çalışan uygulamaların IBM MQ kaynaklarına erişmelerini sağlar. Gelen ve giden iletişimi destekler.

### Kaynak bağıdaştırıcısının içerdiği

Java Platform, Enterprise Edition Connector Architecture (JCA), Java EE ortamında çalışan uygulamaların IBM MQ ya da Db2 gibi bir Enterprise Information System (EIS) ortamına bağlanmasını standart bir şekilde sağlar. IBM MQ kaynak bağıdaştırıcısı, JCA 1.7 arabirimlerini uygular ve IBM MQ classes for JMS ögesini içerir. It allows JMS applications and message driven beans (MDBs), running in an application server, to access the resources of an IBM MQ queue manager. Kaynak bağıdaştırıcısı, hem noktadan noktaya iletişim etki alanını, hem de yayınlama/abone olma etki alanını destekler.

IBM MQ kaynak bağıdaştırıcısı, bir uygulama ile kuyruk yöneticisi arasında iki iletişim tipini destekler:

#### Giden iletişim

An application starts a connection to a queue manager, and then sends JMS messages to JMS destinations and receives JMS messages from JMS destinations in a synchronous manner.

#### Gelen iletişim

JMS hedefine ulaşan bir JMS iletisi, iletiyi zamanuyumsuz olarak işleyen bir MDB ' ye teslim edilir.

Kaynak bağıdaştırıcısı IBM MQ classes for Java de içerir. Sınıflar otomatik olarak, kaynak bağıdaştırıcısının konuşlandırıldığı bir uygulama sunucusunda çalışan uygulamalara otomatik olarak sunulur ve o uygulama sunucusunda çalışan uygulamaların IBM MQ API ' nin kaynaklarına erişirken IBM MQ classes for Java API ' yi kullanmasına izin verir.

The use of the IBM MQ classes for Java within a Java EE environment is supported with restrictions. Bu kısıtlamalar hakkında bilgi için bkz. [“Running IBM MQ classes for Java applications within Java EE” sayfa 323.](#)

## Kullanılacak kaynak bağıdaştırıcısının hangi sürümü

Kullandığınız uygulama sunucusunun Java Platform, Enterprise Edition (Java EE) sürümü, kullanmanız gereken kaynak bağıdaştırıcısının sürümünü belirler:

### Java EE 7

IBM MQ 8.0 ve daha sonraki kaynak bağıdaştırıcısı, JCA v1.7 ' yi destekler ve JMS 2.0 desteği sağlar. Bu kaynak bağıdaştırıcısının bir Java EE 7 ve sonraki uygulama sunucusu içinde konuşlandırılması gerekir (bkz. [“IBM MQ kaynak bağıdaştırıcısı desteği bildirim” sayfa 416.](#))

You can install the IBM MQ 8.0 or later resource adapter on any application server that is certified as compliant with the Java Platform, Enterprise Edition 7 specification. Bir uygulama, IBM MQ 8.0 ya da daha sonraki bir kaynak bağıdaştırıcısını kullanarak, bir IBM WebSphere MQ 7.0 ya da sonraki bir kuyruk yöneticisine BAĞLAMAYLAR ya da CLIENT iletimi kullanılarak ya da yalnızca CLIENT iletimi kullanılarak bir IBM WebSphere MQ 6.0 kuyruk yöneticisine bağlanabilir.

**Önemli:** IBM MQ 8.0 ya da daha sonraki bir kaynak bağıdaştırıcısı, yalnızca JMS 2.0' i destekleyen bir uygulama sunucusuna konuşlandırılabilir.

### Java EE 5 ve Java EE 6

IBM WebSphere MQ 7.5 kaynak bağıdaştırıcısı Java EE Connector Architecture (JCA) v1.5 ' i destekler ve JMS 1.1 desteği sağlar. WebSphere Liberty ile tam bütünleştirme sağlamak için, IBM WebSphere MQ 7.5 kaynak bağıdaştırıcısı IBM WebSphere MQ 7.5.0 Fix Pack 2' den [APAR IC92914](#) olarak güncellenir. Bu kaynak bağıdaştırıcısı, diğer Java EE 5 ve sonraki uygulama sunucularıyla tam uyumluluğu korur (bkz. [WebSphere MQ kaynak bağıdaştırıcısı v7.1 ve sonraki destek bildirim](#)).

## Kaynak baędařtırıcısını WebSphere Application Server traditionalile birlikte kullanma

IBM MQ 9.0' tan IBM MQ kaynak baędařtırıcısı, WebSphere Application Server traditional 9.0 ya da sonraki bir yayın düzeyiyle önceden kurulmuş olur. Bu nedenle, yeni bir kaynak baędařtırıcısı takmaya gerek yoktur.

**Not:** Bir IBM MQ 9.0 ya da daha sonraki bir kaynak baędařtırıcısı, CLIENT ya da BRETLE aktarma kipinde hizmet içi IBM MQ kuyruk yöneticisine baęlanabilir.

## Kaynak baędařtırıcısını WebSphere Libertyile birlikte kullanma

IBM MQ 'dan WebSphere Liberty' a baęlanmak için IBM MQ kaynak baędařtırıcısını kullanmanız gerekir. Liberty , IBM MQ kaynak baędařtırıcısını içermedięinden, bunu Fix Central' dan ayrı olarak edinmeniz gerekir. Kullandığınız kaynak baędařtırıcısının sürümü, uygulama sunucusunun Java EE sürümüne baęlıdır.

Kaynak baędařtırıcısının nasıl yükleneceęi ve kurulacaęı hakkında daha fazla bilgi için bkz. [“Kaynak baędařtırıcısının Libertyiçine kurulması” sayfa 423.](#)

### İlgili kavramlar

[“Kaynak baędařtırıcısının gelen iletiřim için yapılandırılması” sayfa 429](#)

Gelen iletiřimi yapılandırmak için bir ya da daha çok ActivationSpec nesnesinin özelliklerini tanımlayın.

[“Kaynak baędařtırıcısının giden iletiřim için yapılandırılması” sayfa 445](#)

Giden iletiřim yapılandırmak için bir ConnectionFactory nesnesinin özelliklerini ve denetlenen bir hedef nesneyi tanımlayın.

[“kullanmaIBM MQ classes for JMS” sayfa 77](#)

IBM MQ classes for Java Message Service (IBM MQ classes for JMS), IBM MQ ile birlikte verilen JMS sağlayıcısıdır. javax.jms paketinde tanımlanan arabirimlerin yanı sıra, IBM MQ classes for JMS , JMS API ' ye iki uzantı kümesi sağlar.

[“kullanmaIBM MQ classes for Java” sayfa 321](#)

Java ortamında IBM MQ kullanın. IBM MQ classes for Java allow a Java application to connect to IBM MQ as an IBM MQ client, or connect directly to an IBM MQ queue manager.

### İlgili görevler

[Uygulama sunucusunun en son kaynak baędařtırıcısı bakım düzeyini kullanacak şekilde yapılandırılması](#)

[IBM MQ kaynak baędařtırıcısı için sorun belirleme](#)

### WebSphere Application Server Konular

[IBM MQ kaynak baędařtırıcısını koruma](#)

[Deploying JMS applications to Liberty to use the IBM MQ messaging provider](#)

## IBM MQ kaynak baędařtırıcısı desteęi bildirim

IBM MQ 8.0 ya da daha sonraki bir yayın düzeyiyle birlikte gelen kaynak baędařtırıcısı, JMS 2.0 belirtimini uygular. Yalnızca Java Platform, Enterprise Edition 7 (Java EE 7) uyumlu bir uygulama sunucusuna konuřlandırılabilir ve bu nedenle JMS 2.0' ı destekler.

Sertifikalı uygulama sunucularının bir listesi, [Oracle' ın web sitesindedir.](#)

## WebSphere Libertyiçinde devreye alma

WebSphere Liberty 8.5.5 Fix Pack 6 ve daha sonraki bir sürümü, WebSphere Application Server Liberty 9.0 ve daha sonraki bir sürümü Java EE 7 sertifikalı uygulama sunucularıdır; bu nedenle IBM MQ 9.0 kaynak baędařtırıcısı bunlara konuřlandırılabilir.

WebSphere Liberty , kaynak baędařtırıcılarıyla çalışmak için kullanılabilir iki özellięe sahiptir:

- JMS 1.1 kaynak baędařtırıcılarıyla çalışmaya izin vermek için wmqJmsClient-1.1 özellięi.
- JMS 2.0 kaynak baędařtırıcılarıyla çalışmaya izin vermek için wmqJmsClient-2.0 özellięi.



**Önemli:** IBM MQ 8.0 ya da daha sonraki bir kaynak bağdaştırıcısı, wmqJmsClient-2.0 özelliğiyle konuşlandırılmalıdır.

Bu yapılandırmayla ilgili bilgiler [Connecting WebSphere Liberty Liberty to IBM MQ](#) senaryosunda yer alıyor.

## WebSphere Application Server traditional içinde devreye alma

WebSphere Application Server traditional 9.0 , kurulu bir IBM MQ 9.0 kaynak bağdaştırıcısı ile birlikte sağlanır. Bu nedenle, yeni bir kaynak bağdaştırıcısı takmaya gerek yoktur. Kurulu kaynak bağdaştırıcısı, desteklenen bir IBM MQ ya da IBM WebSphere MQ sürümünde çalışmakta olan kuyruk yöneticilerine CLIENT ya da BRELIPLA iletim kipinde bağlanabilir. Daha fazla bilgi için bkz [“IBM MQ 8.0 ya da sonraki kuyruk yöneticilerine bağlanırlık” sayfa 417.](#)

**Önemli:** The IBM MQ 9.0 resource adapter cannot be deployed into versions of WebSphere Application Server traditional before IBM MQ 9.0, as these versions are not Java EE 7 certified.

Desteklenen herhangi bir WebSphere Application Server sürümü, desteklenen herhangi bir IBM MQ sürümüne bağlanmak için bu ürünle birlikte paketlenmiş IBM MQ kaynak bağdaştırıcısını kullanabilir.

## Kaynak bağdaştırıcısının diğer uygulama sunucularıyla kullanılması

For all other Java EE 7 compliant application servers, problems that occur following the successful completion of the IBM MQ resource adapter [Kuruluş Doğrulama Testi \(IVT\)](#) can be reported to IBM for the investigation of IBM MQ product trace and other IBM MQ diagnostic information. IBM MQ kaynak bağdaştırıcısı IVT başarıyla çalıştırılmazsa, saptanan tüm sorunlar, uygulama sunucusuna özgü yanlış konuşlandırma ya da yanlış kaynak tanımlarından kaynaklanıyor olabilir ve sorunlar, uygulama sunucusu belgeleri ve/ya da o uygulama sunucusu için destek kuruluşu kullanılarak soruşturulmalıdır.

## Java Yürütme Ortamı

Uygulama sunucusunu çalıştırmak için kullanılan Java Runtime (JRE), IBM MQ 9.0 ya da sonraki istemciyle desteklenen bir JRE olmalıdır. Daha fazla bilgi için bkz [IBM MQ](#). (Daha sonra görmek istediğiniz sürümü ve işletim sistemini ya da bileşen raporunu seçin ve **Supported Software** (Desteklenen Yazılım) sekmesinin altında listelenen **Java** bağlantısını izleyin.)

## IBM MQ 8.0 ya da sonraki kuyruk yöneticilerine bağlanırlık

The full range of JMS 2.0 functionality is available when connecting to an IBM MQ 8.0 or later queue manager by using the resource adapter that has been deployed into a Java EE 7 certified application server. WebSphere Application Server ile birlikte gönderilen kaynak bağdaştırıcısının sürümleriyle ilgili daha fazla bilgi için bkz. teknik not [WebSphere MQ Resource Adapter \(RA\) ürününün hangi sürümü WebSphere Application Server ile birlikte teslim edilir?](#).

To make use of the JMS 2.0 functionality, the resource adapter needs to connect to the queue manager by using IBM MQ messaging provider normal mode. Daha fazla bilgi için bakınız: [Configuring the JMS PROVIDERVERSION](#) property.

## IBM WebSphere MQ 7.5 ya da daha önceki kuyruk yöneticilerine bağlanırlık

IBM MQ 9.0 ya da daha sonraki bir kaynak bağdaştırıcısının JMS 2.0 ' u destekleyen bir Java EE 7 sertifikalı uygulama sunucusuna konuşlandırılması ve bu kaynak bağdaştırıcısının IBM WebSphere MQ 7.5 ya da daha önceki bir kuyruk yöneticisine bağlanması desteklenir. Kullanılabilir işlevsellik, kuyruk yöneticisinin yetenekleriyle sınırlıdır. Daha fazla bilgi için bakınız: [Configuring the JMS PROVIDERVERSION](#) property.

## MQ Uzantıları

JMS 2.0 belirtimi, belirli davranışların nasıl çalıştığıyla ilgili değişiklikleri tanıtır. Because IBM MQ 8.0 or later implements this specification, there are changes in behavior between IBM MQ 8.0 and later and versions before IBM MQ 8.0. In IBM MQ 8.0 and later, the IBM MQ classes for JMS include support for the

Java system property `com.ibm.mq.jms.SupportMQExtensions` that, when set to TRUE, causes these versions of IBM MQ to revert these behaviors to those of IBM WebSphere MQ 7.5 or earlier. Özelliğin varsayılan değeri FALSE'dir.

The IBM MQ 9.0 or later resource adapter also includes a resource adapter property called `supportMQExtensions` that has the same effect and default value as the `com.ibm.mq.jms.SupportMQExtensions` Java system property. Bu kaynak bağdaştırıcısı özelliği, varsayılan olarak `ra.xml` 'de false olarak ayarlanır.

Hem kaynak bağdaştırıcısı özelliği, hem de Java sistem özelliği ayarlandıysa, sistem özelliği öncelikli olur.

WebSphere Application Server traditional 9.0 içinde konuşlandırılmış olan kaynak bağdaştırıcısı içinde, geçiş işlemi için bu özelliğin otomatik olarak TRUE değerine ayarlandığına dikkat edin.

Daha fazla bilgi için bkz "[SupportMQExtensions özelliği](#)" sayfa 304.

## Genel sorunlar

### Oturum ara bırakma desteklenmiyor

Some application servers provide a capability called session interleaving, where the same JMS session can be used in multiple transactions, although it is only enlisted in one at a time. IBM MQ kaynak bağdaştırıcısı, aşağıdaki sorunlara yol açabilen bu yeteneği desteklemez:

Bir MQ kuyruğuna ileti koyma girişimi, neden kodu 2072 ile başarısız olur (MQRC\_SYNCPOINT\_NOT\_AVAILABLE).

`xa_close()` çağrıları, neden kodu -3 (XAER\_PROTO) ile başarısız olur ve uygulama sunucusundan erişilmekte olan IBM MQ kuyruk yöneticisinden AT040010 bağlantı denetimi tanıtıcısına sahip bir FDC oluşturulur. Bu yetenekten nasıl geçersiz kılınabilmeye ilişkin bilgi için uygulama sunucusu belgelerinize bakın.

### XA hareketlerinin XA hareketlerinin kurtarılması için nasıl kurtarılacağını içeren Java Transaction API (JTA) belirtimi

JTA belirtiminin 3.4.8 bölümü XA işlemsel kurtarma gerçekleştirmek için XA kaynaklarının yeniden yaratılacağı belirli bir mekanizmayı tanımlamaz. Bu nedenle, XA hareketlerinde yer alan XA kaynaklarının kurtarıldığı her bir hareket yöneticisine (ve dolayısıyla uygulama sunucusu) bağlı olur. Bazı uygulama sunucularında IBM MQ 9.0 kaynak bağdaştırıcısı, XA işlemsel kurtarma işlemi gerçekleştirmek için kullanılan uygulama sunucusuna özgü mekanizmaları gerçekleştirmez.

### ManagedConnection Üreticisinde eşleşen bağlantılar

Bir uygulama sunucusu, IBM MQ kaynak bağdaştırıcısı tarafından sağlanan bir `ManagedConnectionFactory` örneğinde `matchManagedConnections` yöntemini çağırabilir. Yöntem, uygulama sunucusu tarafından yonteme geçirilen hem **`javax.security.auth.Subject`** hem de **`javax.resource.spi.ConnectionRequestInfo`** bağımsız değişkenleriyle eşleşen bir yöntem bulursa `ManagedConnection` döndürülür.

## IBM MQ kaynak bağdaştırıcısına ilişkin sınırlamalar

IBM MQ kaynak bağdaştırıcısı tüm IBM MQ platformlarında desteklenir. Ancak, IBM MQ kaynak bağdaştırıcısını kullandığınızda, IBM MQ 'un bazı özellikleri kullanılamaz ya da sınırlandırılır.

IBM MQ kaynak bağdaştırıcısı aşağıdaki sınırlamalara sahiptir:

- IBM MQ 8.0, 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 sonraki bir kaynak bağdaştırıcısı bir Java EE 7 ya da daha sonraki bir uygulama sunucusunda kurulu olmalıdır. İstemci ya da bağ tanımları taşıma kipinde herhangi bir hizmet içi kuyruk yöneticisinde bağlanabilir.
- When running inside the WebSphere Liberty application server, the stabilized IBM MQ classes for Java are not supported. Diğer uygulama sunucuları içinde IBM MQ classes for Java 'un kullanım için önerilmez. See the IBM technote [J2EE/JEE Ortamlarındaki WebSphere MQ Java Arabirimlerini Kullanma](#) for details of IBM MQ classes for Java considerations within Java EE.
- z/OS üzerinde WebSphere Liberty uygulama sunucusu içinde çalışırken, `wmqJmsClient-2.0` özelliği kullanılmalıdır. Genel JCA desteği z/OS için mümkün değildir.

- IBM MQ kaynak bağıdaştırıcısı, Javadişındaki dillerde yazılmış kanal çıkış programlarını desteklemez.
- Bir uygulama sunucusu çalışırken, tüm JCA kaynakları için sslFipsRequired özelliğinin değeri true (doğru) ya da tüm JCA kaynakları için false (yanlış) olmalıdır. JCA kaynakları koşut zamanlı olarak kullanılmasa da bu bir gereksinimdir. If the sslFipsRequired property has different values for different JCA resources, IBM MQ issues the reason code MQRC\_UNSUPPORTED\_CIPHER\_SUITE, even if a TLS connection is not being used.
- Bir uygulama sunucusu için birden çok anahtar deposu belirtebilirsiniz. Bağlantılar birden çok kuyruk yöneticisinden yapılırsa, tüm bağlantıların aynı anahtar deposunu kullanması gerekir. Bu sınırlama WebSphere Application Server için geçerli değildir.
- İstemci kanal tanımlama çizelgesi (CCDT), birden çok uygun istemci bağlantısı kanal tanımlamasıyla birlikte kullanırsanız, kaynak bağıdaştırıcısının farklı bir kanal tanımlaması ve dolayısıyla CCDT ' den farklı bir kuyruk yöneticisi seçmesi durumunda, işlem kurtarma sorunlarına neden olabilir. Kaynak bağıdaştırıcısı, böyle bir yapılandırmanın kullanılmasını önlemek için herhangi bir işlem yapmaz ve hareket kurtarmasına ilişkin sorunlara neden olabilecek yapılandırmalardan kaçınmak sizin sorumluluğunuzda olur.
- Bir Java EE kapsayıcısında (EJB/Servlet) çalışırken giden bağlantılar için IBM WebSphere MQ 7.0.1 içinde sunulan bağlantı yeniden deneme işlevselliği desteklenmez. Connection retry is not supported at all for outbound JMS when the adapter is used in a JEE container context, regardless of transaction configuration or for non-transacted use.
- Re-authentication, as defined in Section 9.1.9 of the Java EE Connector Architecture version 1.7 specification, of JMS connections is not supported. IBM MQ kaynağı içindeki ra.xml dosyası, **reauthentication-support** adlı özelliğın false değerine ayarlı olması gerekir. An attempt by the application server to re-authenticate a JMS connection results in the IBM MQ resource adapter throwing a javax.resource.spi.SecurityException with the MQJCA1028 message code.

### İlgili görevler

MQI istemcisinde çalıştırma sırasında yalnızca FIPS onaylı CipherSpecs ' in kullanıldığını belirtme

### İlgili başvurular

[Federal Information Processing Standards \(FIPS\) for UNIX, Linux and Windows](#)

## WebSphere Application Server ve IBM MQ kaynak bağıdaştırıcısı

IBM MQ kaynak bağıdaştırıcısı, WebSphere Application Server' de IBM MQ ileti alışverişi sağlayıcısıyla JMS ileti sistemini gerçekleştiren uygulamalar tarafından kullanılır.

**Önemli:** Do not use the IBM MQ or IBM WebSphere MQ resource adapter with WebSphere Application Server 6.0 or WebSphere Application Server 6.1.

WebSphere Application Server 7.0 ve WebSphere Application Server 8.0 , IBM WebSphere MQ 7.0 kaynak bağıdaştırıcısına ilişkin bir sürümünü içerir.

WebSphere Application Server 8.5.5 , IBM WebSphere MQ 7.1 kaynak bağıdaştırıcısına ilişkin bir sürümünü içerir.

WebSphere Application Server traditional 9.0 , IBM MQ 9.0 kaynak bağıdaştırıcısının bir sürümünü içerir. IBM MQ 9.0 ya da daha sonraki bir kaynak bağıdaştırıcısı Java EE 7 onaylı olmadığından, önceki WebSphere Application Serversürümlerine konuşlandırılmaz.

WebSphere Application Server için bir IBM MQ kuyruk yöneticisinin kaynaklarına erişmek için bir JMS uygulaması kullanmak istiyorsanız, WebSphere Application Server için IBM MQ ileti alışverişi sağlayıcısını kullanın. IBM MQ ileti alışverişi sağlayıcısı, IBM MQ classes for JMS sürümünü içerir. Daha fazla bilgi için bkz. teknik not [WebSphere MQ Resource Adapter \(RA\) ürününün hangi sürümü WebSphere Application Server ile birlikte teslim edilir?](#)

**Önemli:** Uygulamanızın içinde IBM MQ classes for JMS ya da IBM MQ classes for Java JAR dosyalarının hiçbirini eklemeyin. Bunu yapmak ClassCastKural Dışı Durumlarıyla sonuçlanabilir ve bakımı zor olabilir.

## Liberty ve IBM MQ kaynak bağıdaştırıcısı

IBM MQ kaynak bağıdaştırıcısı, kurmakta olduğunuz kaynak bağıdaştırıcısının hangi sürümüne bağılı olarak, wmqJmsClient-1.1 ya da wmqJmsClient-2.0 özelliğini kullanarak WebSphere Liberty 8.5.5 Fix Pack 2 ya da sonraki bir sürüme kurulabilir. Diđer bir seçenek olarak, bazı kısıtlamalara tabi olarak, soysal Java Platform, Enterprise Edition Connector Architecture (Java EE JCA) desteğini kullanarak kaynak bağıdaştırıcısını kurabilirsiniz.

## Kaynak bağıdaştırıcısını Liberty için kurarken genel sınırlamalar

wmqJmsClient-1.1 ya da wmqJmsClient-2.0 özelliğini kullanırken ve soysal JCA desteğini kullanırken, kaynak bağıdaştırıcısı için aşağıdaki kısıtlamalar geçerlidir:

- IBM MQ classes for Java , Liberty içinde desteklenmiyor. Bunlar, IBM MQ Liberty ileti sistemi özelliğiyle ya da genel JCA desteği ile kullanılmamalıdır. Daha fazla bilgi için [J2EE/JEE Ortamlarında WebSphere MQ Java Arabirimlerini Kullanmabaşlıklı konuya](#) bakın.
- IBM MQ kaynak bağıdaştırıcısı, BINDINGS\_THEN\_CLIENT iletim tipine sahiptir. Bu iletim tipi, IBM MQ Liberty ileti sistemi özelliği içinde desteklenmez.
- IBM MQ 9.0 öncesinde, Advanced Message Security (AMS) özelliği, IBM MQ Liberty ileti sistemi özelliğine dahil edilmemesiydi. Ancak, AMS , bir IBM MQ 9.0 ya da daha sonraki kaynak bağıdaştırıcısıyla desteklenir.

## Liberty özelliklerini kullanırken kısıtlamalar

WebSphere Liberty 8.5.5 Fix Pack 2 ile WebSphere Liberty 8.5.5 Fix Pack 5 arasında bu değerler de içinde olmak üzere, yalnızca wmqJmsClient-1.1 özelliği kullanılabiliyordu ve yalnızca JMS 1.1 kullanılabilir. WebSphere Liberty 8.5.5 Fix Pack 6 , wmqJmsClient-2.0 özelliğini, JMS 2.0 kullanılabileceği şekilde ekledi.

Ancak kullanmanız gereken özellik, kullandığınız kaynak bağıdaştırıcısının hangi sürümüne bağılı olarak değişir:

- IBM WebSphere MQ 7.5.0 Fix Pack 6 ve sonraki IBM WebSphere MQ 7.5 kaynak bağıdaştırıcısı, yalnızca wmqJmsClient-1.1 özelliğiyle kullanılabilir.
- IBM MQ 8.0.0 Fix Pack 3 ve sonraki IBM MQ 8.0 kaynak bağıdaştırıcısı yalnızca wmqJmsClient-2.0 özelliği ile kullanılabilir.
- IBM MQ 9.0 kaynak bağıdaştırıcısı yalnızca wmqJmsClient-2.0 özelliği ile kullanılabilir.

## Soysal JCA desteğini kullanılırken kısıtlamalar

Soysal JCA desteğini kullanıyorsanız, aşağıdaki kısıtlamalar geçerli olur:

- Soysal JCA desteğini kullanılırken JMS düzeyini belirtmeniz gerekir:
  - JMS 1.1 ve JCA 1.6 , yalnızca IBM WebSphere MQ 7.5.0 Fix Pack 6 ve sonraki IBM WebSphere MQ 7.5 kaynak bağıdaştırıcılarıyla birlikte kullanılmalıdır.
  - JMS 2.0 ve JCA 1.7 , yalnızca IBM MQ 8.0.0 Fix Pack 3 ve sonraki IBM MQ 8.0 kaynak bağıdaştırıcılarıyla birlikte kullanılmalıdır.
- Soysal JCA desteğini kullanarak z/OS üzerinde IBM MQ kaynak bağıdaştırıcısını çalıştırmak olanaklı değildir. In order to run the IBM MQ resource adapter on z/OS, it must be run with the wmqJmsClient-1.1 or wmqJmsClient-2.0 feature.
- Kaynak bağıdaştırıcısının yeri, aşağıdaki xml ögesi kullanılarak belirtilir:

```
<resourceAdapter id="mqJms" location="{server.config.dir}/wmq.jmsra.rar">  
  <classloader apiTypeVisibility="spec, ibm-api, api, third-party"/>  
</resourceAdapter>
```

**Önemli:** Tanıtıcı etiketinin değeri, wmqJms için EXCEPT dışında herhangi bir değer olabilir. Tanıtıcı olarak wmqJms kullanıyorsanız, Liberty kaynak bağıdaştırıcısını doğru biçimde yükleyemez. This is because

wmqJms is the ID that is used internally to refer to the specific feature for IBM MQ. Aslında bir NullPointer kural dışı durumu yaratır.

Aşağıdaki örneklerde, bir `server.xml` dosyasından bazı parçacıklar gösterilir:

```
<!-- 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 kullanımına ve wmqJmsClient-2.0 özelliğinin eksikliğini not edin.

```
<resourceAdapter id="mqJms" location="{server.config.dir}/wmq.jmsra.rar">
  <classloader apiTypeVisibility="spec, ibm-api, api, third-party"/>
</resourceAdapter>
```

**İpucu:** Tercih edilen tanıtıcı için mqJms ' un kullanılmasına 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 ' i mqJmstanıtıcısı aracılığıyla kaynak bağdaştırıcısına geri not edin; bu, IBM MQ ' in belirli sınıfların yüklenmesine izin vermek için.

## Soysal JCA desteği kullanılarak izleme sırasında kısıtlamalar

İzleme ve günlüğe kaydetme, Liberty izleme sistemi içinde bütünleştirilmedi. Instead, the IBM MQ resource adapter trace must be enabled using either Java system properties, or an IBM MQ classes for JMS configuration file, as described in [JMS uygulamaları için IBM MQ sınıflarının izlenmesi](#). Liberty' ta Java sistem özelliklerinin nasıl ayarlanabilmesiyle ilgili ayrıntılar için [WebSphere Liberty belgelerine](#) bakın.

Örneğin, Liberty 19.0.0.9' da IBM MQ kaynak bağdaştırıcısının izlenmesini etkinleştirmek için Liberty kütüğüne bir giriş ekleyin: `jvm.options`:

1. `jvm.options` adlı bir metin dosyası oluşturun.
2. Her satırda bir satır olmak üzere izlemeyi 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, şu adresi kullanarak `jvm.options` kaydedin:

```
{server.config.dir}/jvm.options
```

Bu değişiklikleri tüm Liberty' de uygulamak için, şu adresi kullanarak `jvm.options` kaydedin:

```
{wlp.install.dir}/etc/jvm.options
```

Bu işlem, yerel olarak tanımlanmış bir `jvm.options` dosyası olmayan tüm JVM ' ler için geçerli olur.

4. Değişiklikleri etkinleştirmek için sunucuyu yeniden başlatın.

Bu sonuç, `<path_to_trace_to>` dizininde `MQRA-WLP_<process identifier>.trc` adlı bir izleme dosyasına yazılmakta olan izne göre sonuçlanıyor.

## İstemci kanal tanımlama çizelgelerine sahip tam Liberty XA desteği

V 9.1.2

WebSphere Liberty 18.0.0.2 kullanıldığında, IBM MQ 9.1.2 ile, XA hareketleriyle bağlantılı olarak istemci kanal tanımlama çizelgesi (CCDT) içinde kuyruk yöneticisi gruplarını kullanabilirsiniz. Bu, hareket bütünlüğünü korurken kuyruk yöneticisi grupları tarafından sağlanan iş yükü dağıtımını ve kullanılabilirliğini artık kullanıma sunmanın mümkün olduğu anlamına gelir.

Kuyruk yöneticisinde bağlantı hataları olması durumunda, kuyruk yöneticisinin yeniden kullanılabilir olması gerekir. Böylece, işlem çözülebilir. Hareket kurtarma işlemi Liberty tarafından yönetilir ve kuyruk yöneticilerinin yeniden kullanılabilir olması için uygun bir süre için izin verilmesi için hareket yöneticisini yapılandırmanız gerekebilir. Ek bilgi için, WebSphere Liberty ürün belgelerindeki [Transaction manager \(transaction\)](#) başlıklı konuya bakın.

Bu bir istemci tarafı özelliğidir; bir IBM MQ 9.1.2 kuyruk yöneticisi değil, bir IBM MQ 9.1.2 kaynak bağdaştırıcısına gereksinim duyarsınız.

## IBM MQ kaynak bağdaştırıcısının takılması

IBM MQ kaynak bağdaştırıcısı bir kaynak arşivi (RAR) dosyası olarak sağlanır. Uygulama sunucunuzda RAR dosyasını kurun. Dizin eklemek için sistem yoluna dizin eklemeniz gerekebilir.

### Bu görev hakkında

IBM MQ kaynak bağdaştırıcısı, `wmq.jmsra.rar` adlı bir kaynak arşivi (RAR) dosyası olarak sağlanır. RAR dosyası, IBM MQ classes for JMS ve Java EE Connector Architecture (JCA) arabirimlerinin IBM MQ uygulamasını içerir.

When you install the resource adapter as part of the IBM MQ product installation, `wmq.jmsra.rar` is installed with IBM MQ classes for JMS in the directory shown in [Çizelge 60 sayfa 422](#).

<i>Çizelge 60. Her platform için <code>wmq.jmsra.rar</code> dosyası içeren dizin</i>	
<b>Platform</b>	<b>Dizin</b>
UNIX and Linux	<code>MQ_INSTALLATION_PATH/java/lib/jca</code>
IBM i	<code>/QIBM/ProdData/mqm/java/lib/jca</code>
Windows	<code>MQ_INSTALLATION_PATH\java\lib\jca</code>
z/OS	<code>MQ_INSTALLATION_PATH/java/lib/jca</code>

`MQ_INSTALLATION_PATH`, IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder.

Bir uygulama sunucusundan IBM MQ olanağına bağlanmak için IBM MQ kaynak bağdaştırıcısını kullanmanız gerekir. Kullanmakta olduğunuz uygulama sunucusuna bağlı olarak, kaynak bağdaştırıcısı önceden kurulmuş olabilir ya da kendiniz kurmanız gerekebilir.

<i>Çizelge 61. Kaynak bağdaştırıcısının bir uygulama sunucusuna kurulması</i>	
<b>Uygulama sunucusu</b>	<b>Önceden kurulmuş ya da kuruluş gerekli mi?</b>
WebSphere Application Server traditional 9.0	IBM MQ 9.0 kaynak bağdaştırıcısı, WebSphere Application Server traditional 9.0 içinde önceden kurulmuş olmalıdır. Bu nedenle, WebSphere Application Server traditional 9.0 içine yeni bir kaynak bağdaştırıcısı kurmanıza gerek yoktur.
WebSphere Liberty	WebSphere Liberty, IBM MQ kaynak bağdaştırıcısını içermez; bu nedenle bunu Fix Central' dan ayrı olarak edinmeniz gerekir.
Diğer Java EE uygulama sunucusu	Obtain the resource adapter separately from Fix Central, as for WebSphere Liberty.

## Yordam

- If you are connecting to IBM MQ from WebSphere Liberty, or another Java EE application server, download and install the IBM MQ resource adapter as described in [“Kaynak baędařtırıcısının Liberty’ine kurulması” sayfa 423.](#)



UNIX and Linux sistemlerindeki baę tanımları baęlantıları için, Java Native Interface (JNI) kitaplıklarını içeren dizinin sistem yolunda olduğundan emin olun.

IBM MQ classes for JMS kitaplıklarını da içeren bu dizinin yerini görmek için [“Java Native Interface \(JNI\) kitaplıklarının yapılandırılması” sayfa 86’ e bakın.](#)



Windows' ta bu dizin, IBM MQ classes for JMS kurulumu sırasında otomatik olarak sistem yoluna eklenir.

**İpucu:** Sistem yolunu ayarlamaya alternatif olarak, IBM MQ kaynak baędařtırıcısında JNI kitaplığının yerini belirtmek için kullanılabilir nativeLibraryPath adı verilir. Örneęin, WebSphere Liberty içinde, ařaęıdaki örnekte gösterildięi gibi yapılandırılacaktır:

```
<wmqJmsClient nativeLibraryPath="/opt/mqm/java/lib64"/>
```

İřlemler hem istemci, hem de baę tanımları kipinde desteklenir.

## Kaynak baędařtırıcısının Liberty’ine kurulması

IBM MQ 'dan WebSphere Liberty' a ya da dięer Java EE uygulama sunucularından baęlanmak için IBM MQ kaynak baędařtırıcısını kullanmanız gerekir. Liberty , IBM MQ kaynak baędařtırıcısını içermedięinden, bunu Fix Central' dan ayrı olarak edinmeniz gerekir.

## Başlamadan önce

**Not:** Bu konudaki bilgiler WebSphere Application Server traditional 9.0 için geçerli deęildir. IBM MQ 9.0 kaynak baędařtırıcısı, WebSphere Application Server traditional 9.0 içinde önceden kuruludur. Bu nedenle, bu durumda yeni bir kaynak baędařtırıcısı takmaya gerek yoktur.

Bu göreve başlamadan önce, makineniz üzerinde kurulu bir Java runtime environment (JRE) kurulu olduğundan ve JRE ' nin sistem yoluna eklendięinden emin olun.

Bu kuruluş işleminde kullanılan Java kuruluş programı, kök (root) ya da belirli bir kullanıcı olarak çalışmamaya gerek yoktur. Tek gereksinim, çalıştırıldığı kullanıcının, dosyaların içeri girmesini istedięiniz dizine yazma erişimi olduğu gibi çalıştırılır.

## Bu görev hakkında

Fix Central olanaęından yükleyebileceğiniz kaynak baędařtırıcısına ilişkin JAR dosyası yürütülebilir. Bu yürütülebilir dosyayı çalıştırdığınızda, bu dosya, kabul edilmesi gereken IBM MQ lisans sözleşmesini görüntüler. Bu, IBM MQ kaynak baędařtırıcısının kurulacaęı bir dizin ister. Daha sonra bu dizine kaynak baędařtırıcısı RAR dosyası ve kuruluş doęrulama sınaması (IVT) programı kurulur. Varsayılan deęeri kabul edebilir ya da başka bir dizin belirtebilirsiniz; bu dizin, bir uygulama sunucusunun kaynak baędařtırıcılı dizini ya da sisteminizdeki herhangi bir dizin olabilir. Bu dizin yoksa, kuruluş programı bir parçası olarak yaratılır.

IBM MQ 9.0 öncesinde, karřıdan yüklenecek dosyanın adı *V.R.M.F-WS-MQ-Java-InstallRA.jar* biçiminde (örneęin, *8.0.0.6-WS-MQ-Java-InstallRA.jar*) biçimdeydi. IBM MQ 9.0' tan, dosya adının biçimi *V.R.M.F-IBM-MQ-Java-InstallRA.jar* biçimidir (örneęin, *9.0.0.0-IBM-MQ-Java-InstallRA.jar*).

Kaynak baędařtırıcısını karřıdan yükleyip kurduktan sonra, WebSphere Liberty' ta yapılandırmayı hazırlamaya hazır olun.

## Yordam

1. IBM MQ kaynak bağıdaştırıcısını Fix Centralolanağından yükleyin.

- Bu bağlantıyı tıklatın: [IBM MQ Resource Adapter](#).
- Görüntülenen kullanılabilir düzeltmeler listesinde, IBM MQ sürümünüze ilişkin kaynak bağıdaştırıcısını 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 bağıdaştırıcısı dosya adını tıklatın ve karşıdan yükleme işlemi izleyin.

2. Kuruluşu başlatmak için, dosyayı karşıdan yüklediğiniz dizinden aşağıdaki komutu girin.

IBM MQ 9.0' tan komutun biçimi aşağıdaki gibidir:

```
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 Central' den aşağı yüklenen dosyanın adıdır.

For example, to install the IBM MQ resource adapter for the version 9.1.4.0 release, you would use the following command:

```
java -jar 9.1.4.0-IBM-MQ-Java-InstallRA.jar
```

**Not:** Bu kuruluşu gerçekleştirmek için, makineniz üzerinde kurulu bir JRE kurulu olmalıdır ve sistem yoluna eklenmelidir.

Komutu girdiğinizde, aşağıdaki bilgiler görüntülenir:

IBM MQ V9.1' i kullanmadan, çıkarmadan ya da kurmadan önce kabul etmeniz gerekir  
1. terimler. IBM International License Agreement for Evaluation of  
Programlar 2. IBM Uluslararası Program Lisans Sözleşmesi ve ek  
lisans bilgileri. Lütfen aşağıdaki lisans sözleşmelerini dikkatli bir şekilde okuyun.

Lisans sözleşmesi ayrı ayrı görüntülenebilir  
--viewLicenseSözleşme seçeneği.

Şimdi lisans koşullarını görüntülemek için Enter tuşuna ya da atlamak için 'x' tuşuna basın.

3. Lisans koşullarını gözden geçirin ve kabul edin:

- Lisansı görüntülemek için Enter tuşuna basın.

Diğer bir seçenek olarak, x tuşuna basılması lisansın görüntülenmesini sağlar.

Lisans görüntüledikten sonra ya da x seçeneğini belirledikten hemen sonra, ek lisans koşullarını görüntülemeyi seçebileceğiniz için aşağıdaki ileti görüntülenir:

Ek lisans bilgileri ayrı ayrı görüntülenebilir  
--viewLicenseBilgi seçeneği.

Ek lisans bilgilerini şimdi görüntülemek için Enter tuşuna ya da atlamak için 'x' tuşuna basın.

- Ek lisans koşullarını görüntülemek için Enter tuşuna basın.

Diğer bir seçenek olarak, x tuşuna basılması ek lisans koşullarının görüntülenmesini sağlar.

Ek lisans koşullarının görüntülenmesinin ardından ya da x seçildikten hemen sonra, lisans sözleşmesini kabul etmenizi isteyen aşağıdaki ileti görüntülenir:

Aşağıdaki "I Agree" (Kabul ediyorum) seçeneğini belirleyerek,  
lisans sözleşmesi veIBM dışı koşullar (varsa). Eğer yapmazsanız  
Katılıyorum, "Kabul etmiyorum" seçeneğini belirleyin.

[ 1] Kabul ediyorum seçeneğini belirleyin ya da [ 2] Kabul etmiyorum:

- Lisans sözleşmesini kabul etmek ve kuruluş dizinini seçmeye devam etmek için 1 'i seçin.

Diğer bir seçenek olarak, 2 'yi seçerseniz kuruluş hemen sonlandırılır.

1 'i seçtiyseniz, hedef kuruluş dizinini 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 olur.  
Ürün dosyaları için hedef dizin?

#### 4. Kaynak bağdaştırıcısına ilişkin kuruluş dizinini belirtin:

- Kaynak bağdaştırıcısını varsayılan konuma kurmak istiyorsanız, bir değer belirtmeden Enter tuşuna basın.
- Kaynak bağdaştırıcısını varsayılan değer olarak farklı bir yere kurmak istiyorsanız, kaynak bağdaştırıcısını kurmak istediğiniz dizinin adını belirleyin ve sonra Enter tuşuna basın.

Dosyalar seçilen yere kurulduktan sonra, aşağıdaki örnekte gösterildiği gibi bir onay iletisi görüntülenir:

```
Dosyalar H ' ye çıkarılıyor: \Liberty\WMQ \wmq  
Tüm ürün dosyaları başarıyla çıkarıldı.
```

Kuruluş sırasında, seçilen kuruluş dizini içinde wmq adını taşıyan yeni bir dizin yaratılır ve aşağıdaki dosyalar wmq dizinine kurulur:

- Kurulum doğrulama test programı, wmq.jmsra.ivt.
- IBM MQ RAR dosyası, wmq.jmsra.rar.

#### 5. Configure the resource adapter in WebSphere Liberty.

Kaynak bağdaştırıcısını Liberty ' ta yapılandırmak için yapmanız gereken adımlar aşağıdaki gibidir. Daha fazla bilgi için, [WebSphere Application Server ürün belgelerine](#) bakın.

- a) IBM MQ 9.1 kaynak bağdaştırıcısıyla çalışmaya izin vermek için, wmqJmsClient-2.0 özelliğini server.xml dosyasına ekleyin.

Daha fazla bilgi için bkz “Kullanılacak kaynak bağdaştırıcısının hangi sürümü” sayfa 415.

- b) Kurmuş olduğunuz wmq.jmsra.rar dosyasına bir başvuru ekleyin.

**Not:** For Liberty versions up to WebSphere Liberty 8.5.5 Fix Pack 1, if an EJB is deployed using solely the configuration within the ejb-jar.xml, the version of WebSphere Application Server that the Liberty Profile is using must have APAR PM89890 applied. Bu yapılandırma yöntemi, kaynak bağdaştırıcısının kuruluş doğrulama programı (IVT) için kullanılır; bu nedenle, IVT ' nin çalışması için bu APAR gereklidir.

Sunucu uygulamalarını ve MDB'lerini desteklemek için bir örnek yapılandırma, JNDI ile şu şekilde 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"/>
```

## 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. Kaynak bağdaştırıcısını, kuruluş doğrulama sınavı (IVT) programını çalıştırabilmek için de yapılandırmanız gerekir. Bu, IBM hizmetinin, IBM dışı uygulama sunucusunun doğru bir şekilde yapılandırıldığını belirtmek için bu programın çalıştırılmasını gerektirebileceğinden önemlidir.

### Başlamadan önce

Bu görev, JMS ve IBM MQ classes for JMS ile önceden bilgi sahibi olduğunuzu varsayar. IBM MQ kaynak bağdaştırıcısını yapılandırmak için kullanılan özelliklerin birçoğu, IBM MQ classes for JMS nesnelerinin özelliklerine eşdeğerdir ve aynı işleve sahiptir.

## Bu görev hakkında

Her uygulama sunucusu, kendi denetim arabirimleri kümesini sağlar. Bazı uygulama sunucuları, JCA kaynaklarını tanımlamak için grafik kullanıcı arabirimleri sağlar, ancak diğer kullanıcılar denetimcinin XML konuşlandırma planlarını yazmasını gerektirir. Bu nedenle, her uygulama sunucusu için IBM MQ kaynak bağdaştırıcısının nasıl yapılandırılacağı hakkında bilgi sağlamak için bu belge kapsamı dışındadır.

Bu nedenle, yalnızca konfigürasyonunu tanımlamanız gerekenin üzerine odaklanıyorsanız, aşağıdaki adımlar kullanılabilir. Bir JCA kaynak bağdaştırıcısının nasıl yapılandırılabilmesiyle ilgili bilgi için uygulama sunucunuzla birlikte sağlanan belgelere bakın.

## Yordam

JCA kaynaklarını aşağıdaki kategorilerde tanımlayın:

- ResourceAdapter nesnesinin özelliklerini tanımlayın.  
Tanılama izleme düzeyi gibi kaynak bağdaştırıcısının genel özelliklerini temsil eden bu özellikler, [“ResourceAdapter nesne özellikleri yapılandırması” sayfa 427](#) içinde açıklanmıştır.
- Bir ActivationSpec nesnesine ilişkin özellikleri tanımlayın.  
Bu özellikler, gelen iletişim için bir MDB 'nin nasıl etkinleştirildiğini belirler. Daha fazla bilgi için bkz [“Kaynak bağdaştırıcısının gelen iletişim için yapılandırılması” sayfa 429](#).
- Bir ConnectionFactory nesnesine ilişkin özellikleri tanımlayın.  
Uygulama sunucusu, giden iletişim için bir JMS ConnectionFactory nesnesi yaratmak için bu özellikleri kullanır. Daha fazla bilgi için bkz [“Kaynak bağdaştırıcısının giden iletişim için yapılandırılması” sayfa 445](#).
- Yönetilen hedef nesnenin özelliklerini tanımlayın.  
Uygulama sunucusu, giden iletişim için bir JMS Kuyruk nesnesi ya da JMS Konu nesnesi yaratmak için bu özellikleri kullanır. Daha fazla bilgi için bkz [“Kaynak bağdaştırıcısının giden iletişim için yapılandırılması” sayfa 445](#).
- İsteğe bağlı: Kaynak bağdaştırıcısı için bir konuşlandırma planı tanımlayın.  
IBM MQ kaynak bağdaştırıcısı RAR dosyası, kaynak bağdaştırıcısına ilişkin konuşlandırma tanımlayıcısını içeren META-INF/rar.xml adlı bir dosya içerir. Bu konuşlandırma tanımlayıcısı, [https://xmlns.jcp.org/xml/ns/javaee/connector\\_1\\_7.xsd](https://xmlns.jcp.org/xml/ns/javaee/connector_1_7.xsd) konumundaki XML şemasıyla tanımlanır ve kaynak bağdaştırıcısına ve sağladığı hizmetlere ilişkin bilgileri içerir. Bir uygulama sunucusu, kaynak bağdaştırıcısı için bir konuşlandırma planı da gerektirebilir. Bu devreye alma planı uygulama sunucusuna özgüdür.

JVM sistem özelliklerini gereken şekilde belirtin:

- TLS (Transport Layer Security; İletim Katmanı Güvenliği) kullanıyorsanız, aşağıdaki örnekte olduğu gibi, anahtar deposu dosyası ve güvenilirlik deposu dosyasının JVM sistem özellikleri olarak yerlerini 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 fazla anahtar deposu belirtemezsiniz. Özellikler tüm JVM için geçerlidir ve uygulama sunucusunda çalışmakta olan diğer uygulamalar TLS bağlantıları kullanıyorsa uygulama sunucusunu etkileyebilir. Uygulama sunucusu, bu özellikleri farklı değerlere de döndürebilir. TLS 'nin IBM MQ classes for JMS ile kullanılmasına ilişkin daha fazla bilgi için bkz. [“Using TLS with IBM MQ classes for JMS” sayfa 225](#).

- İsteğe bağlı: Gerekliyse, uygulama sunucunuzun standart çıkış günlüğüne uyarı iletilerini günlüğe kaydetmek için kaynak bağdaştırıcısını yapılandırın.  
Kaynak bağdaştırıcısı günlükleri, uyarı ve hata iletileri, IBM MQ classes for JMS ile aynı mekanizmayı kullanır. Daha fazla bilgi için bkz. [IBM MQ classes for JMS için günlüğe kaydetme hataları](#). Bu,

varsayılan olarak, iletilerin mqjms.log adlı bir dosyaya gitmesiyle ilgili bir ifade anlamına gelir. Kaynak bağıdaştırıcısını, uygulama sunucunuzun standart çıkış günlüğüne uyarı iletileri olarak kaydetmek üzere 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 JMS izleme izini denetlemek için kullanılan özelliklerle aynıdır. IBM MQ classes for JMS ile olduğu gibi, jms.config dosyasını gösteren bir sistem özelliğini kullanmak (bkz. "IBM MQ classes for JMS yapılandırma dosyası" sayfa 88) mümkündür. Bir JVM sistem özelliği ayarlamaya ilişkin bilgi edinmek için uygulama sunucusu belgelerinize bakın.

Kuruluş doğrulama sınavmasını çalıştırmak için kaynak bağıdaştırıcısını yapılandırın

- Kaynak bağıdaştırıcısını, IBM MQ kaynak bağıdaştırıcısıyla birlikte sağlanan kuruluş doğrulama sınavması (IVT) programını çalıştırmak için yapılandırın.

IVT programını çalıştırmak için yapılandırmanız gerekenlerle ilgili bilgi için "[Kaynak bağıdaştırıcısı kuruluşunun doğrulanması](#)" sayfa 464' e bakın.

Bu, IBM hizmetinin, IBM dışı uygulama sunucusunun doğru bir şekilde yapılandırıldığını belirtmek için bu programın çalıştırılmasını gerektirebileceğinden önemlidir.

**Önemli:** Programı çalıştırabilmek için önce kaynak bağıdaştırıcısını yapılandırmalısınız.

### **ResourceAdapter nesne özellikleri yapılandırması**

ResourceAdapter nesnesi, tanılama izleme düzeyi gibi, IBM MQ kaynak bağıdaştırıcısının genel özelliklerini sarsar. Bu özellikleri tanımlamak için, uygulama sunucunuzla birlikte sağlanan belgelerde açıklandığı gibi, kaynak bağıdaştırıcınızla ilgili olanakları kullanın.

ResourceAdapter nesnesi için iki özellik kümesi vardır:

- Tanılama izlemesi ile ilişkili özellikler
- Kaynak bağıdaştırıcısı tarafından yönetilen bağlantı havuzuyla ilişkili özellikler


Bu özellikleri tanımlamanızın yolu, uygulama sunucunuzun sağladığı denetim arabirimlerine bağlıdır. If you are using WebSphere Application Server traditional, see "[WebSphere Application Server traditional yapılandırması](#)" sayfa 429 or if you are using WebSphere Liberty, see "[WebSphere Liberty yapılandırması](#)" sayfa 429. Diğer uygulama sunucuları için, uygulama sunucunuza ilişkin ürün belgelerine bakın.

Tanılama izlemesiyle ilişkili özelliklerin tanımlamaya ilişkin ek bilgi için [IBM MQ Kaynak Bağıdaştırıcısının İzlenmesi](#) başlıklı konuya bakın.

The resource adapter manages an internal connection pool of JMS connections that are used to deliver messages to MDBs. [Çizelge 62 sayfa 427](#) , bağlantı havuzuyla ilişkilendirilmiş ResourceAdapter nesnesinin özelliklerini listeler.

Özellik adı	Tip	Varsayılan değer	Tanım
maxConnections	Dizgi	50	Bir IBM MQ kuyruk yöneticisine bağlantı sayısı üst sınırı ve konuşlandırılan MDBs sayısı üst sınırı.
connectionConcurrency	Dizgi	1	Bir JMS bağlantısını paylaşmak için en çok MDBs ' nin sayısı. Paylaşım bağlantıları olanaklı değil ve bu özellik her zaman 1 değerine sahip olur.
reconnectionRetrySayı	Dizgi	5	Bir bağlantı başarısız olursa, kaynak bağıdaştırıcısı tarafından bir IBM MQ kuyruk yöneticisine yeniden bağlanmak için yapılan deneme sayısı üst sınırı.

Çizelge 62. Bağlantı havuzuyla ilişkili ResourceAdapter nesnesinin özellikleri (devamı var)

Özelliğin adı	Tip	Varsayılan değer	Tanım
reconnectionRetryInterval	Dizgi	300 000	Kaynak bağdaştırıcısının bir IBM MQ kuyruk yöneticisine yeniden bağlanmayı denemeden önce bekleyeceği süre (milisaniye olarak).
startupRetrySayısı	Dizgi	0	Uygulama sunucusu başlatıldığında kuyruk yöneticisi çalışmıyorsa, başlatma sırasında bir MDB ' yi çalıştırmaya ve bağlamaya ilişkin varsayılan değer sayısı.
startupRetryAralığı	Dizgi	30 000	Başlatma bağlantısı girişimleri arasındaki varsayılan uyku süresi (milisaniye cinsinden).
supportMQExtensions	Dizgi	yanlış	IBM MQ JMS davranışını JMS 2.0 öncesi davranışına geri çevirir. Daha fazla bilgi için bkz "SupportMQExtensions özelliği" sayfa 304.
nativeLibraryYolu	Dizgi	<empty>	Bağ tanımları kipi bağlantılarına izin vermek için IBM MQ JNI kitaplığını yüklemek için kullanılacak yol.  Windows üzerinde, sistem yolunda eşleşen IBM MQ kuruluşunun konumunu da içermesi gerekir.

Uygulama sunucusunda bir MDB konuşlandırıldığında, yeni bir JMS bağlantısı yaratılır ve maxConnection özelliği tarafından belirtilen bağlantı sayısı üst sınırı aşılmadığında, kuyruk yöneticisiyle başlayan bir etkileşim başlatılır. Bu nedenle, MDSB sayısı üst sınırı bağlantı sayısı üst sınırına eşittir. Devreye alınan MDSB sayısı bu üst sınıra ulaşırsa, başka bir MDB ' yi devreye alma girişimi başarısız olur. Bir MDB durdurulduysa, bağlantısı başka bir MDB tarafından kullanılabilir.

Genel olarak, birçok MDB konuşlandırılacaksa, maxConnections özelliğinin değerini artırmanız gerekir.

The reconnectionRetryCount and reconnectionRetryInterval properties govern the behavior of the resource adapter when connections to an IBM MQ queue manager fail, because of a network failure for example. When a connection fails, the resource adapter suspends the delivery of messages to all MDSBs supplied by that connection for an interval specified by the reconnectionRetryInterval property. Bundan sonra, kaynak bağdaştırıcısı kuyruk yöneticisine yeniden bağlanmayı dener. Girişim başarısız olursa, kaynak bağdaştırıcısı, reconnectionRetryInterval özelliği tarafından belirtilen aralıklarla yeniden bağlanma girişiminde bulunulursa, reconnectionRetrySayı özelliği tarafından belirtilen sınıra ulaşıncaya kadar. Tüm denemeler başarısız olursa, MDSB ' ler el ile yeniden başlatılincaya kadar teslim kalıcı olarak durdurulur.

Genel olarak, ResourceAdapter nesnesi denetim gerektirmez. Ancak, UNIX and Linux sistemlerinde tanılama izlemenin geçerli kılınmasını sağlamak için aşağıdaki özellikleri ayarlayabilirsiniz:

```
traceEnabled: true
traceLevel: 10
```

Kaynak bağdaştırıcısı başlatılmamışsa, bu özellikler herhangi bir etkiye sahip değildir; örneğin, IBM MQ kaynaklarını kullanan uygulamalar yalnızca istemci kapsayıcısında çalıştırılıyorsa, bu durum bu özelliklerden etkilenmez. Bu durumda, tanılama izlemesi için özellikleri Java Virtual Machine (JVM) sistem

özellikleri olarak ayarlayabilirsiniz. Aşağıdaki örnekte gösterildiği gibi, özellikleri **java** komutundaki -D işareti kullanarak ayarlayabilirsiniz:

```
java ... -DtraceEnabled=true -DtraceLevel=6
```

ResourceAdapter nesnesinin tüm özelliklerini tanımlamanıza gerek yoktur. Belirtilmeyen tüm özellikler varsayılan değerlerini alır. Yönetilen bir ortamda, özellikleri belirtmenin iki yolunu karıştırmamak daha iyi olur. Bunları karıştırırsanız, JVM sistem özellikleri ResourceAdapter nesnesinin özelliklerine göre öncelik kazanır.

## WebSphere Application Server traditional yapılandırması

WebSphere Application Server traditionalindeki kaynak bağdaştırıcısı için aynı özellikler kullanılabilir, ancak bunlar kaynak bağdaştırıcısının özellikler panosu içinde ayarlanmalıdır ( WebSphere Application Server traditional ürün belgelerindeki [JMS sağlayıcısı ayarları](#) konusuna bakın). İzleme, WebSphere Application Server traditional yapısının tanımlama bölümü tarafından denetlenir. Ek bilgi için, WebSphere Application Server traditional ürün belgelerindeki [Diagnostic with Diagnostic Providers](#) başlıklı konuya bakın.

## WebSphere Liberty yapılandırması

Kaynak bağdaştırıcısı, aşağıdaki örnekte gösterildiği gibi server.xml dosyasında XML öğeleri kullanılarak yapılandırılır:

```
<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 etkinleştirildi:

```
<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, bir ileti sürücüsü 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 zamanuysuz olarak teslim eden bir JMS bağlantı tüketicisi oluşturmak için kullanılan özellikler

ActivationSpec nesnesinin özelliklerini tanımlama şekliniz, uygulama sunucunuz tarafından sağlanan denetim arabirimlerine bağlıdır.

## JMS 2.0 içinde yeni ActivationSpec 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 tanımlandığını ve bu nesnenin JNDI adının bir etkinleştirme belirtimine ilişkin connectionFactoryLookup özelliğinde belirtildiğini varsayın. JNDI içinde tanımlanan bağlantı üreticisinin tüm özellikleri, Çizelge 63 sayfa 430 içindeki özelliklerin tercihi olarak kullanılır.

JNDI içinde bir hedef tanımlıysa ve JNDI adı ActivationSpec' in destinationLookup özelliğinde ayarlandıysa, Çizelge 64 sayfa 439 içindeki değerlere tercih etmek için kullanılan değerler kullanılır. Bu iki özelliğin nasıl kullanıldığıyla ilgili daha fazla bilgi için bkz. "ActivationSpec connectionFactoryLookup ve destinationLookup özellikleri" sayfa 442.

## IBM MQ kuyruk yöneticisine JMS bağlantısı yaratmak için kullanılan özellikler

Çizelge 63 sayfa 430 içindeki tüm özellikler isteğe bağlıdır.

Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
applicationName	Dizgi	<ul style="list-style-type: none"> <li>Çağırın sınıf adı (varsa), 28 karakterden uzun olmayacak şekilde ayarlanır. Yoksa, WebSphere MQ Client for Java dizgisi kullanılır.</li> </ul>	Bir uygulamanın kuyruk yöneticisine kayıtlı olduğu ad. Bu uygulama adı <b>DISPLAY CONN MQSC/PCF</b> komutuyla gösterilir (burada alan <b>APPLTAG</b> olarak adlandırılır) ya da IBM MQ Explorer <b>Uygulama Bağlantıları</b> ekranında (alanın <b>App name</b> olarak adlandırıldığı yerde).
brokerCCDurSubQueue <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li><b>SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE</b></li> <li>Kuyruk adı</li> </ul>	Bağlantı tüketicisinin sürekli abonelik iletileri aldığı kuyruğun adı
brokerCCSubKuyruk <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li><b>SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE</b></li> <li>Kuyruk adı</li> </ul>	Bağlantı tüketicisinin sürekli olmayan abonelik iletilerini aldığı kuyruğun adı
brokerControlKuyruk <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li><b>SYSTEM.BROKER.CONTROL.QUEUE</b></li> <li>Kuyruk adı</li> </ul>	Aracı denetim kuyruğunun adı
brokerQueueManager <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li><b>"" (boş dizgi)</b></li> <li>Kuyruk yöneticisi adı</li> </ul>	Aracının çalıştığı kuyruk yöneticisinin adı
brokerSubKuyruk <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li><b>SYSTEM.JMS.ND.SUBSCRIBER.QUEUE</b></li> <li>Kuyruk adı</li> </ul>	Sürekli olmayan bir ileti tüketicisinin iletileri aldığı kuyruğun adı

Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
brokerVersion <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>belirlenmedi</b> -Aracı V6 'dan V7' ye geçirildikten sonra, bu özelliği RFH2 üstbilgilerinin artık kullanılmayacak şekilde ayarlayın. Geçişten sonra bu özellik artık ilgili değildir.</li> <li>• <b>V1</b> -Bir IBM MQ yayınlama/abone olma broker.This değeri varsayılan değerdir.</li> <li>• <b>V2</b> -Yerel kipte IBM Integration Bus aracısını kullanmak için. TRANSPORT değeri DIRECT ya da DIRECTHTTP olarak ayarlandıysa, bu değer varsayılan değerdir.</li> </ul>	Kullanılmakta olan aracının sürümü
ccdtURL	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Birörnek kaynak yeri belirleyici (URL)</li> </ul>	İstemci kanal tanımlama çizelgesini (CCDT) içeren dosyanın adını ve yerini tanımlayan ve dosyaya nasıl erişilebileceğini belirten URL
CCSID	Dizgi	<ul style="list-style-type: none"> <li>• <b>819</b></li> <li>• Java sanal makinesi (JVM) tarafından desteklenen bir kodlanmış karakter takımı tanıtıcısı</li> </ul>	Bağlantıya ilişkin kodlanmış karakter takımı tanıtıcısı
kanal	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.DEF.SVRCONN</b></li> <li>• MQI kanalının adı</li> </ul>	Kullanılacak MQI kanalının adı
cleanupInterval <sup>1</sup> (Temizleme Aralığı)	int	<ul style="list-style-type: none"> <li>• <b>3600 000</b></li> <li>• Artı bir tamsayı</li> </ul>	Yayınlama/abone olma temizleme yardımcı programının arka plan çalıştırmaları arasındaki milisaniye cinsinden aralık
cleanupLevel <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>GÜVENLİ</b></li> <li>• YOK</li> <li>• güçlü</li> <li>• ZORLA</li> <li>• NONDUR</li> </ul>	Aracıya dayalı abonelik deposu için temizleme düzeyi
clientID	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Bir istemci tanıtıcısı</li> </ul>	Bağlantıya ilişkin istemci tanıtıcısı
cloneSupport	Dizgi	<ul style="list-style-type: none"> <li>• <b>DEVRE Dışı</b> -Bir kerede tek bir sürekli konu abonesi çalıştırılabilir.</li> <li>• <b>ENABLED</b>-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.</li> </ul>	Aynı sürekli konu abonesinin iki ya da daha fazla eşgörünümünün aynı anda çalışıp çalışmayacağını belirler

Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
connectionFactoryArama	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Bir ConnectionFactory nesnesine ilişkin JNDI adı</li> </ul>	<p>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 "<a href="#">ActivationSpec connectionFactoryLookup ve destinationLookup özellikleri</a>" sayfa 442.</p>
connectionNameListesi	Dizgi	<ul style="list-style-type: none"> <li>• <b>localhost (1414)</b></li> <li>• Her öğenin biçimi aldığı, virgüllerle ayrılmış öğelerden oluşan bir dizgi:</li> </ul> <div style="background-color: #e0e0e0; padding: 5px; margin: 10px 0;"> <p><i>HOSTNAME (PORT)</i></p> </div> <p>Burada <i>HOSTNAME</i> , bir DNS adı ya da bir IP adresidir.</p>	<p>Gelen iletişim için kullanılan TCP/IP bağlantı adlarının listesi.</p> <p>Belirtildiğinde, <b>connectionNameList hostname</b> ve <b>port</b> özelliklerinin yerini alır.</p> <p>Bu özellik, çok eşgörünümlü kuyruk yöneticilerine yeniden bağlanmak için kullanılır.</p> <p><b>connectionNameList</b> , formda <b>localAddress</b> ile benzer, ancak bununla karıştırılmamalıdır. <b>localAddress</b> , yerel iletişimin özelliklerini belirtirken, <b>connectionNameList</b> uzak kuyruk yöneticisine nasıl erişileceğini belirtir.</p>
failIfSusturma	Boole	<ul style="list-style-type: none"> <li>• <b>doğru</b></li> <li>• yanlış</li> </ul>	<p>Kuyruk yöneticisi susturma durumundaysa, belirli yöntemlere yönelik çağrılarının başarısız olup olmayacağını belirler</p>
headerCompression	Dizgi	<ul style="list-style-type: none"> <li>• <b>YOK</b></li> <li>• SYSTEM-RLE ileti üstbilgisi sıkıştırması gerçekleştirilir</li> </ul>	<p>Bir bağlantıdaki üstbilgi verilerini sıkıştırmak için kullanılacak tekniklerin listesi</p>



Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
hostName	Dizgi	<ul style="list-style-type: none"> <li>• <b>localhost</b></li> <li>• Anasistem adı</li> <li>• Bir IP adresi</li> </ul>	<p>Kuyruk yöneticisinin bulunduğu sistemin anasistem adı ya da IP adresi.</p> <p><b>hostname</b> ve <b>port</b> özellikleri, belirtildiğinde <b>connectionNameList</b> özelliği tarafından değiştirilir.</p>
localAddress	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Şu biçimde bir dizgi:</li> </ul> <pre>[ host_name ][(low_port [, high_port ])]</pre> <p>Burada <i>anasistem_adi</i> bir anasistem adı ya da IP adresi, <i>alt_kapı</i> ve <i>yüksek_kapı</i> TCP kapı numaralarıdır ve köşeli ayraçlar isteğe bağlı bir bileşeni gösterir</p>	<p>Bir kuyruk yöneticisine yönelik bağlantı için, bu özellik aşağıdakilerden birini ya da her ikisini de belirtir:</p> <ul style="list-style-type: none"> <li>• Kullanılacak yerel ağ arabirimi</li> <li>• Kullanılacak yerel kapı ya da yerel kapı aralığı</li> </ul> <p><b>localAddress</b> , formda <b>connectionNameList</b> ile benzer, ancak bununla karıştırılmamalıdır. <b>localAddress</b> , yerel iletişimin özelliklerini belirtirken, <b>connectionNameList</b> uzak kuyruk yöneticisine nasıl erişileceğini belirtir.</p>
messageCompression	Dizgi	<ul style="list-style-type: none"> <li>• <b>YOK</b></li> <li>• Boş karakterlerle ayrılmış olarak aşağıdaki değerlerden birinin ya da daha fazlasının listesi:</li> </ul> <p>RLE ZLIBFAST ZLIBHIGH</p>	<p>Bir bağlantıdaki ileti verilerini sıkıştırma tekniklerinin listesi</p>
messageRetention <sup>1</sup>	Boole	<ul style="list-style-type: none"> <li>• <b>true</b> -İstenmeyen iletiler giriş kuyruğunda kalır</li> <li>• <b>false</b>-İstenmeyen iletiler, yok etme seçeneklerine göre ele alınmaz</li> </ul>	<p>Bağlantı tüketicisinin giriş kuyruğunda istenmeyen iletileri alıkoyup saklamadığını belirler</p>
messageSelection <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>İstemci</b></li> <li>• Aracı</li> </ul>	<p>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.</p>

Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
parola	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Parola</li> </ul>	Kuyruk yöneticisine bağlantı yaratılırken kullanılacak varsayılan parola
pollingInterval <sup>1</sup>	int	<ul style="list-style-type: none"> <li>• <b>5000</b></li> <li>• Pozitif bir tamsayı</li> </ul>	Bir oturumdaki her ileti dinleyicinin kuyruğunda uygun bir ileti yoksa, bu değer, her ileti dinleyicisinin kuyruğundan bir ileti almaya 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	<ul style="list-style-type: none"> <li>• <b>1414</b></li> <li>• TCP kapı numarası</li> </ul>	Kuyruk yöneticisinin dinlediği kapı. <b>hostname</b> ve <b>port</b> özellikleri, belirtildiğinde <b>connectionNameList</b> özelliği tarafından değiştirilir.
providerVersion	dizgi	<ul style="list-style-type: none"> <li>• <b>belirtilmedi</b></li> <li>• Aşağıdaki biçimlerden birindeki bir dizgi <ul style="list-style-type: none"> <li>– V.R.M.F</li> <li>– V.R.M</li> <li>– V.R</li> <li>– V</li> </ul> </li> </ul> <p>Burada V, R, M ve F sıfırdan büyük ya da sıfıra eşit tamsayı değerleridir.</p>	MDB 'nin bağlanmak istediği kuyruk yöneticisinin sürümü, yayını, değişiklik düzeyi ve düzeltme paketi.
queueManager	Dizgi	<ul style="list-style-type: none"> <li>• "" (<b>boş dizgi</b>)</li> <li>• Kuyruk yöneticisi adı</li> </ul>	Bağlanılacak kuyruk yöneticisinin adı
receiveExit <sup>3</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• 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).</li> </ul>	Bir kanal alma çıkış programını ya da art arda çalıştırılacak bir alma çıkış programı sırasını tanımlar

Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğın adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
receiveExitBaşlatma	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Virgüllerle ayrılmış bir ya da daha fazla kullanıcı verisi ögesini içeren bir dizgi</li> </ul>	Kanala geçirilen kullanıcı verileri, çağrıldığında çıkış programlarını alır
rescanInterval <sup>1</sup>	int	<ul style="list-style-type: none"> <li>• <b>5000</b></li> <li>• Pozitif bir tamsayı</li> </ul>	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 <b>MsgDeliverySequence</b> özniteliği tarafından belirlenen sırada uygun iletileri arar. IBM MQ classes for JMS uygun bir ileti bulunduğunda 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 ulaşınca kadar ya da bu özelliğın 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 <sup>3</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• IBM MQ classes for Java arabirimini gerçekleştiren bir sınıfın tam olarak nitelenmiş adı, MQSecurityExit</li> </ul>	Kanal güvenliği çıkış programını tanımlar
securityExitBaşlatma	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Kullanıcı verileri dizgisi</li> </ul>	Çağrıldığında bir kanal güvenliği çıkış programına geçirilen kullanıcı verileri
sendExit <sup>3</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• 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 ).</li> </ul>	Kanal gönderme çıkış programını ya da art arda çalıştırılacak çıkış gönderme programlarını tanımlar


Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
sendExitInit	Dizgi	<ul style="list-style-type: none"> <li><b>boş değerli</b></li> <li>Virgüllerle ayrılmış bir ya da daha fazla kullanıcı verisi ögesini içeren bir dizgi</li> </ul>	Çağrıldığında çıkış programlarını kanala gönderen kullanıcı verileri
shareConvİzin Verilir	Boole	<ul style="list-style-type: none"> <li>NO-Bir istemci bağlantısı yuvasını paylaşmıyor.</li> <li><b>YES (EVET)</b>-Bir istemci bağlantısı yuvasını paylaşabilir.</li> </ul>	Bir istemci bağlantısının, kanal tanımlamaları eşleşiyorsa, aynı işlemde aynı kuyruk yöneticisine diğer üst düzey JMS bağlantılarıyla yuvasını paylaşıp paylaşamayacağını belirler.
sparseSubscriptions <sup>1</sup>	Boole	<ul style="list-style-type: none"> <li><b>false</b> -Abonelikler sık sık eşleşen iletiler alır.</li> <li><b>true</b>-Abonelikler nadiren eşleşen iletiler alır. Bu değer, abonelik kuyruğunun göz atmak için açılabilmesini gerektirir.</li> </ul>	TopicSubscriber nesnesinin ileti alma ilkesini denetler
sslCertMağazaları	Dizgi	<ul style="list-style-type: none"> <li><b>boş değerli</b></li> <li>Boşluklarla ayrılmış bir ya da daha çok LDAP URL dizesi. Her LDAP URL şu biçimdedir:  <pre>ldap://host_name [: port ]</pre>                     Burada <i>anasistem_adi</i> bir anasistem adı ya da IP adresidir, <i>kayı</i> bir TCP kayı numarasıdır ve köşeli parantezler isteğe bağlı bir bileşeni gösterir.                 </li> </ul>	TLS bağlantısında kullanılmak üzere sertifika iptal listelerini (CRL) bulunduran LDAP sunucuları
sslCipherÜrün Grubu	Dizgi	<ul style="list-style-type: none"> <li><b>boş değerli</b></li> <li>CipherSuite adı</li> </ul>	TLS bağlantısı için kullanılacak CipherSuite
sslFipsGerekli <sup>2</sup>	Boole	<ul style="list-style-type: none"> <li><b>yanlış</b></li> <li>doğru</li> </ul>	TLS bağlantısının IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) tarafından desteklenen bir CipherSuite kullanması gerekip gerekmediği
sslPeerAdı	Dizgi	<ul style="list-style-type: none"> <li><b>boş değerli</b></li> <li>Ayırt edici adlar için şablon</li> </ul>	TLS bağlantısı için, kuyruk yöneticisi tarafından sağlanan sayısal sertifikada ayırt edici adı denetlemek için kullanılan bir şablon
sslResetSayısı	int	<ul style="list-style-type: none"> <li><b>ERROR! SEGMENT DATA CORRUPTED, SEGDATA=0</b></li> <li>0-999 999 999 aralığında bir tamsayı</li> </ul>	TLS tarafından kullanılan gizli anahtarlar yeniden anlaşılmadan önce bir TLS bağlantısı tarafından gönderilen ve alınan toplam bayt sayısı

Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellğin 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 araç içine alınmış bir bağımsız değişken de içerilir.	Yönetilen nesne kapsamında kurulan tüm bağlantılar, SSLSocketFactory arabiriminin bu somutlamasından elde edilen yuvaları kullanır.
statusRefreshInterval <sup>1</sup> (Durum Yenileme)	int	<ul style="list-style-type: none"><li>• <b>60000</b></li><li>• Pozitif bir tamsayı</li></ul>	Bir abonenin kuyruk yöneticisine olan bağlantısını kaybettiği zaman algılanan, uzun süren hareketin yenilenmesi arasındaki milisaniye cinsinden aralık. Bu özellik yalnızca <b>subscriptionStore</b> QUEUEdeğerine sahipse geçerlidir.
subscriptionStore <sup>1</sup>	Dizgi	<ul style="list-style-type: none"><li>• <b>Aracı</b></li><li>• GEÇİŞ YAPIN</li><li>• kuyruk</li></ul>	IBM MQ classes for JMS etkin aboneliklerle ilgili kalıcı verilerin nerede saklanacağını belirler

Çizelge 63. JMS bağlantısı yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
transportType	Dizgi	<ul style="list-style-type: none"> <li>• <b>İstemci</b></li> <li>• Bağ Tanımları</li> <li>• BINDINGS_THEN_CLIENT</li> </ul>	<p>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.</p> <p> 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.</p>
kullanıcı adı	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Kullanıcı adı</li> </ul>	Bir kuyruk yöneticisine bağlantı yaratırken kullanılacak varsayılan kullanıcı adı
wildcardFormat	Dizgi	<ul style="list-style-type: none"> <li>• CHAR-Aracı sürüm 1 'de kullanıldığı şekilde, yalnızca genel arama karakterlerini tanır</li> <li>• <b>TOPIC</b> -Aracı sürüm 2 'de kullanıldığı şekilde, yalnızca konu düzeyi genel arama karakterlerini tanır.</li> </ul>	Genel arama karakteri sözdiziminin hangi sürümünün kullanılacağı

**Notlar:**

1. Bu özellik, IBM MQ classes for JMSürününün 7.0 sürümüyle birlikte kullanılabilir. **providerVersion** özelliği 7 'den küçük bir sürüm numarasına ayarlanmadıkça, IBM WebSphere MQ 7.0 kuyruk yöneticisine bağlı bir uygulamayı etkilemez.
2. sslFipsRequired özelliğini kullanma hakkında önemli bilgi için bkz. [“IBM MQ kaynak bağdaştırıcısına ilişkin sınırlamalar” sayfa 418.](#)

3. Kaynak bağdaştırıcısının bir çıkış bulabilmesi için nasıl yapılandırılacağı hakkında bilgi için bkz. [“Configuring IBM MQ classes for JMS to use channel exits”](#) sayfa 260.

## JMS bağlantı tüketicisi oluşturmak için kullanılan özellikler

**Not:** **destination** ve **destinationType** belirtik olarak tanımlanmalıdır. [Çizelge 64 sayfa 439](#) içindeki diğer tüm özellikler isteğe bağlıdır.

<i>Çizelge 64. JMS bağlantı tüketicisi yaratmak için kullanılan ActivationSpec nesnesinin özellikleri</i>			
Özellğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
Hedef	Dizgi	Hedef adı	İletilerin alınacağı hedef. <b>useJNDI</b> özelliği, bu özelliğin değerinin nasıl yorumlanacağını belirler.
destinationLookup	Dizgi	<ul style="list-style-type: none"><li>• <b>boş değerli</b></li><li>• Hedef nesneye ilişkin JNDI adı</li></ul>	Bu özellik ayarlanırsa, ActivationSpec , uygulama sunucusunun JNDI ad alanında belirtilen JNDI adına sahip bir JMS Hedef nesnesini arar ve daha sonra, ActivationSpec içinde belirtilen diğer özelliklere göre JMS bağlantı tüketicisi yaratmak için o nesnenin özelliklerini kullanır. Daha fazla bilgi için bkz <a href="#">“ActivationSpec connectionFactoryLookup ve destinationLookup özellikleri”</a> sayfa 442.
destinationType	Dizgi	<ul style="list-style-type: none"><li>• javax.jms.Queue</li><li>• javax.jms.Topic</li></ul>	Hedef, kuyruk ya da konu tipi
maxMessages	int	<ul style="list-style-type: none"><li>• <b>1</b></li><li>• Artı bir tamsayı</li></ul>	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	<ul style="list-style-type: none"><li>• <b>10</b></li><li>• Artı bir tamsayı</li></ul>	Bağlantı tüketicisi tarafından kullanılan sunucu oturumu havuzundaki sunucu oturumu sayısı üst sınırı
messageSelector	Dizgi	<ul style="list-style-type: none"><li>• <b>boş değerli</b></li><li>• SQL92 ileti seçici ifadesi</li></ul>	Teslim edilecek iletileri belirten bir ileti seçici ifadesi

Çizelge 64. JMS bağlantı tüketicisi yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
nonASFTimeout	int	<ul style="list-style-type: none"> <li>• <b>ERROR! SEGMENT DATA CORRUPTED, SEGDATA=0</b></li> <li>• Artı bir tamsayı</li> </ul>	<p>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.</p> <p>Bu parametre aşağıdaki durumda geçerlidir:</p> <ul style="list-style-type: none"> <li>• Uygulama, WebSphere Application Server 7.0 ya da daha sonraki bir yayın düzeylerinde çalışıyor.</li> <li>• Uygulama, wmqJmsClient özelliğinin uygun düzeyini kullanarak WebSphere Liberty içinde çalışıyor. Daha fazla bilgi için bkz "<a href="#">Liberty ve IBM MQ kaynak bağdaştırıcısı</a>" sayfa 420.</li> </ul>
nonASFRollbackEtkin	Boole	<ul style="list-style-type: none"> <li>• <b>false</b> -MDB başarısız olsa da ileti tüketilir.</li> <li>• true-MDB içindeki hata, iletinin kuyruğa geri dönmesine neden olur.</li> </ul>	<p>MDB işleminden geçmediyse, ileti tesliminin bir IBM MQ eşitleme noktası içinde olup olmadığını belirtir. MDB işlem yapıldıysa ya da <b>nonASFTimeout</b> , 0olarak ayarlandıysa yoksayılr.</p>
poolTimeout	int	<ul style="list-style-type: none"> <li>• <b>300000</b></li> <li>• Artı bir tamsayı</li> </ul>	<p>Kullanılmayan bir sunucu oturumunun, etkinlik dışı durum nedeniyle kapatılmadan önce sunucu oturumu havuzunda açık tutulduğu süre (milisaniye)</p>
readAheadİzin verilir	int	<ul style="list-style-type: none"> <li>• <b>DESTINATION</b> -Kuyruk ya da konu tanımlamasına başvurarak önden okumaya izin verilip verilmediğini belirleyin.</li> <li>• DISABLED-Önden okumaya izin verilmez.</li> <li>• ENABLED-Önden okumaya izin verilir.</li> <li>• QUEUE-Kuyruk tanımına başvurarak önden okumaya izin verilip verilmediğini belirleyin.</li> <li>• KONU-Konu tanımlamasına başvurarak önden okumaya izin verilip verilmediğini belirleyin.</li> </ul>	<p>MDB ' nin hedeften kalıcı olmayan iletileri almadan önce bir iç arabelleğe almak için önden okuma özelliğini kullanıp kullanmasına izin verilip verilmediğini belirler</p>



Çizelge 64. JMS bağlantı tüketicisi yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özelliğın adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
readAheadClosePolicy	int	<ul style="list-style-type: none"> <li>• <b>ALL</b> -İç önden okuma arabelleğindeki tüm iletiler, durdurulmadan önce MDB ' ye teslim edilir.</li> <li>• <b>CURRENT</b>-İletileri daha sonra atılacak iç önden okuma arabelleğinde bırakarak, yalnızca yürürlükteki MDB çağırısı tamamlanır.</li> </ul>	MDB yönetici tarafından durdurulduğunda iç önden okuma arabelleğindeki iletilere ne olur?
receiveCCSID	int	<ul style="list-style-type: none"> <li>• <b>0</b> -JVM kullan Charset.defaultCharset</li> <li>• 1208- UTF-8</li> <li>• Desteklenen bir kodlanmış karakter takımı tanıtıcısı</li> </ul>	Kuyruk yöneticisi ileti dönüştürmesi için hedef CCSID ' yi belirleyen hedef özellik. <b>receiveConversion</b> , QMGRolarak ayarlanmadıkça değer yoksayılr.
receiveConversion	Dizgi	<ul style="list-style-type: none"> <li>• <b>CLIENT_MSG</b></li> <li>• QMGR</li> </ul>	Veri dönüştürme işleminin kuyruk yöneticisi tarafından gerçekleştirilip gerçekleştirilmeyeceğini belirleyen hedef özellik.
sharedSubscription	Boole	<ul style="list-style-type: none"> <li>• <b>False</b> -MDB, aboneliği paylaşılan abonelik olarak açmamalıdır.</li> <li>• <b>True</b>-MDB, aboneliği paylaşılan abonelik olarak açmalıdır ( JMS 2.0 ' in belirttiği kurallarla birlikte, <a href="#">Java.net</a>adresindeki JMS 2.0 belirtimine bakın).</li> </ul>	Bir MDB ' nin paylaşılan bir abonelikten nasıl yönlendirildiğini denetler. Bu özelliğın nasıl kullanılacağına ilişkin daha fazla bilgi için bkz. <a href="#">“sharedSubscription özelliğının nasıl tanımlamaya ilişkin örnekler” sayfa 444.</a>
startTimeout	int	<ul style="list-style-type: none"> <li>• <b>10 000</b></li> <li>• Artı bir tamsayı</li> </ul>	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	<ul style="list-style-type: none"> <li>• <b>NonDurable</b> -Sürekli olmayan abonelik, konuya abone olan bir MDB ' ye ileti teslim etmek için kullanılır.</li> <li>• <b>Sürekli</b>-Sürekli abonelik, konuya abone olan bir MDB ' ye ileti göndermek için kullanılır.</li> </ul>	Konuya abone olan bir MDB ' ye ileti teslim etmek için sürekli ya da sürekli olmayan bir aboneliğın kullanılıp kullanılmayacağı
subscriptionName	Dizgi	<ul style="list-style-type: none"> <li>• <b>"" (boş dizgi)</b></li> <li>• Abonelik adı</li> </ul>	Sürekli aboneliğın adı

Çizelge 64. JMS bağlantı tüketicisi yaratmak için kullanılan ActivationSpec nesnesinin özellikleri (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
useJNDI	Boole	<ul style="list-style-type: none"> <li><b>false</b> -Hedef adı verilen özellik, bir IBM MQ kuyruğunun ya da konusunun adı olarak yorumlanır.</li> <li><b>true</b>-Hedef adı verilen özellik, uygulama sunucusunun JNDI ad alanında bir javax.jms.Queue nesnesinin ya da javax.jms.Topic nesnesinin adı olarak yorumlanır.</li> </ul>	<p>Hedef adı verilen özelliğin değerinin nasıl yorumlanacağını belirler</p> <p><b>Not:</b> Bu özellik IBM MQ 9.0içinde kullanımdan kaldırılmıştır. Bunun yerine <u>destinationLookup</u> özelliği kullanılmalıdır.</p>

## 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 yoksayılr.

Diğer özelliklerin tanımlanmasını gerektiren bir özellik tanımlarsanız, ancak bu diğer özellikleri tanımlamazsanız, ActivationSpec nesnesi bir MDB konuşlandırması sırasında validate () yöntemi çağırıldığında 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 tanımlanan istemci kanal tanımlama çizelgesi kullanılır.

## ActivationSpec connectionFactoryLookup ve destinationLookup özellikleri

Bu iki özellik, Çizelge 63 sayfa 430 ve Çizelge 64 sayfa 439içinde 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 63 sayfa 430içinde 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. ActivationSpecüzzerinde **ClientID** ayarlanmazsa, bağlantı üreticisinin herhangi bir değeri kullanılır.

## destinationLookup

ActivationSpec içinde 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 JNDI tarafından aranır. Bu özellik, **useJNDI** özelliğinden önceliklidir.

**destinationLookup** özelliği, aynı işlevi gerçekleştirmenin JMS 2.0 belirtimi eşdeğeri olduğundan, useJNDI özelliği 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 nesnelerinin özelliklerine ya da IBM MQ classes for JMS yöntemlerinin parametrelerine eşdeğerdendir. Ancak, üç ayarlama özelliği ve bir kullanılabilirlik özelliği, IBM MQ classes for JMS içinde 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 zaman aşımına 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 etkinlik dışı kalma süresi alt sınırını gösterir.

### useJNDI

Bu özelliğin açıklaması için bkz. [Çizelge 64 sayfa 439](#).

## 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 kümesidir:

```
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 422.](#)

Aynı hedeften birden çok 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 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, 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 209.](#)

### **İlgili görevler**

[MQI istemcisinde çalıştırma zamanında yalnızca FIPS onaylı CipherSpecs kullanılmasının belirtilmesi](#)  
[WebSphere Application Server ' de JMS kaynaklarının yapılandırılması](#)

### **İlgili başvurular**

[UNIX, Linux ve Windows için Federal Bilgi İşleme Standartları \(FIPS\)](#)

*sharedSubscription özelliğinin nasıl tanımlamaya ilişkin örnekler*

Bir WebSphere Liberty server.xml dosyası içinde bir etkinleştirme belirtiminin sharedSubscription özelliğini tanımlayabilirsiniz. Diğer bir seçenek olarak, ek açıklamaları kullanarak bir ileti odaklı Bean (MDB) içinde özelliği tanımlayabilirsiniz.

## **Örnek: bir Liberty server.xml dosyası içinde tanımlama**

Bir WebSphere Liberty server.xml dosyası içinde, aşağıdaki örnekte gösterildiği gibi bir etkinleştirme belirtimi tanımladınız. Bu örnek, yerel anasistem/kapı 1490 'da 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: bir MDB içinde tanımlama**

Ayrıca, aşağıdaki örnekteki gibi ek açıklamaları kullanarak MDB içindeki sharedSubscription özelliğini tanımlayabilirsiniz:

```
@ActioncationConfigProperty(propertyName = "sharedSubscription",  
propertyValue = "true")
```

Aşağıdaki örnekte, ek açıklama yöntemini kullanan bir MDB kodu örneği gösterilmektedir:

```

/**
 * 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

[“Eşkopyalanmış ve paylaşılan aboneli” sayfa 303](#)

IBM MQ 8.0 ya da sonraki bir yayın düzeyiyle, birden çok tüketicinin aynı aboneliğe erişmesine izin vermek için iki yöntem vardır. Bu iki yöntem, eşkopyalanmış abonelikler kullanılarak ya da paylaşılan abonelikler kullanılarak olur.

## Kaynak bağıdaştırıcısının 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:

```

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 uygulamacığı bir HTTP GET isteği aldığı anda, JNDI ad alanından bir ConnectionFactory nesnesi ve bir Kuyruk nesnesi alır ve nesnelere IBM MQ kuyruğuna ileti göndermek için 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 nesnelere, 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 2.0 ile 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 nesnelere içermesini sağlayarak havuzu daha verimli hale getirir.

## ConnectionFactory nesnesinin özellikleri

Çizelge 65 sayfa 446 içinde bir ConnectionFactory nesnesinin özellikleri listelenir. Uygulama sunucusu, bir JMS ConnectionFactory nesnesi yaratmak için bu özellikleri kullanır.

Çizelge 65. ConnectionFactory nesnesinin özellikleri			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
applicationName	Dizgi	<ul style="list-style-type: none"> <li>• Çağırılan sınıf adı (varsa), 28 karakterden uzun olmayacak şekilde ayarlanır. Yoksa, WebSphere MQ Client for Java dizisi kullanılır.</li> </ul>	<p>Bir uygulamanın kuyruk yöneticisine kayıtlı olduğu ad. Bu uygulama adı, <b>DISPLAY CONN MQSC/PCF</b> komutuyla (alanın <b>APPLTAG</b> olarak adlandırıldığı) ya da IBM MQ Gezgün <b>Uygulama Bağlantıları</b> görüntüsünde (burada alan <b>App name</b> olarak adlandırılır) gösterilir.</p>

Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
brokerCCSubKuyruk <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE</b></li> <li>• Kuyruk adı</li> </ul>	Bağlantı tüketicisinin sürekli olmayan abonelik iletileri aldığı kuyruğun adı.
brokerControlKuyruk <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.BROKER.CONTROL.QUEUE</b></li> <li>• Kuyruk adı</li> </ul>	Aracı denetim kuyruğunun adı.
brokerPubKuyruk <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.BROKER.DEFAULT.STREAM</b></li> <li>• Kuyruk adı</li> </ul>	Yayınlanan iletilerin gönderildiği kuyruğun adı (akış kuyruğu).
brokerQueueManager <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• "" (<b>boş dizgi</b>)</li> <li>• Kuyruk yöneticisi adı</li> </ul>	Aracının çalıştığı kuyruk yöneticisinin adı.
brokerSubKuyruk <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.JMS.ND.SUBSCRIBER.QUEUE</b></li> <li>• Kuyruk adı</li> </ul>	Sürekli olmayan bir ileti tüketicisinin iletileri aldığı kuyruğun adı.  Ek bilgi için <a href="#">BROKERSUBQ</a> özelliğine bakın.
brokerVersion <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>belirlenmedi</b> -Aracı V6 'dan V7' ye geçirildikten sonra, bu özelliği RFH2 üstbilgilerinin artık kullanılmayacak şekilde ayarlayın. Geçişten sonra bu özellik artık ilgili değildir.</li> <li>• <b>V1</b> - IBM MQ Yayınla/Abone Ol aracısını kullanmak için. TRANSPORT, BIND ya da CLIENT olarak ayarlandıysa, bu değer varsayılan değerdir.</li> <li>• <b>V2</b> -Yerel kipte IBM Integration Bus aracısını kullanmak için. TRANSPORT değeri DIRECT ya da DIRECTHTTP olarak ayarlandıysa, bu değer varsayılan değerdir.</li> </ul>	Kullanılmakta olan aracının sürümü.
ccdtURL	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Birörnek kaynak konum belirleyici (URL)</li> </ul>	İstemci kanal tanımlama çizelgesini (CCDT) içeren dosyanın adını ve yerini tanımlayan ve dosyaya nasıl erişilebileceğini belirten URL.
CCSID	Dizgi	<ul style="list-style-type: none"> <li>• <b>819</b></li> <li>• Java sanal makinesi (JVM) tarafından desteklenen bir kodlanmış karakter takımı tanıtıcısı</li> </ul>	Bağlantıya ilişkin kodlanmış karakter takımı tanıtıcısı.
kanal	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.DEF.SVRCONN</b></li> <li>• MQI kanalının adı</li> </ul>	Kullanılacak MQI kanalının adı.

Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
cleanupInterval <sup>1</sup> (Temizleme Aralığı)	int	<ul style="list-style-type: none"> <li>• <b>3600 000</b></li> <li>• Artı bir tamsayı</li> </ul>	Yayınlama/abone olma temizleme yardımcı programının arka plan çalıştırmaları arasındaki milisaniye cinsinden aralık.
cleanupLevel <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>GÜVENLİ</b></li> <li>• YOK</li> <li>• güçlü</li> <li>• ZORLA</li> <li>• NONDUR</li> </ul>	Aracıya dayalı bir abonelik deposunun temizleme düzeyi.
clientID	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Bir istemci tanıtıcısı</li> </ul>	Bağlantıya ilişkin istemci tanıtıcısı.
cloneSupport	Dizgi	<ul style="list-style-type: none"> <li>• <b>DEVRE DIŞI</b> -Bir kerede tek bir sürekli konu abonesi çalıştırılabilir.</li> <li>• <b>ENABLED</b>-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.</li> </ul>	Aynı sürekli konu abonesinin iki ya da daha fazla eşgörünümünün aynı anda çalışıp çalışmayacağını belirler.
connectionNameListesi	Dizgi	<ul style="list-style-type: none"> <li>• <b>localhost (1414)</b></li> <li>• Her öğenin biçimi aldığı, virgüllerle ayrılmış öğelerden oluşan bir dizgi:   <div style="background-color: #e0e0e0; padding: 5px; text-align: center;">HOSTNAME (PORT)</div> <p>Burada <i>HOSTNAME</i> , bir DNS adı ya da bir IP adresidir.</p> </li> </ul>	<p>Giden iletişim için kullanılan TCP/IP bağlantı adlarının listesi.</p> <p><b>connectionNameList</b> , <b>hostname</b> ve <b>port</b> özelliklerinin yerini alır.</p> <p>Bu özellik, çok eşgörünümlü kuyruk yöneticilerine yeniden bağlanmak için kullanılır.</p> <p><b>connectionNameList</b> , formda <b>localAddress</b> ile benzer, ancak bununla karıştırılmamalıdır. <b>localAddress</b> , yerel iletişimin özelliklerini belirtirken, <b>connectionNameList</b> uzak kuyruk yöneticisine nasıl erişileceğini belirtir.</p>
failIfSusturma	Boole	<ul style="list-style-type: none"> <li>• <b>doğru</b></li> <li>• yanlış</li> </ul>	Kuyruk yöneticisi susturma durumundaysa, belirli yöntemlere yönelik çağrılarının başarısız olup olmayacağını belirler.
headerCompression	Dizgi	<ul style="list-style-type: none"> <li>• <b>YOK</b></li> <li>• <b>SYSTEM-RLE</b> iletici üstbilgisi sıkıştırması gerçekleştirilir.</li> </ul>	Bir bağlantıdaki üstbilgi verilerinin sıkıştırılması için kullanılacak tekniklerin listesi.



Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)

Özellğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
hostName	Dizgi	<ul style="list-style-type: none"> <li>• <b>localhost</b></li> <li>• Anasistem adı</li> <li>• Bir IP adresi</li> </ul>	<p>Kuyruk yöneticisinin bulunduğu sistemin anasistem adı ya da IP adresi.</p> <p><b>hostname</b> ve <b>port</b> özellikleri, belirtildiğinde <b>connectionNameList</b> özelliği tarafından değiştirilir.</p>
localAddress	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Şu biçimde bir dizgi:</li> </ul> <pre>[ host_name ][(low_port [, high_port ])]</pre> <p>Burada <i>anasistem_adi</i> bir anasistem adı ya da IP adresi, <i>alt_kapı</i> ve <i>yüksek_kapı</i> TCP kapı numaralarıdır ve köşeli ayraçlar isteğe bağlı bir bileşeni gösterir</p>	<p>Bir kuyruk yöneticisine yönelik bağlantı için, bu özellik aşağıdakilerden birini ya da her ikisini de belirtir:</p> <ul style="list-style-type: none"> <li>• Kullanılacak yerel ağ arabirimi</li> <li>• Kullanılacak yerel kapı ya da yerel kapı aralığı</li> </ul> <p><b>localAddress</b> , formda <b>connectionNameList</b> ile benzer, ancak bununla karıştırılmamalıdır. <b>localAddress</b> , yerel iletişimin özelliklerini belirtirken, <b>connectionNameList</b> uzak kuyruk yöneticisine nasıl erişileceğini belirtir.</p>
messageCompression	Dizgi	<ul style="list-style-type: none"> <li>• <b>YOK</b></li> <li>• Boş karakterlerle ayrılmış olarak aşağıdaki değerlerden birinin ya da daha fazlasının listesi:</li> </ul> <p>RLE ZLIBFAST ZLIBHIGH</p>	<p>Bir bağlantıdaki ileti verilerini sıkıştırmada kullanılabilecek tekniklerin listesi.</p>
messageSelection <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>İstemci</b></li> <li>• Aracı</li> </ul>	<p>İleti seçiminin IBM MQ classes for JMS tarafından mı, yoksa aracı tarafından mı yapılacağını belirler. <b>brokerVersion</b> 1 değerine sahip olduğunda aracı tarafından ileti seçimi desteklenmez.</p>
parola	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Parola</li> </ul>	<p>Kuyruk yöneticisine bağlantı yaratılırken kullanılacak varsayılan parola.</p>

Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğın adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
pollingInterval <sup>1</sup>	int	<ul style="list-style-type: none"> <li>• <b>5000</b></li> <li>• Pozitif bir tamsayı</li> </ul>	Bir oturumdaki her ileti dinleyicinin kuyruğunda uygun bir ileti yoksa, bu değer, her ileti dinleyicisinin kuyruğundan bir ileti almaya çalışmasından önce geçmesi gereken milisaniye cinsinden aralık üst sınırıdır. Sık sık oturumdaki ileti dinleyicilerden herhangi biri için uygun bir ileti yoksa, bu özelliğın değerini artırmayı düşünebilirsiniz. Bu özellik yalnızca <b>TRANSPORT</b> , <b>BIND</b> ya da <b>CLIENT</b> değerine sahipse geçerlidir.
kapı	int	<ul style="list-style-type: none"> <li>• <b>1414</b></li> <li>• TCP kapı numarası</li> </ul>	Kuyruk yöneticisinin dinlediğı kapı. <b>hostname</b> ve <b>port</b> özellikleri, belirtildiğinde <b>connectionNameList</b> özelliğı tarafından değıştirilir.
providerVersion	dizgi	<ul style="list-style-type: none"> <li>• <b>belirtilmedi</b></li> <li>• Aşağıdaki biçimlerden birindeki bir dizgi <ul style="list-style-type: none"> <li>– V.R.M.F</li> <li>– V.R.M</li> <li>– V.R</li> <li>– V</li> </ul> </li> </ul> <p>Burada V, R, M ve F sıfırdan büyük ya da sıfıra eşit tamsayı değerleridir.</p>	Uygulamanın bağlanmayı planladığı kuyruk yöneticisinin sürümü, yayın düzeyi, değışiklik düzeyi ve düzeltme paketi.
pubAckAralık <sup>1</sup>	int	<ul style="list-style-type: none"> <li>• <b>25</b></li> <li>• Artı bir tamsayı</li> </ul>	IBM MQ classes for JMS aracından bir alındı bildirimini istemeden önce bir yayınlayıcı tarafından yayınlanan iletilerin sayısı.
queueManager	Dizgi	<ul style="list-style-type: none"> <li>• <b>"" (boş dizgi)</b></li> <li>• Kuyruk yöneticisi adı</li> </ul>	Bağlanılacak kuyruk yöneticisinin adı.
receiveExit <sup>3</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• 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).</li> </ul>	Bir kanal alma çıkış programını ya da arka arkaya çalıştırılacak bir alma çıkış programları sırasını tanımlar.

Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
receiveExitBaşlatma	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Virgüllerle ayrılmış bir ya da daha fazla kullanıcı verisi ögesini içeren bir dizgi</li> </ul>	Kanala geçirilen kullanıcı verileri, çağrıldıklarında çıkış programlarını alır.
rescanInterval <sup>1</sup>	int	<ul style="list-style-type: none"> <li>• <b>5000</b></li> <li>• Pozitif bir tamsayı</li> </ul>	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 <b>MsgDeliverySequence</b> 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 ulaşınca 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 <sup>3</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• IBM MQ classes for Java arabirimini gerçekleştiren bir sınıfın tam olarak nitelenmiş adı, MQSecurityExit</li> </ul>	Bir kanal güvenliği çıkış programını tanımlar.
securityExitBaşlatma	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• Kullanıcı verileri dizgisi</li> </ul>	Çağrıldığında kanal güvenliği çıkış programına geçirilen kullanıcı verileri.
sendCheckSayısı	int	<ul style="list-style-type: none"> <li>• <b>ERROR! SEGMENT DATA CORRUPTED, SEGDATA=0</b></li> <li>• Pozitif bir tamsayı</li> </ul>	Hareket etmeyen tek bir JMS oturumunda zamanuyumsuz koyma hatalarının denetlenmesi arasında izin verilecek gönderme çağrılarının sayısı.
sendExit <sup>3</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>boş değerli</b></li> <li>• 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 ).</li> </ul>	Kanal gönderme çıkış programını ya da art arda çalıştırılacak çıkış gönderme programlarını tanımlar.

Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
sendExitInit	Dizgi	<ul style="list-style-type: none"> <li><b>boş değerli</b></li> <li>Virgüllerle ayrılmış bir ya da daha fazla kullanıcı verisi ögesini içeren bir dizgi</li> </ul>	Kanala geçirilen kullanıcı verileri, çağrıldığı anda çıkış programlarını gönderir.
shareConvİzin Verilir	Boole	<ul style="list-style-type: none"> <li>NO-Bir istemci bağlantısı yuvasını paylaşmıyor.</li> <li><b>YES (EVET)</b>-Bir istemci bağlantısı yuvasını paylaşabilir.</li> </ul>	Bir istemci bağlantısının, kanal tanımlamaları eşleşiyorsa, aynı işlemden aynı kuyruk yöneticisine diğer üst düzey JMS bağlantılarıyla yuvasını paylaşıp paylaşamayacağını belirler.
sparseSubscriptions <sup>1</sup>	Boole	<ul style="list-style-type: none"> <li><b>false</b> -Abonelikler sık sık eşleşen iletiler alır.</li> <li>true-Abonelikler nadiren eşleşen iletiler alır. Bu değer, abonelik kuyruğunun göz atmak için açılabilmesini gerektirir.</li> </ul>	TopicSubscriber nesnesinin ileti alma ilkesini denetler.
sslCertMağazaları	Dizgi	<ul style="list-style-type: none"> <li><b>boş değerli</b></li> <li>Boşluklarla ayrılmış bir ya da daha çok LDAP URL dizesi. Her LDAP URL adresi şu biçimdedir: <pre>ldap://host_name [: port ]</pre> <p>Burada <i>anasistem_adi</i> bir anasistem adı ya da IP adresidir, <i>kapı</i> bir TCP kapı numarasıdır ve köşeli parantezler isteğe bağlı bir bileşeni gösterir.</p> </li> </ul>	TLS bağlantısında kullanılmak üzere sertifika iptal listelerini (CRL 'ler) tutan LDAP sunucuları.
sslCipherÜrün Grubu	Dizgi	<ul style="list-style-type: none"> <li><b>boş değerli</b></li> <li>CipherSuite adı</li> </ul>	TLS bağlantısı için kullanılacak CipherSuite .
sslFipsGerekli <sup>2</sup>	Boole	<ul style="list-style-type: none"> <li><b>yanlış</b></li> <li>doğru</li> </ul>	Bir TLS bağlantısının, IBM Java JSSE FIPS sağlayıcısı (IBMJSSEFIPS) tarafından desteklenen bir CipherSuite kullanması gerekip gerekmediğini belirler.
sslPeerAdı	Dizgi	<ul style="list-style-type: none"> <li><b>boş değerli</b></li> <li>Ayırt edici adlar için şablon</li> </ul>	TLS bağlantısı için, kuyruk yöneticisi tarafından sağlanan sayısal sertifikada ayırt edici adı denetlemek için kullanılan bir şablon.
sslResetSayısı	int	<ul style="list-style-type: none"> <li><b>ERROR! SEGMENT DATA CORRUPTED, SEGDATA=0</b></li> <li>0-999 999 999 aralığında bir tamsayı</li> </ul>	TLS tarafından kullanılan gizli anahtarlar yeniden anlaşılmadan önce bir TLS bağlantısı tarafından gönderilen ve alınan toplam bayt sayısı.

Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)

Özellğin 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; isteğe bağlı olarak, oluşturucu yöntemine geçirilecek bir bağımsız değişken de içinde olmak üzere, araç içine alınmış olarak.	Yönetilen hedef nesne kapsamında kurulan tüm bağlantılar, SSLSocketFactory arabiriminin bu somutlamasından elde edilen yuvaları kullanır.
statusRefreshInterval <sup>1</sup> (Durum Yenileme)	int	<ul style="list-style-type: none"> <li>• <b>60000</b></li> <li>• Pozitif bir tamsayı</li> </ul>	Bir abonenin kuyruk yöneticisine olan bağlantısını kaybettiği zaman algılanan, uzun süren hareketin yenilenmesi arasındaki milisaniye cinsinden aralık. Bu özellik yalnızca <b>SUBSTORE</b> QUEUEdeğerine sahipse geçerlidir.
subscriptionStore <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>Aracı</b></li> <li>• GEÇİŞ YAPIN</li> <li>• kuyruk</li> </ul>	IBM MQ classes for JMS , etkin aboneliklerle ilgili kalıcı verilerin nerede saklanacağını belirler.
targetClientEşleştirme	Boole	<ul style="list-style-type: none"> <li>• <b>doğru</b></li> <li>• yanlış</li> </ul>	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 " <a href="#">Bir etkinleştirme belirtimi için targetClientEşleştirme özelliğinin yapılandırılması</a> " sayfa 462.


Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğın adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
temporaryModel	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.DEFAULT.MODEL.QUEUE</b></li> <li>• SYSTEM.JMS.TEMPQ.MODEL</li> <li>• Herhangi bir dizgi</li> </ul>	<p>JMS geçici kuyruklarının yaratıldığı model kuyruğunun adı. SYSTEM.DEFAULT.MODEL.QUEUE :</p> <ul style="list-style-type: none"> <li>• Uygulamanız, kalıcı olmayan iletileri kabul edecek geçici bir kuyruk kullanıyor.</li> <li>• 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.</li> </ul> <p>SYSTEM.JMS.TEMPQ.MODEL :</p> <ul style="list-style-type: none"> <li>• Uygulamanız kalıcı iletileri kabul edecek geçici bir kuyruk kullandığında.</li> <li>• Birden çok uygulama ConnectionFactory ' in gösterdiği kuyruk yöneticisine bağlanabiliyorsa ve bu uygulamaların aynı anda geçici kuyruklar oluşturmaları gerekiyorsa.</li> </ul> <p><b>DEFPSIST</b> özniteliği YESdeğerine ayarlanmış olarak yeni bir model kuyruğu tanımlayın ve <b>DEFSOPT</b> özniteliği aşağıdaki durumda SHARED değerine ayarlandı:</p> <ul style="list-style-type: none"> <li>• Uygulamanız kalıcı olmayan iletileri kabul edecek geçici bir kuyruk kullandığında ve birden çok uygulama ConnectionFactory ' in işaret ettiği kuyruk yöneticisine bağlandığında ve bu uygulamaların aynı anda geçici kuyruklar oluşturmaları gerekir.</li> </ul> <p>Yeni model kuyruğu yaratıldığında, <b>temporaryModel</b> özelliğini yeni model kuyruğunun adına ayarlayın.</p>

Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)

Özelliğın adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
tempQPrefix	Dizgi	<ul style="list-style-type: none"><li>• "" (boş dizgi)</li><li>• Bir IBM MQ dinamik kuyruğunun adını oluşturmak için kullanılabilir bir önek. Öneki oluşturmaya ilişkin kurallar, IBM MQ nesne tanımlayıcısı, MQOD yapısı içindeki <b>DynamicQName</b> alanının içeriğini oluşturmaya ilişkin kurullarla aynıdır, ancak son boş olmayan karakter bir yıldız işareti (*) olmalıdır. Özelliğın değeri boş dizgiyse, IBM MQ classes for JMS AMQ.* değerini kullanır. Dinamik bir kuyruk yaratırken.</li></ul>	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 değeriyle bırakılırsa, yalnızca "TEMP/unique_id". Boş olmayan bir <b>TEMPTOPICPREFIX</b> belirtilmesi, abonelerin bu bağlantı altında yaratılan geçici konulara ilişkin yönetilen kuyrukları yaratmak için belirli model kuyruklarının tanımlanmasına olanak sağlar.

Çizelge 65. ConnectionFactory nesnesinin özellikleri (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
transportType	Dizgi	<ul style="list-style-type: none"> <li>İstemci</li> <li>Bağ Tanımları</li> <li>BINDINGS_THEN_CLIENT</li> </ul>	<p>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.</p> <p> 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.</p>
kullanıcı adı	Dizgi	<ul style="list-style-type: none"> <li>boş değerli</li> <li>Kullanıcı adı</li> </ul>	Bir kuyruk yöneticisine bağlantı yaratırken kullanılacak varsayılan kullanıcı adı.
wildcardFormat	int	<ul style="list-style-type: none"> <li>CHAR-Aracı sürüm 1 'de kullanıldığı şekilde, yalnızca genel arama karakterlerini tanır</li> <li>TOPIC-Aracı sürüm 2 'de kullanıldığı şekilde, yalnızca konu düzeyi genel arama karakterlerini tanır</li> </ul>	Genel arama karakteri sözdiziminin hangi sürümünün kullanılacağı.

#### Notlar:

1. Bu özellik, IBM WebSphere MQ classes for JMS sürüm 7.0 ile birlikte kullanılabilir, ancak providerVersion özelliği 7 'den küçük bir sürüm numarasına ayarlanmadıkça IBM WebSphere MQ 7.0 kuyruk yöneticisine bağlı bir uygulamayı etkilemez.
2. sslFipsRequired özelliğini kullanma hakkında önemli bilgi için bkz. [“IBM MQ kaynak bağdaştırıcısına ilişkin sınırlamalar” sayfa 418.](#)
3. Kaynak bağdaştırıcısının bir çıkış bulabilmesi için nasıl yapılandırılacağı hakkında bilgi için bkz. [“Configuring IBM MQ classes for JMS to use channel exits” sayfa 260.](#)

Aşağıdaki örnek, bir ConnectionFactory nesnesinin tipik bir özellik kümesini göstermektedir:

```
channel1:          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 66 sayfa 457 içinde bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler listelenir.

Çizelge 66. Bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler			
Özellğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
CCSID	Dizgi	<ul style="list-style-type: none"><li>• <b>1208</b></li><li>• Java sanal makinesi (JVM) tarafından desteklenen bir kodlanmış karakter takımı tanıtıcısı</li></ul>	Hedefe ilişkin kodlanmış karakter takımı tanıtıcısı.
Kodlama	Dizgi	<ul style="list-style-type: none"><li>• <b>Yerel</b></li><li>• Üç karakterden oluşan bir dizgi:<ul style="list-style-type: none"><li>– İlk karakter ikili tamsayıların gösterimini belirtir:<ul style="list-style-type: none"><li>- <i>N</i>, normal kodlamayı gösterir.</li><li>- <i>R</i> ters kodlamayı belirtir.</li></ul></li><li>– İkinci karakter paketlenmiş ondalık tamsayıların gösterimini belirtir:<ul style="list-style-type: none"><li>- <i>N</i>, normal kodlamayı gösterir.</li><li>- <i>R</i> ters kodlamayı belirtir.</li></ul></li><li>– Üçüncü karakter, kayan noktalı sayıların gösterimini belirtir:<ul style="list-style-type: none"><li>- <i>N</i> standart IEEE kodlamasını belirtir.</li><li>- <i>R</i>, ters IEEE kodlamasını belirtir.</li><li>- <i>3</i>, zSeries kodlamasını belirtir.</li></ul></li></ul></li></ul> <p>NATIVE, NNN dizesine eşdeğerdir.</p>	Hedef için ikili tamsayıların, paketlenmiş ondalık tamsayıların ve kayan noktalı sayıların gösterimi.
Son kullanma tarihi	Dizgi	<ul style="list-style-type: none"><li>• <b>APP</b> -İletinin süre bitimi, ileti üreticisi tarafından belirlenir.</li><li>• UNLIM-Bir iletinin süresi hiçbir zaman dolmaz.</li><li>• 0-Bir iletinin süresi hiçbir zaman dolmaz.</li><li>• Bir iletinin milisaniye cinsinden süre bitimini gösteren pozitif bir tamsayı.</li></ul>	Hedefe gönderilen bir iletinin süre bitimi.
failIfSusturma	Dizgi	<ul style="list-style-type: none"><li>• <b>doğru</b></li><li>• yanlış</li></ul>	Kuyruk yöneticisi susturma durumundaysa, hedefe erişme girişiminin başarısız olup olmadığını belirler.

Çizelge 66. Bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler (devamı var)

Özellğin adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
messageBodyBiçemi	Dizgi	<ul style="list-style-type: none"><li>• <b>Belirtilmedi</b></li><li>• JMS</li><li>• MQ</li></ul>	<p><b>messageBodyStyle</b> özelliğini JMS kuyruklarında ve konularında ayarlayabilirsiniz: UNSPECIFIED (varsayılan)</p> <ul style="list-style-type: none"><li>• Gönderirken, WMQ_TARGET_CLIENT değerine bağlı olarak IBM MQ classes for JMS bir MQRFH2 üstbilgisi oluşturun ve ekleyin.</li><li>• Alma sırasında, IBM MQ classes for JMS JMS ileti özelliklerini MQRFH2' deki değerlere göre ayarlayın (varsa). MQRFH2 , JMS ileti gövdesinin bir parçası olarak sunulmaz.</li></ul> <p>JMS</p> <ul style="list-style-type: none"><li>• Gönderirken IBM MQ classes for JMS otomatik olarak bir MQRFH2 üstbilgisi oluşturur ve IBM MQ iletisindeki üstbilgiyi içerir.</li><li>• Alma sırasında, IBM MQ classes for JMS JMS ileti özelliklerini MQRFH2' deki değerlere göre ayarlayın (varsa). MQRFH2 , JMS ileti gövdesinin bir parçası olarak sunulmaz.</li></ul> <p>MQ</p> <ul style="list-style-type: none"><li>• IBM MQ classes for JMS gönderirken bir MQRFH2oluşturmayın.</li><li>• IBM MQ classes for JMS , alırken JMS ileti gövdesinin bir parçası olarak MQRFH2 ' yi sunar.</li></ul>

Çizelge 66. Bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler (devamı var)

Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
Kalıcılık	Dizgi	<ul style="list-style-type: none"> <li>• <b>APP</b> -Bir iletinin sürekliliği, ileti üreticisi tarafından belirlenir.</li> <li>• <b>QDEF</b>-Bir iletinin kalıcılığı, IBM MQ kuyruğunun <b>DefPersistence</b> özneliği tarafından belirlenir.</li> <li>• <b>PERS</b>-Bir ileti kalıcı.</li> <li>• Hayır-Bir ileti kalıcı değil.</li> <li>• <b>HIGH</b>-Bir iletinin kalıcı olarak saklanması, IBM MQ kuyruğunun <b>NonPersistentMessageClass</b> özneliği, “JMS kalıcı iletileri” sayfa 224’indeki açıklamaya göre belirlenir.</li> </ul>	Hedefe gönderilen bir iletinin kalıcılığı.
öncelik	Dizgi	<ul style="list-style-type: none"> <li>• <b>APP</b> -Bir iletinin önceliği, ileti üreticisi tarafından belirlenir.</li> <li>• <b>QDEF</b>-Bir iletinin önceliği, IBM MQ kuyruğunun <b>DefPriority</b> özneliğine göre belirlenir.</li> <li>• 0, en düşük öncelik, 9, en yüksek öncelik aralığında bir tamsayı.</li> </ul>	Hedefe gönderilen bir iletinin önceliği.
putAsync' e izin verilir	Dizgi	<ul style="list-style-type: none"> <li>• <b>QUEUE</b>-Kuyruk tanımına başvurarak zamanuyumsuz girişlere izin verilip verilmediğini saptayın.</li> <li>• <b>KONU</b>-Konu tanımlamasına başvurarak zamanuyumsuz girişlere izin verilip verilmediğini saptayın.</li> <li>• <b>DESTINATION</b>-Kuyruk ya da konu tanımlamasına başvurarak zamanuyumsuz girişlere izin verilip verilmediğini belirler.</li> <li>• <b>DISABLED</b>-Zamanuyumsuz girişlere izin verilmez.</li> <li>• <b>ENABLED</b>-Zamanuyumsuz girişlere izin verilir.</li> </ul>	İleti üreticilerinin bu hedefe ileti göndermek için zamanuyumsuz girişler kullanmalarına izin verilip verilmediğini belirler.
readAheadİzin verilir	int	<ul style="list-style-type: none"> <li>• <b>DESTINATION</b> -Kuyruk ya da konu tanımlamasına başvurarak önden okumaya izin verilip verilmediğini belirleyin.</li> <li>• <b>DISABLED</b>-Önden okumaya izin verilmez.</li> <li>• <b>ENABLED</b>-Önden okumaya izin verilir.</li> <li>• <b>QUEUE</b>-Kuyruk tanımına başvurarak önden okumaya izin verilip verilmediğini belirleyin.</li> <li>• <b>KONU</b>-Konu tanımlamasına başvurarak önden okumaya izin verilip verilmediğini belirleyin.</li> </ul>	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 66. Bir Kuyruk nesnesi ve Konu nesnesi için ortak olan özellikler (devamı var)			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
receiveCCSID	int	<ul style="list-style-type: none"> <li>• 0 -JVM kullan Charset.defaultCharset</li> <li>• 1208- UTF-8</li> <li>• Desteklenen bir kodlanmış karakter takımı tanıtıcısı</li> </ul>	Kuyruk yöneticisi ileti dönüştürmesi için hedef CCSID ' yi belirleyen hedef özellik. <b>receiveConversion</b> , QMGRolarak ayarlanmadıkça değer yoksayılr.
receiveConversion	Dizgi	<ul style="list-style-type: none"> <li>• <b>CLIENT_MSG</b></li> <li>• QMGR</li> </ul>	Veri dönüştürme işleminin kuyruk yöneticisi tarafından gerçekleştirilip gerçekleştirilmeyeceğini belirleyen hedef özellik.
targetClient	Dizgi	<ul style="list-style-type: none"> <li>• <b>JMS</b> -İletinin hedefi bir JMS uygulamasıdır.</li> <li>• MQ -Bir iletinin hedefiJMS IBM MQ dışı bir uygulamadır.</li> </ul>	Hedefe gönderilen bir iletinin hedefinin bir JMS uygulaması olup olmadığını belirler. JMS uygulaması olan bir hedef içeren bir ileti, MQRFH2 üstbilgisi içeriyor.

Çizelge 67 sayfa 460 içinde bir Kuyruk nesnesine özgü özellikler listelenir.

Çizelge 67. Kuyruk nesnesine özgü özellikler			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
baseQueueManagerName	Dizgi	<ul style="list-style-type: none"> <li>• "" (boş dizgi)</li> <li>• Kuyruk yöneticisi adı</li> </ul>	Temel IBM MQ kuyruğunun sahibi olan kuyruk yöneticisinin adı.
baseQueueAdı	Dizgi	<ul style="list-style-type: none"> <li>• "" (boş dizgi)</li> <li>• Kuyruk adı</li> </ul>	Temel IBM MQ kuyruğunun adı.

Çizelge 68 sayfa 460 içinde, bir Konu nesnesine özgü özellikler listelenir.

Çizelge 68. Konu nesnesine özgü özellikler			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
baseTopicAdı	Dizgi	<ul style="list-style-type: none"> <li>• "" (boş dizgi)</li> <li>• Konu adı</li> </ul>	Temeldeki konunun adı.
brokerCCDurSubQueue <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE</b></li> <li>• Kuyruk adı</li> </ul>	Bağlantı tüketicisinin sürekli abonelik iletileri aldığı kuyruğun adı.
brokerDurSubQueue <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>SYSTEM.JMS.D.SUBSCRIBER.QUEUE</b></li> <li>• Kuyruk adı</li> </ul>	Sürekli bir konu abonesinin iletileri aldığı kuyruğun adı. Ek bilgi için IBM MQ Explorer belgelerinde <b>BROKEDURRSUBQ</b> özelliğine bakın.

Çizelge 68. Konu nesnesine özgü özellikler (devamı var)			
Özellik adı	Tip	Geçerli değerler (koyu varsayılan değer)	Açıklama
brokerPubKuyruk <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>Ayarlanmadı</b></li> <li>• Kuyruk adı</li> </ul>	Yayınlanan iletilerin gönderildiği kuyruğun adı (akış kuyruğu). Bu özelliğin değeri, ConnectionFactory nesnesinin <b>brokerPubQueue</b> özelliğinin değerini geçersiz kılar. Ancak, bu özelliğin değerini ayarlamazsanız, bunun yerine ConnectionFactory nesnesinin <b>brokerPubQueue</b> özelliğinin değeri kullanılır.
brokerPubQueueManager <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• "" (<b>boş dizgi</b>)</li> <li>• Kuyruk yöneticisi adı</li> </ul>	Konu üzerinde yayınlanan iletilerin gönderildiği kuyruğun iyisi olan kuyruk yöneticisinin adı.
brokerVersion <sup>1</sup>	Dizgi	<ul style="list-style-type: none"> <li>• <b>Ayarlanmadı</b></li> <li>• 1</li> <li>• 2</li> </ul>	Kullanılmakta olan aracının sürümü. Bu özelliğin değeri, ConnectionFactory nesnesinin <b>brokerVersion</b> özelliğinin değerini geçersiz kılar. Ancak, bu özelliğin değerini ayarlamazsanız, bunun yerine ConnectionFactory nesnesinin <b>brokerVersion</b> özelliğinin değeri kullanılır.

#### Not:

1. Bu özellik, IBM WebSphere MQ classes for JMS ' un 7.0 sürümüyle kullanılabilir, ancak ConnectionFactory nesnesinin providerVersion özelliği 7 'den küçük bir sürüm numarasına ayarlanmadıkça IBM WebSphere MQ 7.0 kuyruk yöneticisine bağlı bir uygulamayı etkilemez.

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

UNIX, Linux, and Windows için Federal Bilgi İşleme Standartları (FIPS)

**yapılandırılması**

İstek iletileri bir MQRFH2 üstbilgisi içermediğinde yanıt iletilerine bir MQRFH2 üstbilgisi eklenebilmesi için bir etkinleştirme belirtimine ilişkin **targetClientMatching** özelliğini yapılandırabilirsiniz. Bu, bir uygulamanın yanıt iletileri üzerinde tanımladığı ileti özelliklerinin, ileti gönderildiğinde içerileceği anlamına gelir.

**Bu görev hakkında**

İletiyi yönlendirilen bir Bean (MDB) uygulaması, bir IBM MQ JCA kaynak bağdaştırıcısı etkinleştirme belirtimi aracılığıyla bir MQRFH2 üstbilgisi içermeyen iletileri tüketir ve daha sonra istek iletilerinin JMSReplyTo alanından oluşturulan JMS Hedefine yanıt iletileri gönderirse, istek iletileri olmasa bile yanıt iletilerinin bir MQRFH2 üstbilgisi içermesi gerekir. Ters durumda, uygulamanın bir yanıt iletileri üzerinde tanımladığı ileti özellikleri kaybedilir.

**targetClientMatching** özelliği, gelen bir iletilerin JMSReplyTo üstbilgi alanı tarafından tanımlanan kuyruğa gönderilen bir yanıt iletilerinin, yalnızca gelen iletilerin bir MQRFH2 üstbilgisi varsa MQRFH2 üstbilgisine sahip olup olmadığını tanımlar. Bu özelliği, bir etkinleştirme belirtimi için hem WebSphere Application Server traditional hem de WebSphere Liberty' da yapılandırabilirsiniz.

**targetClientMatching** özelliğinin değerini false olarak ayarladıysanız, bir MQRFH2 içermeyen bir istek iletilerinin JMSReplyTo üstbilgisinden yaratılmış bir JMS hedefine gönderilen bir yanıt iletilerine bir MQRFH2 üstbilgisi eklenebilir. Bunun nedeni, JMS Hedefindeki **targetClient** özelliğinin 0 değerine ayarlandığından, bu da iletilerin bir MQRFH2 üstbilgisi içermesi anlamına gelir. Giden iletide MQRFH2 üstbilgisinin varlığı, IBM MQ kuyruğuna gönderildiğinde, ileti üzerinde kullanıcı tanımlı ileti özelliklerinin saklamaya izin verir.

**targetClientMatching** özelliği true değerine ayarlıysa ve bir istek iletileri bir MQRFH2 üstbilgisi içermiyorsa, yanıt iletilerine bir MQRFH2 üstbilgisi eklenmez.

**Yordam**

- In WebSphere Application Server traditional, use the administration console to define the **targetClientMatching** property as a custom property on the IBM MQ activation specification:
  - a) Gezinme bölmesinde **Resources-> JMS-> Activation spesifikasyonları** seçeneğini tıklayın.
  - b) Görüntülemek ya da değiştirmek istediğiniz etkinleştirme belirtiminin adını seçin.
  - c) **Custom properties-> New** (Özel özellikler-> Yeni) öğelerini tıklayın ve yeni özel özelliğinin ayrıntılarını girin.  
Set the name of the property to `targetClientMatching`, the type to `java.lang.Boolean` and the value to `false`.
- WebSphere Liberty' ta, `server.xml` içindeki bir etkinleştirme belirtiminin tanımındaki **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 198](#)

Bir JMS uygulaması, Java Naming and Directory Interface (JNDI) ad alanından yönetilen nesnelere hedefler 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 tanıttıcısını (URI) kullanabilir.

[“Kaynak bağdaştırıcısının giden iletişim için yapılandırılması” sayfa 445](#)

Giden iletişim yapılandırmak için bir ConnectionFactory nesnesinin özelliklerini ve denetlenen bir hedef nesneyi tanımlayın.

### V 9.1.1 IBM MQ message-driven bean pause in WebSphere Liberty

Bir etkinleştirme belirtimine ilişkin **maxSequentialDeliveryFailures** özelliği, kaynak bağdaştırıcısının MDB ' yi duraklatmadan önce katlandığı, ileti odaklı bir bean (MDB) örneğine sıralı ileti teslimi hatası sayısı üst sınırını tanımlar.

## Başlamadan önce

You need to be aware of the set of events that might cause an MDB to pause in WebSphere Liberty. Kaynak bağdaştırıcısı, aşağıdakilerden herhangi birini ileti teslim hatası olarak kabul eder:

- MDB ' nin **onMessage** yönteminden denetlenmemiş bir kural dışı durum ortaya çıktı.
- İleti, MDB ' ye iletilmeden önce, kaynak bağdaştırıcısının işlenmesinde oluşan bir **JMSEException** .
- Kaynak bağdaştırıcısının işlenmesinde ortaya çıkan bir **JMSEException** , iletiyi MDB ' ye teslim eder.
- Geri döndürülemede olan iletiyi kullanmak için kullanılan XA hareketi ya da yerel hareket.
- İletiyi MDB ' ye teslim etmek için uygulama sunucusunda kullanılabilir iş parçacığı yok.

## Bu görev hakkında

**maxSequentialDeliveryFailures** özelliğinin varsayılan değeri -1'dir. Bu, MDB' nin hiçbir zaman duraklatılmadığını gösterir. Diğer herhangi bir negatif değer de -1 ile aynı işlem görür. Bir değer:

- 0 , ilk hatada MDB duraklatlarının olduğu anlamına gelir
- 1 , MDB ' nin iki sıralı hata üzerinde durakladığı anlamına gelir
- 2 , MDB ' nin üç sıralı hata üzerinde durakladığı ve bu şekilde devam eden

Bu özelliği bir etkinleştirme belirtimi için, yalnızca WebSphere Liberty'inde ve Liberty düzeyi 18.0.0.4 ya da daha yüksek bir belirtim için yapılandırabilirsiniz.



**Uyarı:** Özniteliği Liberty dışında herhangi bir uygulama sunucusu ortamında varsayılan olmayan bir değere ayarlıyorsanız, değer yoksayılır ve günlüğe yazılan bir uyarı iletisi olur.

Buna ek olarak, IBM MQ kaynak bağdaştırıcısını soysal bir kaynak bağdaştırıcısı olarak WebSphere Liberty içine kurmak da mümkündür. Bunu yapmak, tüm IBM MQ ve WebSphere Application Server bütünleştirme yeteneklerini devre dışı bırakır ve kaynak bağdaştırıcısının Liberty' ta çalıştığını algılayabilmesini önler. Bu nedenle, **maxSequentialDeliveryFailures** ayarının 0 değerinden büyük ya da bu değere eşit olması desteklenmez ve günlükteki bir uyarı iletisinde sonuçlar elde edilir.

## Yordam

- WebSphere Liberty' ta, `server.xml` içindeki bir etkinleştirme belirtiminin tanımındaki **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 445](#)

Giden iletişim yapılandırmak için bir ConnectionFactory nesnesinin özelliklerini ve denetlenen bir hedef nesneyi tanımlayın.

## Kaynak bağıdaştırıcısı kuruluşunun doğrulanması

IBM MQ kaynak bağıdaştırıcısı için kuruluş doğrulama sınaması (IVT) programı, bir EAR dosyası olarak sağlanır. Programı kullanmak için, programı konuşlandırmanız ve bazı nesnelere JCA kaynakları olarak tanımlamanız gerekir.

### Bu görev hakkında

Kuruluş doğrulama sınaması (IVT) programı, `wmq.jmsra.ivt.ear` adlı kurum arşivi (EAR) dosyası olarak sağlanır. This file is installed with IBM MQ classes for JMS in the same directory as the IBM MQ resource adapter RAR file, `wmq.jmsra.rar`. Bu dosyaların nereye takıldığı hakkında bilgi için bkz. ["IBM MQ kaynak bağıdaştırıcısının takılması"](#) sayfa 422.

Uygulama sunucunuzda IVT programını konuşlandırmanız gerekir. IVT programı, bir sunucu uygulamacığı ve bir iletinin IBM MQ kuyruğundan gönderilebileceğini ve bu kuyruktan alınabileceğini test eden bir MDB içerir. You can use the IVT program to verify that the IBM MQ resource adapter has been correctly configured to support distributed transactions. IBM MQ kaynak bağıdaştırıcısını IBM olmayan bir uygulama sunucusunda devreye alıyorsanız, IBM Hizmet, uygulama sunucunuzun doğru yapılandırıldığı doğrulamak için IVT 'nin çalıştığını göstermenizi isteyebilir.

IVT programını çalıştırabilmeniz için önce, bir ConnectionFactory nesnesi, bir Kuyruk nesnesi ve olasılıkla JCA kaynakları olarak bir Etkinleştirme Belirtimi nesnesi tanımlamanız ve uygulama sunucunuzun bu tanımlamalardan JMS nesnelere oluşturduğundan ve bunları bir JNDI ad alanına bağlamadığından emin olmanız gerekir. Kendi QueueManager'ın anasistem ve kapı ayarlarını eşleştirmek için nesnelere özelliklerini seçebilirsiniz, ancak aşağıdaki özellik 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 nesnelere tanımlamak için kullanılan düzenek, uygulama sunucunuza bağılı olarak değışir. Örneğin, bu özellikleri WebSphere Liberty'inde ayarlamak için, uygulama sunucusunun `server.xml` dosyasına aşağıdaki girdileri ekleyin:

```
<!-- 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, `jms/ivt/IVTCF` adı ve `jms/ivt/IVTQueue` adı ile bağı tanımlanacak bir Kuyruk nesnesinin bulunduğu JNDI ad alanında bir ConnectionFactory nesnesinin bağlanabilmesini bekler. Farklı adlar kullanabilirsiniz, ancak bunu yapmak için, IVT programının başlangıç sayfasındaki nesnelere adlarını girmeniz ve EAR dosyasını uygun bir şekilde değıştirmelisiniz.



IVT programını konuşturduktan ve uygulama sunucusu JMS nesnelerini oluşturduktan ve bunları JNDI ad alanına bağladıktan sonra, aşağıdaki adımları tamamlayarak IVT programını başlatabilirsiniz.

## Yordam

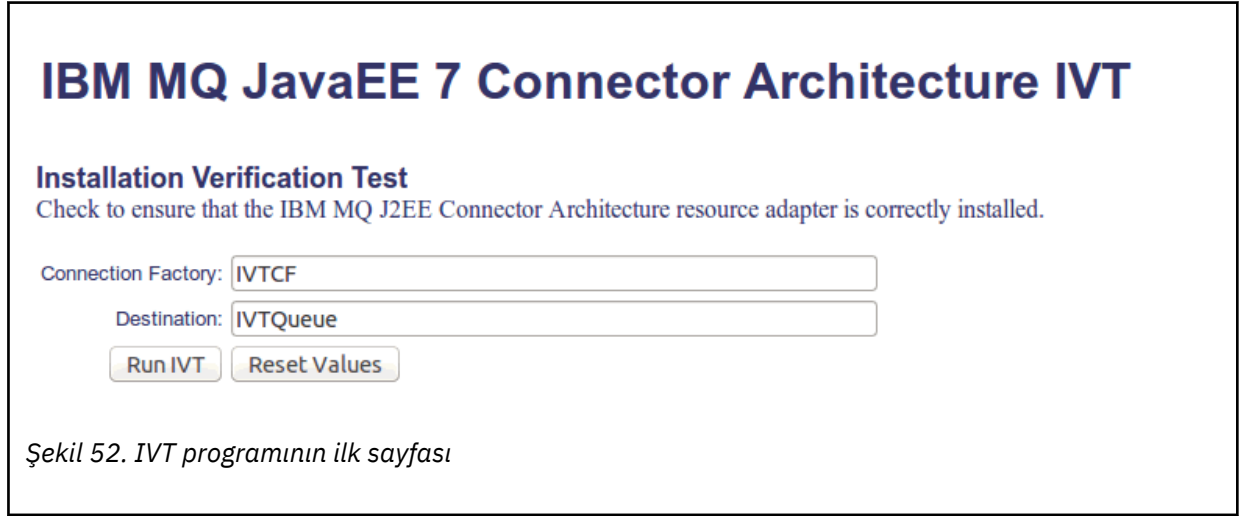
1. Web tarayıcınıza aşağıdaki biçimde bir URL adresi girerek IVT programını başlatın:

```
http://app_server_host: port/WMQ_IVT/
```

Burada *app\_server\_host* , uygulama sunucunuzun çalıştırıldığı sistemin IP adresi ya da anasistem adı ve *kayı* , uygulama sunucusunun dinlediği TCP kapısının numarasıdır. Örnek:

```
http://localhost:9080/WMQ_IVT/
```

Şekil 52 sayfa 465 , IVT programının başlangıç sayfasını gösterir.



**IBM MQ JavaEE 7 Connector Architecture IVT**

**Installation Verification Test**  
Check to ensure that the IBM MQ J2EE Connector Architecture resource adapter is correctly installed.

Connection Factory:

Destination:

Şekil 52. IVT programının ilk sayfası

2. Sınamayı çalıştırmak için **IVT Çalıştırdüğmesini** tıklayın.

Şekil 53 sayfa 466 , IVT başarılı olduğunda görüntülenen sayfayı gösterir.

# IBM MQ JavaEE 7 Connector Architecture IVT

## Running Installation Verification Test:

Using Connection Factory: *IVTCF*  
Using Destination: *IVTQueue*

Creating initial context...	☑
Looking up MQ Connection Factory...	☑
Looking up Destination...	☑
Creating connection...	☑
Starting connection...	☑
Creating session...	☑
Creating a temporary reply queue...	☑
Creating message consumer...	☑
Creating message producer...	☑
Creating message...	☑
Sending message to the MDB...	☑
Receiving response message from the MDB...	☑
Closing connection...	☑

## Installation Verification Test completed successfully!

[View Message Contents](#)

[Re-run Installation Verification Test](#)

Şekil 53. Başarılı bir IVT 'nin sonuçlarını gösteren sayfa

IVT başarısız olursa, Şekil 54 sayfa 466 içinde gösterilen bir sayfa görüntülenir. Hatanın nedenine ilişkin daha fazla bilgi edinmek için **Yığın İzlemesini Görüntüle**düğmesini tıklatın.

# IBM MQ JavaEE 7 Connector Architecture IVT

## Running Installation Verification Test:

Using Connection Factory: *IVTCF*  
Using Destination: *IVTQueue*

Creating initial context...	☑
Looking up MQ Connection Factory...	☑
Looking up Destination...	☑
Creating connection...	☑
Starting connection...	☑
Creating session...	☑
Creating a temporary reply queue...	☑
Creating message consumer...	☑
Creating message producer... failed to create message producer!	☒

## Installation Verification Test failed!

Error received - JMS Exception:

com.ibm.msg.client.jms.DetailedJMSSecurityException: JMSMQ2008: Failed to open MQ queue 'TEST.QUEUE'.  
JMS attempted to perform an MOOPEN, but IBM MQ reported an error.  
Use the linked exception to determine the cause of this error. Check that the specified queue and queue manager are defined correctly.

[View Stack Trace](#)

## Installation Verification Test failed!

[Retry Installation Verification Test](#)  
[Change IVT parameters](#)

Şekil 54. Başarısız bir IVT 'nin sonuçlarını gösteren sayfa

## Kaynak bağıdaştırıcısının GlassFish Server sunucusuna kurulması ve sınanması

IBM MQ kaynak bağıdaştırıcısını bir Windows işletim sisteminde GlassFish Server 'a kurmak için önce bir etki alanı yaratmalı ve başlatmalısınız. Daha sonra, kaynak bağıdaştırıcısını konuşlandırabilir ve yapılandırabilir ve kuruluş doğrulama sınaması (IVT) uygulamasını konuşlandırabilir ve çalıştırabilirsiniz.

### Bu görev hakkında

**Önemli:** Bu yönergeler, GlassFish Server sürüm 4 için geçerli olur.

Bu görev, çalışmakta olan bir GlassFish Server uygulama sunucusunu çalıştırdığınızı ve bunun için standart yönetim görevlerini aşına olduğunuzu varsayar. Bu görev ayrıca, yerel sisteminizde bir IBM MQ kuruluşuna sahip olduğunuzu ve standart yönetim görevlerini tanıdığınızı varsayar.

**Not:** Aşağıdaki görev adımlarının tamamlanması için, çalışan bir IBM MQ kurulumuna sahip olmanız gerekir. Bu kuruluş aşağıdaki nesnelere yapılandırmış olmalıdır:

- A queue manager called QM, that is started on port 1414, that uses channel SYSTEM.DEF.SVRCONN, and that connects using Client transport.
- A queue called Q1.

### Yordam

1. GlassFish Server **asadmin** kabuk programını başlatın.
  - a) Windows komut satırını açın ve *GlassFish/bin* dizinine gidin; burada *GlassFish* , GlassFish Server sürüm 4 'in kurulu olduğu dizindir.
  - b) Komut satırına **asadmin** komutunu girin.  
**asadmin** komutu, komut satırında yeni bir etki alanı yaratmanıza olanak tanıyan bir kabuk programı açar.
2. Bir etki alanı yaratın ve ardından bir etki alanı başlatın.
  - a) Yeni bir etki alanı yaratmak için kapı ve etki alanı adını belirterek **create-domain** komutunu kullanın. Komut satırına aşağıdaki komutu girin:

```
create-domain --adminport port domain_name
```

Burada *kapı* , kapı numarasıdır ve *etki\_alanı\_adi* , etki alanının kullanmasını istediğiniz addir.

**Not:** **create-domain** komutu, kendisiyle ilişkilendirilmiş birçok isteğe bağlı parametreye sahiptir. Ancak, bu görev için yalnızca `--adminport` parametresine gereksinim duyarsınız. Daha fazla bilgi için, GlassFish Server sürüm 4 için ürün belgelerine bakın.

Belirttiğiniz kapı kullanımdaysa, aşağıdaki ileti görüntülenir:

*domain\_name* kapısız için kapı kullanımda

Belirlediğiniz etki alanı adı kullanımdaysa, belirttiğiniz adın kullanımda olduğu ve şu anda kullanılamaz durumda olan tüm etki alanı adlarının bir listesi kullanımınıza ilişkin bir ileti alırsınız.

- b) Bir kullanıcı adı ve parola girdiğinizde, uygulama sunucusunda bir web tarayıcısı aracılığıyla oturum açmak için kullanılacak kimlik bilgilerini girin.

If the command completes successfully, a message summarizing the domain creation is displayed on the command line, including the message `Command create-domain executed successfully`.

Bir etki alanını başarıyla yaratmış olmanız.

- c) Komut satırına aşağıdaki komutu girerek etki alanınızı başlatın:

```
start-domain domain_name
```

Burada *domain\_name* , daha önce belirttiğiniz etki alanı adıdır.

3. GlassFish uygulama sunucusuna erişmek için bir web tarayıcısı kullanın.

a) Bir web tarayıcısının adres çubuğuna aşağıdaki komutu girin:

```
localhost:port
```

Burada *kayı* , etki alanınızı yaratırken daha önce belirttiğiniz kapıdır.

GlassFish Konsolu görüntülenir.

b) GlassFish Console yüklendiğinde ve sizden bir kullanıcı adı ve parola girmeniz istenirse, 2badımında belirttiğiniz kimlik bilgilerini girin.

4. Kaynak bağdaştırıcısını GlassFish Server 4 'e yükleyin.

a) On the toolbar **Ortak Görevler** select the **Uygulamalar** menu item to display the **Uygulamalar** page.

b) **Uygulamaları ya da Modülleri Konuşlandır** sayfasını açmak için **Konuşlandır** düğmesini tıklatın.

c) **Göz At** dosyasını tıklatın ve ardından `wmq.jmsra.rar` dosyasının konumuna gidin. Dosyayı seçin ve **Tamam** düğmesini tıklatın.

5. Bir bağlantı havuzu yaratın.

a) Araç çubuğunda, **Kaynaklar**' in altında, **Bağlayıcılar** menü öğesini seçin.

b) Daha sonra, **Bağlayıcı Bağlantı Havuzları** sayfasını açmak için **Bağlayıcı Bağlantı Havuzları** menü öğesini seçin.

c) **Yeni Bağlayıcı Bağlantısı Havuzu (Adım 1/2)** sayfasını açmak için **Yeni** düğmesini tıklatın.

d) **Yeni Bağlayıcı Bağlantısı Havuzu (Adım 1/2)** sayfasında, havuz adını `jms/ivt/IVTCF-Connection-Pool` olarak **Havuz Adı** alanına girin.

e) **Resource Adapter** (Kaynak Bağdaştırıcısı) alanında `wmq.jmsra` seçeneğini belirleyin.

f) **Bağlantı Tanımlaması** alanına `javax.jms.ConnectionFactory` girin.

g) **İleri**'yi seçin ve **Son**' u seçin.

6. Bağlayıcı kaynaklarını yaratın.

a) On the toolbar, under the **Bağlayıcılar** menu, select the **Bağlayıcı Kaynağı** option, to open the **Bağlayıcı Kaynakları** page.

b) **New Connector Resource** (Yeni Bağlayıcı Kaynağı) sayfasını açmak için **New** (Yeni) seçeneğini belirleyin.

c) **JNDI Name** (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 değerini girin:

- ad: `anasistem`; değer: `localhost`
- ad: `kayı`; 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 örnekte gösterilenlerden farklı olabilir.

- g) Araç çubuğunda, **Bağlayıcılar** altında, **Denetim Nesnesi Kaynakları** sayfasını açmak için **Denetim Nesnesi Kaynakları** menü öğesini seçin.
- h) **Yönetici Nesne Kaynakları** sayfasında, **Yeni Yönetici Nesnesi Kaynağı** sayfasını açmak için **Yeni** seçeneğini tıklatın.
- i) **JNDI Name** (JNDI Adı) alanına `IVTQueue` girin.
- j) **Resource Adapter** alanına `wmq.jmsra` girin.
- k) **Resource Type** (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 değerini girin:
- ad: ad; değer: `IVTQueue`
  - ad: `baseQueueManagerName`; değer: `QM`
  - ad: `baseQueueAd`; değer: `Q1`
- Not:** Kendi yapılandırma ayarlarınız için doğru değerleri kullandığınızdan emin olun, bu örnekte gösterilenlerden farklı olabilir.
- n) **Tamam**'ı tıklatın.
- o) **Etkin** onay kutusunu seçin ve **Etkinleştirdüğümü** tıklatın.
7. `wmq.jmsra.ivt.ear` EAR dosyasını GlassFish Server 'a konuşturun.
- a) **Uygulamalar** sayfasını görüntülemek için araç çubuğunda **Uygulamalar** seçeneğini tıklatın.
- b) IVT uygulamasını eklemek için **Deploy** (Konuştur) seçeneğini tıklatın.
- c) In the **Yer** field navigate to, and select, the `wmq.jmsra.ivt.ear`.
- 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 çubuğunda **Uygulamalar** seçeneğini tıklatın.
- b) Konuşturılan Uygulamalar tablosunda `wmq.jmsra.ivt` öğesini tıklatın.
- c) Modüller ve Bileşenler tablolarında **Başlat** düğmesini tıklatın.
- d) `http: link` ' i seçin.
- e) **IVT 'yi çalıştır** ' ı tıklatın.
- IVT programını başlattınız ve başarılı bir şekilde varsa, aşağıdaki çıkış görüntülenir:

## Running Installation Verification Test:

Using Connection Factory: *IVTCF*  
Using Destination: *IVTQueue*

Creating initial context...	✓
Looking up MQ Connection Factory...	✓
Looking up Destination...	✓
Creating connection...	✓
Starting connection...	✓
Creating session...	✓
Creating a temporary reply queue...	✓
Creating message consumer...	✓
Creating message producer...	✓
Creating message...	✓
Sending message to the MDB...	✓
Receiving response message from the MDB...	✓
Closing connection...	✓

## Installation Verification Test completed successfully!

[View Message Contents](#)

[Re-run Installation Verification Test](#)

Şekil 55. Başarılı IVT çıkışı

## Installing and testing the resource adapter in WildFly

IBM MQ kaynak bağıdaştırıcısını WildFly V10içine kuruyorsanız, önce IBM MQ kaynak bağıdaştırıcısı için bir altsistem tanımlaması eklemek üzere bazı yapılandırma dosyası değışiklikleri yapmanız gerekir. Daha sonra, kaynak bağıdaştırıcısını konuşlandırabilir ve kuruluş doęrulama sınaması (IVT) uygulamasını çalıřtırarak bu bağıdaştırıcıyı sınyarak sınebilirsiniz.

### Bu görev hakkında

**Önemli:** Bu yönergeler, WildFly V10için geçerli olur.

Bu görev, çalıřmakta olan bir WildFly uygulama sunucusunu çalıřtırdıęınızı ve bunun için standart denetim görevlerini ařına olduęunuzu varsayar. Bu görev ayrıca, IBM MQ kurulumuna sahip olduęunuzu ve standart yönetim görevlerini tanıdıęınızı varsayar.

### Yordam

1. ExampleQM adlı bir IBM MQ kuyruk yöneticisi yaratın ve [“Çoklu Platformlar üzerinde istemci baęlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102](#)içinde açıkladıęı şekilde ayarlayın.

Kuyruk yöneticisini ayarlarken ařaęıdaki noktalara dikkat edin:

- Dinleyici, kapı 1414 'te başlatılmalıdır.
- Kullanılacak kanalda SYSTEM.DEF.SVRCONN.
- IVT uygulaması tarafından kullanılan kuyruk TEST.QUEUE.

Ayrıca, bu uygulamanın geçici bir yanıt kuyruğu yaratabilmesi için, SYSTEM.DEFAULT.MODEL.QUEUE model kuyruğunun da DSP ve PUT yetkisi verilmesi gerekir.

2. *WildFly\_Home/standalone/configuration/standalone-full.xml* yapılandırma dosyasını 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. Deploy the resource adapter to your server by copying the *wmq.jmsra.rar* file into the directory *WildFly\_Home/standalone/deployments*.
4. Deploy the IVT application by copying the *wmq.jmsra.ivt.ear* file into the directory *WildFly\_Home/standalone/deployments*.
5. Bir komut istemi getirerek uygulama sunucusunu başlatın, *WildFly\_Home/bin* dizinine gidin ve komutu çalıştırın:

```
standalone.bat -c standalone-full.xml
```

6. IVT uygulamasını çalıştırın.

Daha fazla bilgi için bkz “Kaynak bağdaştırıcısı kuruluşunun doğrulanması” sayfa 464. WildFly için varsayılan URL şudur: [http://localhost:8080/WMQ\\_IVT/](http://localhost:8080/WMQ_IVT/).

## IBM MQ ve WebSphere Application Server ' in birlikte kullanılması

Through the IBM MQ messaging provider in WebSphere Application Server, Java Message Service (JMS) messaging applications can use your IBM MQ system as an external provider of JMS messaging resources.

### Bu görev hakkında

WebSphere Application Server altında çalışan Java yazılımında yazılan uygulamalar, ileti sistemini gerçekleştirmek için Java Messaging Service (JMS) belirtimini kullanabilir. Bu ortamdaki ileti alışverişi bir IBM MQ kuyruk yöneticisi tarafından sağlanabilir.

A benefit of using an IBM MQ queue manager is that connecting JMS applications can participate fully in the functionality of an IBM MQ network, which allows the applications to exchange messages with queue managers that are running on a multitude of platforms.

Uygulamalar, kuyruk bağlantısı üreticisi nesnesi için *istemci iletimi* ya da *bağ tanımları aktarımı* ' yı kullanabilir. Bağ tanımları aktarımı için, kuyruk yöneticisi, bağlantı gerektiren uygulamada yerel olarak varolmalıdır.

Varsayılan olarak, IBM MQ kuyruklarında tutulan JMS iletileri, JMS ileti üstbilgisi bilgilerinin bir kısmını tutmak için bir MQRFH2 üstbilgisini kullanır. Birçok eski IBM MQ uygulaması bu üstbilgileri içeren iletileri işleyemez ve kendi karakteristik üstbilgilerini (örneğin, CICS Bridge için MQCIH ya da IBM MQ Workflow uygulamaları için MQWIH) gerektiremez. Bu özel noktalar hakkında daha fazla bilgi için bkz. [JMS iletilerini IBM MQ iletilerine eşleme](#).

### İlgili görevler

[Configuring JMS resources in WebSphere Application Server](#)

[Uygulama sunucusunun en son kaynak bağdaştırıcısı bakım düzeyini kullanacak şekilde yapılandırılması](#)

## WebSphere Application Server ile IBM MQ komutunu kullanma

IBM MQ ve IBM MQ for z/OS , WebSphere Application Server ile birlikte verilen varsayılan ileti alışverişi sağlayıcısıyla birlikte kullanılabilir ya da buna alternatif olarak kullanılabilir.

IBM MQ ileti alışverişi sağlayıcısı, WebSphere Application Server ' nin bir parçası olarak kurulur. This includes a version of the IBM MQ resource adapter, and the IBM MQ Extended Transactional Client functionality, which allows the queue manager to participate in XA transactions managed by the application server. Kaynak bağdaştırıcısını kullanarak, ileti odaklı Bean 'ler etkinleştirme belirtimlerini ya da dinleyici kapılarını kullanacak şekilde yapılandırılabilir.

Uygulama sunucusunun desteklenmesini sağlamak için, IBM MQ kaynak bağdaştırıcısı kuruluş doğrulama test programı ' in uygulama sunucusunda konuşlandırılması ve başarıyla çalıştırılması gerekir. IBM MQ kaynak bağdaştırıcısı kuruluş doğrulama sınama programı başarıyla çalıştırdıktan sonra, IBM MQ kaynak bağdaştırıcısı desteklenen herhangi bir IBM MQ kuyruk yöneticisine bağlanabilir.

## JMS connections from WebSphere Application Server to IBM MQ

WebSphere Application Server ile kullanılacak IBM MQ düzeylerini dikkate almadan önce, uygulama sunucusu içinde çalışan Java Message Service (JMS) uygulamalarının IBM MQ kuyruk yöneticilerine nasıl bağlanacağını anlamak 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 görüntülerine takıldığına kullanılır. BAĞLAMALAR kipini kullanırken, iki ürün arasındaki tüm iletişim, Inter-Process Communication (IPC) kullanılarak yapılır.

IBM MQ ileti alışverişi sağlayıcısı, bir IBM MQ kuyruk yöneticisine bağlanmak için gereken yerel kitaplıkları BRETILER (Bağlamalar) kipinde içermiyor. BINDINGSbağ tanımlama kipi bağlantısı kullanmak için, IBM MQ , uygulama sunucusuyla aynı makineye kurulmalıdır ve kaynak bağıdaştırıcısının yerel kitaplık yolu, bu kitaplıkların bulunduğu IBM MQ dizinini gösterecek şekilde yapılandırılmalıdır. Ek bilgi için WebSphere Application Server ürün belgelerine bakın:

- WebSphere Application Server traditionaliçin bkz. [IBM MQ ileti alışverişi sağlayıcısını yerli kitaplıklarla yapılandırma](#).
- WebSphere Libertyiçin bakınız: [Deploying JMS applications to Liberty to use the IBM MQ messaging provider](#).

**z/OS** z/OSişletim sistemi üzerinde, bir WebSphere Application Server bağlantı üreticisini bağ tanımları kipinde bir IBM MQ kuyruğuna bağlamak istiyorsanız, WebSphere Application Server STEPLIB bitişirmesinde doğru IBM MQ kitaplıklarını belirtmeniz gerekir. Daha fazla bilgi için, WebSphere Application Server ürün belgelerindeki [IBM MQ libraries and the WebSphere Application Server for z/OS STEPLIB](#) belgesine bakın.

## CLIENT

İstemci iletimi, WebSphere Application Server ile IBM MQarasında iletişim kurmak için TCP/IP ' yi kullanır. Uygulama sunucusu ve kuyruk yöneticisi farklı makinelerde bulunuyorsa, aynı makineye ve işletim sistemi görüntülerine iki ürün takıldığına da CLIENT kipi de kullanılabilir.

JMS uygulamaları, BINDINGS\_THEN\_CLIENT taşıma tipini de belirtebilir. Bu iletim tipi kullanıldığında, uygulama ilk olarak BAĞYLER kipini kullanarak kuyruk yöneticisine bağlanmayı dener; bunu yapamazsa, CLIENT iletimini deneyecektir.

## How to find which version of the IBM MQ resource adapter is installed inside WebSphere Application Server

WebSphere Application Serveriçinde kurulu olan IBM MQ kaynak bağıdaştırıcısının hangi sürümü hakkında bilgi almak için bkz. teknik not [WebSphere MQ Resource Adapter \(RA\) ürününün hangi sürümü WebSphere Application Serverile birlikte teslim edilir?](#).

WebSphere Application Server ' in şu anda kullanmakta olduğu kaynak bağıdaştırıcısı düzeyini belirlemek için aşağıdaki Jython ve JAACL komutlarını kullanabilirsiniz:

### Jython

```
wmqInfoBeansUnsplit = AdminControl.queryNames("WebSphere:type=WMQInfo,*")
wmqInfoBeansSplit = AdminUtilities.convertToList(wmqInfoBeansUnsplit)
for wmqInfoMBean in wmqInfoBeansSplit: print wmqInfoMBean; print
AdminControl.invoke(wmqInfoMBean, 'getInfo', '')
```

**Not:** Bu komutu çalıştırmak için, bu komutu girdikten sonra iki kez **Return** (Geri Dön) seçeneğini tıklatmanız gerekir.

### JAACL

```
set wmqInfoBeans [$AdminControl queryNames WebSphere:type=WMQInfo,*]
foreach wmqInfoMBean $wmqInfoBeans {
  puts $wmqInfoMBean;
  puts [$AdminControl invoke $wmqInfoMBean getInfo [] []]
}
```

## Kaynak bağdaştırıcısının güncellenmesi

Updates to the IBM MQ resource adapter that is installed with the application server are included in WebSphere Application Server Fix Packs. **Kaynak bağdaştırıcısını güncelle ...**öğesini kullanarak IBM MQ kaynak bağdaştırıcısının güncellenmesi WebSphere Application Server Administrative Console (Yönetim Konsolu) içindeki tesis önerilmez; bu nedenle, WebSphere Application Server Fix Packs olanağında sağlanan güncellemelerin hiçbir etkisi olmaz.

## MQ\_INSTALL\_ROOT değişkeni

Before version 7.0, WebSphere Application Server could be configured to use the IBM WebSphere MQ classes for JMS located in an external IBM WebSphere MQ installation to connect to a queue manager by setting the WebSphere variable MQ\_INSTALL\_ROOT.

WebSphere Application Server 7.0' tan, MQ\_INSTALL\_ROOT yalnızca yerli kitaplıkları bulmak için kullanılır ve kaynak bağdaştırıcısında yapılandırılmış herhangi bir yerel kitaplık yolu tarafından geçersiz kılınır.

## WebSphere Application Server ile IBM MQ arasında bağlantı kuruluyor



### Uyarı:

1. Any supported version of WebSphere Application Server can use the IBM MQ resource adapter that is bundled with it, to connect to any supported version of IBM MQ.
2. Bağ tanımları kipi kullanılıyorsa, WebSphere Application Server içindeki bazı kitaplıkların, bağlantı kurducağı kuyruk yöneticisinin sürümüyle eşleşmesi gerekir:
  - WebSphere Application Server must be configured to load the native libraries provided with IBM MQ 9.1. Ek bilgi için [“Java Native Interface \(JNI\) kitaplıklarının yapılandırılması” sayfa 86](#) başlıklı konuya bakın.

- **z/OS** z/OS üzerinde, WebSphere Application Server STEPLIB bitişirmesinde doğru IBM MQ kitaplıklarını belirtmeniz gerekir.

Gereksinim duyduğunuz IBM MQ kitaplıklarına ilişkin ayrıntılar için bkz. [IBM MQ libraries and the WebSphere Application Server for z/OS STEPLIB](#) .

LINKLIST (LINKLST) içinde IBM MQ ' un bir sürümüne ilişkin kitaplıklarınız varsa, STEPLIB ile kitaplıkları geçersiz kılarak farklı bir IBM MQ sürümüne bağlanabilirsiniz.

3. IBM MQ Resource Adapter sürümü, kuyruk yöneticisi kuruluşu tarafından sağlanan yerel (paylaşılan) kitaplık sürümlerinden bağımsızdır.

For example, a WebSphere Application Server 8.5 with an IBM WebSphere MQ 7.1 Resource Adapter can still manage a bindings connection to an IBM MQ 9.0 queue manager using the IBM MQ 9.0 native libraries.

Daha fazla bilgi için bkz [“IBM MQ kaynak bağdaştırıcısı desteği bildirimi” sayfa 416](#).

Aşağıdaki tabloda, tüm WebSphere Application Serversürümlerinden IBM MQ ' e bağlanmak için hangi iletim tipleri kullanılabileceği gösterilmektedir.

IBM MQ ya da IBM WebSphere MQsürümü	BAĞLA	İSTEMCI
IBM MQ 9.1	Destekleniyor. <ul style="list-style-type: none"><li>• IBM MQ 9.1 must be installed on the same machine as the application server.</li><li>• WebSphere Application Server must be configured to load the</li></ul>	Destekleniyor

IBM MQ ya da IBM WebSphere MQsürümü	BAĞLA	İSTEMCI
	<p>native libraries provided with IBM MQ 9.1.</p> <ul style="list-style-type: none"> <li>▶ <b>z/OS</b> On z/OS, if you want to connect a WebSphere Application Server connection factory to an IBM MQ queue manger in bindings mode, the correct IBM MQ libraries must be specified in the WebSphere Application Server STEPLIB concatenation.</li> </ul>	
IBM MQ 9.0	<p>Destekleniyor.</p> <ul style="list-style-type: none"> <li>• IBM MQ 9.0 must be installed on the same machine as the application server.</li> <li>• WebSphere Application Server must be configured to load the native libraries provided with IBM MQ 9.0.</li> <li>▶ <b>z/OS</b> On z/OS, if you want to connect a WebSphere Application Server connection factory to an IBM MQ queue manger in bindings mode, the correct IBM MQ libraries must be specified in the WebSphere Application Server STEPLIB concatenation.</li> </ul>	Destekleniyor
IBM MQ 8.0	<p>Destekleniyor.</p> <ul style="list-style-type: none"> <li>• IBM MQ 8.0 must be installed on the same machine as the application server.</li> <li>• WebSphere Application Server must be configured to load the native libraries provided with IBM MQ 8.0.</li> <li>▶ <b>z/OS</b> On z/OS, if you want to connect a WebSphere Application Server connection factory to an IBM MQ queue manger in bindings mode, the correct IBM MQ libraries must be specified in the WebSphere Application Server STEPLIB concatenation.</li> </ul>	Destekleniyor
IBM WebSphere MQ 7.5	Destekleniyor.	Destekleniyor

IBM MQ ya da IBM WebSphere MQsürümü	BAĞLA	İSTEMCI
	<ul style="list-style-type: none"> <li>• IBM WebSphere MQ 7.5 must be installed on the same machine as the application server.</li> <li>• WebSphere Application Server must be configured to load the native libraries provided with IBM WebSphere MQ 7.5.</li> <li>• <b>z/OS</b> z/OSüzerinde, bir WebSphere Application Server bağlantı üreticisini bağ tanımları kipinde bir IBM WebSphere MQ kuyruk yöneticisine bağlamak istiyorsanız, WebSphere Application Server STEPLIB bitişirmesinde doğru IBM WebSphere MQ kitaplıklarının belirtilmesi gerekir.</li> </ul>	
IBM WebSphere MQ 7.1	<p>Destekleniyor.</p> <ul style="list-style-type: none"> <li>• IBM WebSphere MQ 7.1 must be installed on the same machine as the application server.</li> <li>• WebSphere Application Server must be configured to load the native libraries provided with IBM WebSphere MQ 7.1.</li> <li>• <b>z/OS</b> z/OSüzerinde, bir WebSphere Application Server bağlantı üreticisini bağ tanımları kipinde bir IBM WebSphere MQ kuyruk yöneticisine bağlamak istiyorsanız, WebSphere Application Server STEPLIB bitişirmesinde doğru IBM WebSphere MQ kitaplıklarının belirtilmesi gerekir.</li> </ul>	Destekleniyor

The following table shows the versions of WebSphere Application Server that the IBM MQ resource adapter is supported to run in.

IBM MQ kaynak bağdaştırıcısı sürümü	WebSphere Application Server hangi sürümü kaynak bağdaştırıcının bu sürümünü çalıştırıyor?
IBM MQ 9.1	<p>Kaynak bağdaştırıcısı aşağıdaki gibi çalışabilir:</p> <ul style="list-style-type: none"> <li>• Any Java EE 7 compliant version of WebSphere Liberty.</li> </ul>

IBM MQ kaynak bađdařtırıcısı sürümü	WebSphere Application Server hangi sürümü kaynak bađdařtırıcısının bu sürümünü alıřtırıyor?
	<ul style="list-style-type: none"> <li>WebSphere Application Server traditional 9.0.</li> </ul>
IBM MQ 9.0	Kaynak bađdařtırıcısı ařađıdaki gibi alıřabilir: <ul style="list-style-type: none"> <li>Any Java EE 7 compliant version of WebSphere Liberty.</li> <li>WebSphere Application Server traditional 9.0</li> </ul>
IBM MQ 8.0	Kaynak bađdařtırıcısı WebSphere Libertyyle uyumlu herhangi bir Java EE 7 uyumlu sürümünde alıřabilir IBM MQ 8.0 kaynak bađdařtırıcısı, WebSphere Application Server traditionalinde alıřmak üzere desteklenmiyor. The resource adapter already installed in WebSphere Application Server traditional should be used to connect to IBM MQ 8.0 queue managers.
IBM WebSphere MQ 7.5	Kaynak bađdařtırıcısı, J2EE 1.4 ya da üstü uyumlu uygulama sunucularında kullanılabilir. WebSphere Application Server 8.0 ve 7.0 ' te yer alan IBM WebSphere MQ kaynak bađdařtırıcısının sürümü bu ortamlarda kullanılmalıdır. IBM WebSphere MQ 7.5.0 Fix Pack 2 ve APAR IC92914 , kaynak bađdařtırıcısını WebSphere Liberty' e konuřlandırarak için destek ekler.
IBM WebSphere MQ 7.1	Kaynak bađdařtırıcısı, J2EE 1.4 ya da üstü uyumlu uygulama sunucularında kullanılabilir. WebSphere Application Server 8.0 ve 7.0 ' te yer alan IBM WebSphere MQ kaynak bađdařtırıcısının sürümü bu ortamlarda kullanılmalıdır.

### İlgili kavramlar

“IBM MQ kaynak bađdařtırıcısı desteđi bildirimi” sayfa 416

IBM MQ 8.0 ya da daha sonraki bir yayın düzeyiyle birlikte gelen kaynak bađdařtırıcısı, JMS 2.0 belirtimini uygular. Yalnızca Java Platform, Enterprise Edition 7 (Java EE 7) uyumlu bir uygulama sunucusuna konuřlandırılabilir ve bu nedenle JMS 2.0' ı destekler.

### İlgili bilgiler

[IBM MQ için Sistem Gereksinimleri](#)

## WebSphere Application Server ile IBM MQ arasında oluřturulan TCP/IP bađlantılarının sayısı belirleniyor

Paylařma etkileřimleri özelliđini kullanarak, birden ok etkileřim MQI kanalı yönetim ortamlarını paylařabilir; bu, TCP/IP bađlantısı olarak da bilinir.

### Bu görev hakkında

Applications running inside of WebSphere Application Server 7 and 8, that use IBM MQ messaging provider normal mode, will automatically use this feature. Bařka bir deyiřle, aynı uygulama sunucusu

yönetim ortamında çalışan, aynı IBM MQ kuyruk yöneticisine bağlanan birden çok uygulamanın, aynı kanal örneğini paylaşabildiği anlamına gelir.

Tek bir kanal yönetim ortamında paylaşılabilen etkileşimlerin sayısı IBM MQ kanal özelliği **SHARECNV** tarafından belirlenir. Sunucu bağlantı kanalları için bu özelliğin varsayılan değeri 10 'tır.

WebSphere Application Server 7 ve 8 tarafından yaratılan etkileşimlerin sayısına bakarak, yaratılan kanal eşgörünümlerinin sayısını saptamak mümkündür.

IBM MQ ileti alışverişi sağlayıcısı kipine ilişkin ek bilgi için [PROVIDERVERSION](#) normal kipi başlıklı konuya bakın.

### İlgili kavramlar

Paylaşımın paylaşılmasını kullanma

Sohbet paylaşımına izin verildiği bir ortamda, sohbetler bir MQI kanalı eşgörünümünü paylaşabilir.

“IBM MQ classes for JMS içinde bir TCP/IP bağlantısının paylaşılması” sayfa 294

Tek bir TCP/IP bağlantısını paylaşmak için bir MQI kanalının birden çok eşgörünümü yapılabilir.

### JMS bağlantı üreticileri

WebSphere Application Server içinde çalışan ve bağlantı ve oturum oluşturmak için IBM MQ ileti sistemi sağlayıcı bağlantı üreticisini kullanan, bağlantı üreticisinden oluşturulan her JMS bağlantısı için ve JMS bağlantısından oluşturulan her JMS oturumunda etkin sohbetler yapan uygulamalar.

### Bağlantı üreticisinden oluşturulan her JMS bağlantısı için bir etkileşim

Her JMS bağlantı üreticisi ilişkili bir bağlantı havuzuna sahiptir; bu havuz, serbest havuz ve etkin havuz olmak üzere iki bölüme ayrılır. Başlangıçta her iki havuz da boş olur.

Bir uygulama, bir bağlantı üreticisinden JMS bağlantısı oluşturduğunda, WebSphere Application Server boş havuzda bir JMS bağlantısı olup olmadığını denetler. Varsa, etkin havuza taşınır ve uygulamaya verilir. Ters durumda, yeni bir JMS bağlantısı yaratılır, etkin havuza girilir ve uygulamaya geri döndürülür. Bir bağlantı üreticisinden yaratılabilecek bağlantı sayısı üst sınırı, **Maximum connections** bağlantı üreticisi bağlantı havuzu özelliği tarafından belirtilir. Bu özellik için varsayılan değer 10 'dur.

Bir uygulama JMS bağlantısını bitirdikten ve kapattıktan sonra, etkin havuzdan yeniden kullanım için kullanılabilir durumda olan etkin havuzdan bağlantı taşınır. The connection pool property **Unused timeout** defines how long a JMS connection can stay in the free pool before it is disconnected. Bu özelliğe ilişkin varsayılan değer 1800 saniyedir (30 dakika).

Bir JMS bağlantısı ilk yaratıldığında, WebSphere Application Server ile IBM MQ arasında bir etkileşim başlatılır. Serbest havuza ilişkin **Unused timeout** özelliğinin değeri aşıldığında, bağlantı kapatılıncaya kadar etkileşim etkin kalır.

### Bir JMS bağlantısından oluşturulan her JMS oturumu için bir etkileşim

Bir IBM MQ ileti alışverişi sağlayıcı bağlantı üreticisinden yaratılan her JMS bağlantısı, ilişkili bir JMS oturum havuzuna sahiptir. Oturum havuzları, bağlantı havuzlarıyla aynı şekilde çalışır. The maximum number of JMS Sessions that can be created from a single JMS connection is determined by the connection factory session pool property **Maximum connections**. Bu özelliğin varsayılan değeri 10 'tır.

Bir etkileşim, bir JMS oturumu ilk yaratıldığında başlar, etkileşim JMS oturumu kapatılıncaya kadar etkin kalır. Bunun nedeni, oturum havuzuna ilişkin **Unused timeout** özelliğinin değerinden daha uzun bir süre serbest havuzda kaldığı için etkin kalmaya devam eder.

### SHARECNV özelliği için bir değer hesaplanıyor

Aşağıdaki formülü kullanarak tek bir bağlantı üreticisinden IBM MQ ' a yapılan etkileşim sayısı üst sınırını hesaplayabilirsiniz:

Maximum number of conversations =

```
connection Pool Maximum Connections +  
(connection Pool Maximum Connections * Session Pool Maximum Connections)
```

Bu sayıda sohbetin gerçekleşmesine izin vermek için oluşturulacak kanal eşgörünümlerinin sayısı, aşağıdaki hesaplamayı kullanarak çalışılabilir:

```
Maximum number of channel instances =  
Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplamadan kalan herhangi bir geri kalan kısmı yukarı yuvarlanabilir.

**Maximum connections** bağlantı havuzu 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 olan etkileşim sayısı üst sınırı:

```
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 IBM MQ 'e **SHARECNV** özelliği 10 değerine ayarlanmış bir kanal kullanılarak bağlanıyorsa, bu bağlantı üreticisi için yaratılacak kanal yönetim ortamı 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 belirtileri***

Bir etkinleştirme belirtimini kullanmak üzere yapılandırılan, ileti odaklı Bean uygulamaları, bir JMS hedefini izlemek için etkinleştirme belirtimine ilişkin etkin sohbetler ve iletileri işlemek için ileti odaklı bir bean örneğini çalıştırmak için kullanılan her sunucu oturumu için etkin sohbetler içerir.

Aşağıdaki etkileşimler, etkinleştirme belirtimini kullanmak üzere yapılandırılmış, iletilerle yönlendirilen Bean uygulamaları için etkindir:

- Uygun iletiler için bir JMS hedefini izlemek üzere etkinleştirme belirtimine ilişkin bir etkileşim. Bu etkileşim, etkinleştirme belirtimi başlatılmaz başlatılmaz ve etkinleştirme belirtimi duruncaya kadar etkin olmaya devam eder.
- İletileri işlemek için, ileti odaklı bir bean eşgörünümlerini çalıştırmak için 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 özelliğin varsayılan değeri 10 'dur. Sunucu oturumları, gerektiği şekilde oluşturulur ve etkinleştirme belirtimi gelişmiş özelliği **Server session pool timeout** tarafından belirtilen süre boyunca boşta durduysa, kapatılır. Bu özelliğe ilişkin varsayılan değer 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 'a kadar olan maksimum etkileşim sayısının aşağıdaki formül kullanılarak hesaplanabileceği anlamına gelir:

```
Maximum number of conversations = Maximum server sessions + 1
```

Bu sayıda diyaloga izin vermek için oluşturulan kanal eşgörünümlerinin sayısı, aşağıdaki hesaplamada bulunabilecek şekilde bulunabilir:

```
Maximum number of channel instances =  
Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplardan kalan herhangi bir geri kalan kısmı yukarı yuvarlanabilir.

For a simple activation specification, that uses the default value for the **Maximum server sessions** property, the maximum number of conversations that can exist between WebSphere Application Server and IBM MQ for this activation specification is calculated as:

```
Maximum number of conversations = Maximum server sessions + 1
```

Örneğin:

```
= 10 + 1  
= 11
```

If this activation specification is connecting to IBM MQ using a channel that has the **SHARECNV** property set to 10, then the number of channel instances that are created is calculated as:

```
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***

İletilerle yönlendirilen Bean uygulamaları tarafından kullanılan ASF kipinde çalışan dinleyici kapıları, her sunucu oturumu için etkileşim yaratır. Bir ileti, uygun iletiler için bir hedef izler ve başka bir ileti, iletileri işlemek için ileti odaklı bir bean somut örneği çalıştırır. Her dinleyici kapısı için yapılan etkileşimlerin sayısı, oturum sayısı üst sınırından hesaplanabilir.

Varsayılan olarak dinleyici kapıları, messages belirtiminin bir parçası olarak, uygulama sunucularının iletileri saptamak için kullanması gereken mekanizmayı tanımlayan ve bunları işlemek üzere ileti odaklı Bean 'lere teslim eden 1.1 belirtiminin bir parçası olarak çalıştırılır. Bu varsayılan işlem oluşturma kipinde dinleyici kapılarını kullanmak üzere ayarlanmış, iletilerle yönlendirilen Bean uygulamaları, sohbetler oluşturur:

#### **Dinleyici kapısı için uygun iletiler için bir hedefi izlemek üzere bir etkileşim**

Dinleyici kapıları bir JMS bağlantı üretici kullanacak şekilde yapılandırılıyor. Bir dinleyici kapısı başlatıldığında, bağlantı üreticisinin serbest havuzundan JMS bağlantısı için bir istek yapılır. Dinleyici kapısı durdurulduğunda, bağlantı serbest havuza döndürülür. For more information about how the connection pool is used, and how this affects the number of conversations to IBM MQ, see "[JMS bağlantı üreticileri](#)" sayfa 478.

#### **İletileri işlemek için, ileti odaklı bir bean eşgörünümlerini çalıştırmak için 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 oluşturulur ve dinleyici kapısının kullanmakta olduğu JMS bağlantısıyla ilişkili oturum havuzundan alınan JMS oturumlarını kullanır.

If a server session has been idle for the period of time specified by the Message Listener Service custom property **SERVER.SESSION.POOL.UNUSED.TIMEOUT**, the session is closed and the JMS session used is returned to the session pool free pool. JMS oturumu gerekinceye kadar oturum havuzu serbest havuzunda kalır ya da boş havuzda oturum havuzunun **Unused timeout** özelliğinin değerinden uzun süre boşta durduğu için kapatılır.

Oturum havuzunun nasıl kullanıldığı ve WebSphere Application Server ile IBM MQ arasındaki etkileşimlerin nasıl yönetilmesiyle ilgili daha fazla bilgi için bkz. "[JMS bağlantı üreticileri](#)" sayfa 478.

İleti Dinleyici Hizmeti özel özelliği **SERVER.SESSION.POOL.UNUSED.TIMEOUT**, WebSphere Application Server ürün belgelerinde [Dinleyici kapıları için sunucu oturum havuzlarının izlenmesi](#) konusuna bakın.

## Tek bir dinleyici kapısından IBM MQ' a kadar olan etkileşim sayısı üst sınırını hesaplama

Aşağıdaki formülü kullanarak, tek bir dinleyici kapısından IBM MQ ' a en fazla etkileşim sayısını hesaplayabilirsiniz:

```
Maximum number of conversations = Maximum sessions + 1
```

Bu sayıda sohbetin gerçekleşmesine izin vermek için oluşturulacak kanal eşgörünümlerinin sayısı, aşağıdaki hesaplamayı kullanarak çalışabilir:

```
Maximum number of channel instances =  
Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplardan kalan herhangi bir geri kalan kısmı yukarı yuvarlanabilir.

**Maximum sessions** özelliği için varsayılan değeri kullanan basit bir dinleyici kapısı için, bu dinleyici kapısı için WebSphere Application Server ile IBM MQ arasında var olan etkileşim sayısı üst sınırı aşağıdaki gibi hesaplanır:

```
Maximum number of conversations = Maximum sessions + 1
```

Örneğin:

```
= 10 + 1  
= 11
```

Bu dinleyici kapısı, **SHARECNV** özelliğine 10 değerine ayarlanmış bir kanal kullanılarak IBM MQ ' a bağlanıyorsa, yaratılacak kanal eşgörünümlerinin sayısı aşağıdaki gibi 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 olmayan) kipinde çalışan dinleyici kapıları**

ASF olmayan kipte çalışan dinleyici kapıları, sunucu oturumlarını kullanarak kuyruk hedefini ve konu hedefini izlemek üzere yapılandırılabilir. Sunucu oturumlarında birden çok etkileşim olabilir. Her bir durumda en çok kaç sayı hesaplanabilir.

Dinleyici kapıları, dinleyici kapılarının JMS hedeflerini izlemesine yol veren ASF olmayan kipte çalışacak şekilde yapılandırılabilir. İletilerle yönlendirilen Bean uygulamaları, ASF dışı bir işlem kipinde dinleyici kapıları kullanarak, iletileri işlemek için, ileti odaklı bir bean örneğini çalıştırmak için kullanılan her sunucu oturumu için bir etkileşim oluşturun. Dinleyici kapısı özelliği **oturum sayısı üst sınırı**, belirli bir dinleyici

kapısı için herhangi bir zamanda etkin olabilecek Sunucu Oturumları sayısı üst sınırını belirtir. Bu özelliğin varsayılan değeri 10 'tır.

ASF olmayan kipte çalışırken, bir kuyruk hedefini izleyen bir dinleyici kapısı otomatik olarak, dinleyici kapısı özelliği **Oturum sayısı üst sınırı** tarafından belirtilen Sunucu Oturumları sayısını otomatik olarak oluşturur. Bu Sunucu Oturumlarının tümü, dinleyici kapısının kullanmakta olduğu JMS Connection ile ilişkili oturum havuzundan alınan JMS Oturumlarının kullanımını sağlar ve uygun iletiler için sürekli olarak bir JMS Hedefi 'ni izler.

Dinleyici kapısı bir konu hedefini izlenecek şekilde yapılandırıldıysa, **Oturum sayısı üst sınırı** değeri yoksayılır ve tek bir sunucu oturumu kullanılır.

Bir dinleyici kapısı tarafından ASF dışı kipte çalışan bir dinleyici kapısı tarafından kullanılan Sunucu Oturumları, dinleyici kapısı duruncaya kadar etkin kalır. Bu noktada, kullanılan JMS Oturumları, dinleyici kapısının kullandığı JMS Connection için oturum havuzu Serbest Havuzuna döndürülür.

Oturum havuzunun nasıl kullanıldığı ve WebSphere Application Server ile IBM MQ arasındaki etkileşimlerin nasıl yönetilmesiyle ilgili daha fazla bilgi için bkz. ["JMS bağlantı üreticileri" sayfa 478.](#)

ASF ve ASF dışı kipte WebSphere Application Server ile işlem yapma ve dinleyici kapılarının ASF olmayan kipi kullanacak şekilde nasıl yapılandırılacağı hakkında daha fazla bilgi için bkz. [ASF kipindeki ve ASF olmayan kipte ileti işleme.](#)

## Bir kuyruk hedefini izlerken etkileşim sayısı üst sınırının hesaplanması

The maximum number of conversations from a single listener port, running in non-ASF mode and monitoring a queue destination to IBM MQ can be calculated using the following formula:

```
Maximum number of conversations = Maximum sessions
```

Bu sayıda sohbetin gerçekleşmesine olanak tanımak için yaratılacak kanal eşgörünümlerinin sayısı, aşağıdaki hesaplama kullanılarak bulunabilir:

```
Maximum number of channel instances =  
Maximum number of conversations / SHARECNV for the channel being used
```

Bu hesaplama kalan herhangi bir geri kalan kısmı yukarı yuvarlanabilir.

For a simple listener port running in non-ASF mode that is using the default value for the **Oturum Üst Sınırı** property and monitoring a queue destination, the maximum number of conversations that can exist between WebSphere Application Server and IBM MQ for this listener port is:

```
Maximum number of conversations = Maximum sessions
```

Örneğin:

```
= 10
```

Bu dinleyici kapısı, **SHARECNV** özelliğine 10 değerine ayarlanmış bir kanal kullanılarak IBM MQ 'a bağlanıyorsa, yaratılan kanal yönetim ortamlarının sayısı aşağıdaki gibi 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 sınırının hesaplanması

ASF olmayan kipte çalışan ve bir konu hedefini izlemek üzere yapılandırılmış bir dinleyici kapısı için, dinleyici kapısındaki etkileşim sayısı IBM MQ olarak tanımlanır:

```
Maximum number of conversations = 1
```

Bu sayıda sohbetin gerçekleşmesine olanak tanımak için yaratılacak kanal eşgörünümlerinin 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 geri kalan kısmı yukarı yuvarlanabilir.

**Oturum 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 olan etkileşim sayısı üst sınırı şudur:

```
Maximum number of conversations = Maximum sessions
```

Örneğin:

```
= 10
```

Bu dinleyici kapısı, **SHARECNV** özelliğine 10 değerine ayarlanmış bir kanal kullanılarak IBM MQ 'a bağlanıyorsa, yaratılan kanal yönetim ortamlarının sayısı aşağıdaki gibi hesaplanır:

```
Maximum number of channel instances =  
Maximum number of conversations / SHARECNV for the channel being used
```

Örneğin:

```
= 10 / 10  
= 1
```

## Configuring authentication aliases to secure WebSphere Application Server connection to IBM MQ

Authentication aliases map to a user name and password combination that can be used to secure WebSphere Application Server connection to IBM MQ. Bir kimlik doğrulama diğer adı ile bir bağlantı üreticini yapılandırabilirsiniz.

### ***Kurumsal uygulamalarla kimlik doğrulama diğer adlarının kullanılması***

When an enterprise application running inside of WebSphere Application Server attempts to create a JMS connection to IBM MQ, the application looks up an IBM MQ messaging provider connection factory definition from the Java Naming Directory Interface (JNDI) repository of the application server.

When the IBM MQ messaging provider connection factory definition is located from within the JNDI repository of the application server, one of the following methods is called:

- `ConnectionFactory.createConnection()`
- `ConnectionFactory.createConnection(String username, String password)`

Bağlantı üreticisi tanımlanmış bir J2C kimlik doğrulama diğer adı ile yapılandırıldıysa, kimlik doğrulama diğer adındaki kullanıcı adı ve parola, bağlantı oluşturmak için bağlantı üreticisi kullanıldığında IBM MQ değerine aktarılabilir.

## Bağlantı üreticileri ve kimlik doğrulama diğer adları

IBM MQ ileti alışverişi sağlayıcı bağlantı üreticileri, IBM MQ kuyruk yöneticilerine nasıl bağlanabilmeye ilişkin bilgi içerir. Enterprise applications running inside of WebSphere Application Server can use the connection factories to create JMS connections to IBM MQ.

WebSphere Application Server , bağlantı fabrikaları tanımlamalarını JNDI kullanılarak erişilebilecek bir havuzda saklar. Bir bağlantı üreticisi yaratıldığında, bağlantı üreticisine, tanımlandığını uygulama sunucusu kapsamında (Hücre, Düğüm ya da Sunucu kapsamı) benzersiz olarak tanıtmak için bir JNDI adı verilir.

Örneğin, WebSphere Application Server Cell kapsamında tanımlanan bir IBM MQ ileti alışverişi sağlayıcısı bağlantı üreticisi, BYYT iletimi kullanılarak kuyruk yöneticisine (myQM) nasıl bağlanılacağına ilişkin bilgiler içerir. This connection factory is given the JNDI name jms/myCF to uniquely identify it.

Bağlantı üreticileri bir kimlik doğrulama diğer adını kullanacak şekilde de yapılandırılabilir. Kimlik doğrulama diğer adları, bir kullanıcı adı ve parola birleşimiyle eşlenir. Bağlantı üreticinin nasıl kullanılsa bağlı olarak, kimlik doğrulama diğer adı içindeki kullanıcı adı ve parola JMS bağlantısı yaratıldığında IBM MQ ' a aktarılabilir ya da olmayabilir.

**Önemli:** IBM MQ 8.0 öncesinde, varsayılan IBM MQ Object Authority Manager (OAM) bir yetki denetimi gerçekleştirdiğinde, bir bağlantı yapıldığında kullanıcı adının IBM MQ' a iletildiğinden emin olmak için, kuyruk yöneticisine erişme yetkisi edinmiş olmalıdır.

Belirtilen parolayı doğrulamak için herhangi bir denetim yapılmadı. Bir kimlik doğrulama denetimi gerçekleştirmek ve kullanıcı kimliği ve parola eşleşmesini doğrulamak için bir IBM MQ kanal güvenlik çıkışı yazmanız gerekir. Bunun nasıl yapabileceğiyle ilgili ayrıntılı bilgi için [Kanal güvenlik çıkış programları](#) nda bulabilirsiniz.

IBM MQ 8.0' tan, kuyruk yöneticisi, kullanıcı adına ek olarak parolayı denetler.

## Bağlantı üreticisini kullanma

Aşağıdaki konularda, doğrudan ve dolaylı görünüşleri kullanarak bağlantı üreticisini kullanmaya ilişkin bilgiler bulunur:

- [“Doğrudan arama yoluyla bağlantı üreticisini kullanma” sayfa 487](#)
- [“Dolaylı arama yoluyla bağlantı üreticisini kullanma” sayfa 488](#)

## İSTEMCELERİN

İstemci iletimi kullanacak şekilde yapılandırılmış bağlantı üreticileri, kuyruk yöneticisine bağlanmak için hangi IBM MQ sunucusu bağlantı kanalını (SVRCONN) kullanacaklarını belirtmelidir.

If the IBM MQ channel agent user identifier (MCAUSER) property remains blank for the channel that the connection factory has been configured to use, then the connection factory can be used with either a direct look up, or indirect look up.

MCAUSER özelliği bir kullanıcı tanımlayıcısı olarak ayarlandıysa, bu kullanıcı kimliği, kurumsal uygulamanın doğrudan ya da dolaylı bir görünüm kullanıp kullanmamasından bağımsız olarak, bağlantı üreticisi IBM MQ ile bağlantı yaratmak için kullanıldığında IBM MQ ' a aktarılır.

## Özet tablolar

The following tables summarize what user identifiers are flowed down to IBM MQ when the BINDINGS transport, and the CLIENT transport, respectively are used:

Çizelge 69. BAĞLA

Yapılandırma	Uygulama çağrısı <b>ConnectionFactory.createC onnection()</b>	Uygulama çağrısı <b>ConnectionFactory.createC onnection(String username, String password)</b>
Uygulamanın konuşlandırma tanımlayıcısı, bağlantı üreticisi için bir Kaynak Başvurusu içermiyor	The user identifier for the application server process is flowed down to IBM MQ.	ConnectionFactory.createC onnection(String username, String password) yöntemine geçirilen kullanıcı kimliği ve parola, IBM MQ' a aktarılır.
Uygulamanın konuşlandırma tanımlayıcısı, bağlantı üreticisine ilişkin bir Kaynak Başvurusu içeriyor ve <b>res-auth</b> özelliği "Uygulama" olarak ayarlanıyor	The user identifier for the application server process is flowed down to IBM MQ.	ConnectionFactory.createC onnection(String username, String password) yöntemine geçirilen kullanıcı kimliği ve parola, IBM MQ' a aktarılır.
Uygulamanın konuşlandırma tanımlayıcısı, bağlantı üreticisine ilişkin bir Kaynak Başvurusu içeriyor ve <b>res-auth</b> özelliği "Kapsayıcı" olarak ayarlandı	Bağlantı üreticisine ilişkin kimlik doğrulama diğer adı altında belirtilen kullanıcı kimliği ve parola, IBM MQ' ye aktılır.	Bağlantı üreticisine ilişkin kimlik doğrulama diğer adı altında belirtilen kullanıcı kimliği ve parola, IBM MQ' ye aktılır.
Uygulamanın konuşlandırma tanımlayıcısı, <b>res-auth</b> özelliği "Taşıyıcı" olarak ayarlanmış olan bağlantı üreticisi için bir Kaynak Başvurusu içeriyor ve uygulama bir kimlik doğrulama diğer adı ile yapılandırıldı	Uygulamanın kullanmak üzere yapılandırıldığı kimlik doğrulama diğer adı altında belirtilen kullanıcı kimliği ve parola, IBM MQ' a aktılır.	Uygulamanın kullanmak üzere yapılandırıldığı kimlik doğrulama diğer adı altında belirtilen kullanıcı kimliği ve parola, IBM MQ' a aktılır.

Çizelge 70. CLIENT kipi

Yapılandırma	Uygulama çağrısı <b>ConnectionFactory.createC onnection()</b>	Uygulama çağrısı <b>ConnectionFactory.createC onnection(String username, String password)</b>
Uygulamanın konuşlandırma tanımlayıcısı, bağlantı üreticisi için bir Kaynak Başvurusu içermiyor ve bağlantı üreticisi, MCAUSER özelliği ayarlanmamış bir IBM MQ kanalını kullanacak şekilde yapılandırılmış	The user identifier for the application server process is flowed down to IBM MQ.	ConnectionFactory.createC onnection(String username, String password) yöntemine geçirilen kullanıcı kimliği ve parola, IBM MQ' a 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 yapılandırılmış	Bağlantı üreticisini kullanmak üzere yapılandırıldığı IBM MQ kanalındaki MCAUSER özelliği tarafından belirtilen kullanıcı kimliği, IBM MQdeğerine aktarılarak aktarılır.	Bağlantı üreticisini kullanmak üzere yapılandırıldığı IBM MQ kanalındaki MCAUSER özelliği tarafından belirtilen kullanıcı kimliği, IBM MQdeğerine aktarılarak aktarılır.

Çizelge 70. CLIENT kipi (devamı var)

Yapılandırma	Uygulama çağrısı <b>ConnectionFactory.createC onnection()</b>	Uygulama çağrısı <b>ConnectionFactory.createC onnection(String username, String password)</b>
Application's deployment descriptor contains a Resource Reference for the connection factory which has the <b>res-auth</b> property is set to <i>Uygulama</i> and the connection factory is configured to use an IBM MQ channel that has the MCAUSER property unset	The user identifier for the application server process is flowed down to IBM MQ.	ConnectionFactory.createC onnection(String username, String password) yöntemine geçirilen kullanıcı kimliği ve parola, IBM MQ' a aktarılır.
Application's deployment descriptor contains a Resource Reference for the connection factory which has the <b>res-auth</b> property is set to <i>Uygulama</i> and the connection factory is configured to use an IBM MQ channel that has the MCAUSER property set to a user identifier	Bağlantı üreticinin kullanmak üzere yapılandırıldığı IBM MQ kanalındaki MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' a aktılır.	Bağlantı üreticinin kullanmak üzere yapılandırıldığı IBM MQ kanalındaki MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' a aktılır.
Uygulamanın konuşlandırma tanımlayıcısı, <b>res-auth</b> özelliği " <i>Taşıyıcı</i> olarak ayarlanmış olan bağlantı üreticisi için bir Kaynak Başvurusu içeriyor ve bağlantı üreticisi, MCAUSER özelliği ayarlanmamış bir IBM MQ kanalını kullanacak şekilde yapılandırıldı	Bağlantı üreticisine ilişkin kimlik doğrulama diğer adı altında belirtilen kullanıcı kimliği ve parola, IBM MQ' ye aktılır.	Bağlantı üreticisine ilişkin kimlik doğrulama diğer adı altında belirtilen kullanıcı kimliği ve parola, IBM MQ' ye aktılır.
Uygulamanın konuşlandırma tanımlayıcısı, <b>res-auth</b> özelliği " <i>Taşıyıcı</i> olarak ayarlanmış olan bağlantı üreticisi için bir Kaynak Başvurusu içeriyor ve bağlantı üreticisi, MCAUSER özelliği bir kullanıcı kimliğine ayarlı olan bir IBM MQ kanalı kullanacak şekilde yapılandırıldı	Bağlantı üreticinin kullanmak üzere yapılandırıldığı IBM MQ kanalındaki MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' a aktılır.	Bağlantı üreticinin kullanmak üzere yapılandırıldığı IBM MQ kanalındaki MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' a aktılır.
Uygulamanın konuşlandırma tanımlayıcısı, <b>res-auth</b> özelliği " <i>Taşıyıcı</i> olarak ayarlanmış olan bağlantı üreticisi için bir Kaynak Başvurusu içeriyor ve uygulama bir kimlik doğrulama diğer adı ile yapılandırıldı ve bağlantı üreticisi, MCAUSER özelliği ayarlanmamış bir IBM MQ kanalını kullanacak şekilde yapılandırıldı	Uygulamanın kullanmak üzere yapılandırıldığı kimlik doğrulama diğer adı altında belirtilen kullanıcı kimliği ve parola, IBM MQ' a aktılır.	Uygulamanın kullanmak üzere yapılandırıldığı kimlik doğrulama diğer adı altında belirtilen kullanıcı kimliği ve parola, IBM MQ' a aktılır.

Çizelge 70. CLIENT kipi (devamı var)

Yapılandırma	Uygulama çağrısı <b>ConnectionFactory.createC onnection()</b>	Uygulama çağrısı <b>ConnectionFactory.createC onnection(String username, String password)</b>
Uygulamanın konuşlandırma tanımlayıcısı, <b>res-auth</b> özelliği taşıyıcı olarak ayarlanmış olan bağlantı üreticisi için bir Kaynak Başvurusu içeriyor ve uygulama bir kimlik doğrulama diğer adı ile yapılandırıldı ve bağlantı üreticisi, MCAUSER kullanıcı kimliğine ayarlanmış bir IBM MQ kanalını kullanacak şekilde yapılandırıldı	Bağlantı üreticinin kullanmak üzere yapılandırıldığı IBM MQ kanalındaki MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' a aktılır.	Bağlantı üreticinin kullanmak üzere yapılandırıldığı IBM MQ kanalındaki MCAUSER özelliği tarafından belirtilen kullanıcı kimliği IBM MQ' a aktılır.

### **Doğrudan arama yoluyla bağlantı üreticisini kullanma**

Bir IBM MQ ileti alışıverşi sağlayıcısı bağlantı üreticisi tanımlandıktan sonra, bir kurum uygulaması bağlantı üreticisi tanımlamasını artabilir ve bir IBM MQ kuyruk yöneticisine JMS bağlantısı yaratmak için bu tanımlamayı kullanabilir. Bu, doğrudan bir görünüm üzerinden yapılabilir.

Bir kurumsal uygulama, doğrudan arama kullanmak için, aşağıdaki yöntem çağrısını yaparak uygulama sunucusunun JNDI havuzuna bağlanır:

```
InitialContext ctx = new InitialContext();
```

Kuruluş, JNDI havuzuna bağlandıktan sonra, bağlantı üreticisinin JNDI adını kullanarak bağlantı üreticisi tanımlamasını tanıtır ve aşağıdaki gibi olur:

```
ConnectionFactory cf = (ConnectionFactory) ctx.lookup("jms/myCF");
```

#### **Notlar:**

- Uygulama geliştiricinizin, kurumsal uygulama geliştirildiğinde gerekli bağlantı üreticisinin JNDI adını bilmesi gerekir. Because the JNDI name is hard coded inside the application, if the JNDI name changes, you need to re-write and re-deploy the application.
- Bu şekilde bir bağlantı üreticisi tanımlaması kullanıldığında, kimlik doğrulama diğer adında belirtilen kullanıcı adı ve parola (bağlantı üreticisinde kullanılmak üzere yapılandırılmıştır) IBM MQ' a aktarılmaz. Bu, yetkisiz uygulamaların bağlantı üreticisini tanımlanmasını ve güvenli IBM MQ sistemlerine bağlanmak için kullanabilmelerini engellemesidir.

IBM MQ ' a aktılan kullanıcı adı ve parola, bağlantı üreticisinden JMS bağlantısını yaratmak için kullanılan yöntemle bağlıdır.

Bir uygulama yöntemi kullanarak bir JMS bağlantısı oluşturursa:

```
ConnectionFactory.createConnection()
```

Varsayılan kullanıcı kimliği IBM MQ' a iletilir. Bu, kurum uygulamasının çalıştığı uygulama sunucusunu başlatan kullanıcı adı ve paroladır.

Diğer bir seçenek olarak, bir uygulama, yöntemi çağırarak bir JMS bağlantısı yaratabilir:

```
ConnectionFactory.createConnection(String username, String password)
```

Bir uygulama, bir bağlantı üreticisinin doğrudan görünüşü gerçekleştirdiyse ve bu yöntemi çağırdıysa, `createConnection()` yöntemine geçirilen kullanıcı adı ve parola, IBM MQ' e aktarılır.

**Önemli:** IBM MQ 8.0öncesinde, IBM MQ bir yetki denetimi işlediğinden, yalnızca aşağı doğru aktı olan kullanıcı adının kuyruk yöneticisine erişme yetkisi olduğunu doğrulamıştır.

Parolada herhangi bir denetim yapılmadı. Bir kimlik doğrulama denetimi gerçekleştirmek ve kullanıcı adının ve parolanın geçerli olduğunu doğrulamak için bir IBM MQ kanal güvenlik çıkışı yazılmalıdır. Bunun nasıl yapabileceğiyle ilgili ayrıntılı bilgi için [Kanal güvenlik çıkış programları](#)' nda bulabilirsiniz.

IBM MQ 8.0' tan, kuyruk yöneticisi, kullanıcı adına ek olarak parolayı denetler.

### ***Dolaylı arama yoluyla bağlantı üreticisini kullanma***

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**

Rather than directly looking up the connection factory using `java:comp/env/myCF`, an enterprise application contains a resource reference has the local JNDI name of: `java:comp/env/myResourceReferenceCF`.

To use this JNDI name, the application connects to the JNDI repository of the application server, in the same way as if the application is performing a direct look up:

```
InitialContext ctx = new InitialContext();
```

Rather than identifying `java:comp/env/myCF` directly, the application now identifies the JNDI name of the resource reference:

```
ConnectionFactory cf = (ConnectionFactory) ctx.lookup("java:comp/env/jms/myResourceReferenceCF");
```

Uygulama sunucusuna, kurumsal uygulamanın dolaylı bir görünüm gerçekleştirdiğini söylemek için yerel JNDI adı için `java:comp/env` önekinde gereksinim duyarsınız.

Uygulama konuşlandırıldığında, kullanıcı, `java:comp/env/myResourceReferenceCF` kaynak başvurusunun JNDI adını, uygulamanın önceden oluşturduğu bağlantı üreticisinin JNDI adına eşler: `java:comp/env/myCF`.

Uygulama çalıştırıldığında, uygulama sunucusununontoile eşlendiği yerel JNDI adını kullanan bir JMS bağlantı üreticisi arar: `java:comp/env/myCF`. Bu bağlantı üreticisi daha sonra, uygulama tarafından IBM MQile bağlantı oluşturmak için kullanılır.

### **Kimlik doğrulama diğer adları ve dolaylı aramalar**

Bir kaynak başvurusu, sağlanan bağlantı üreticisinin davranışını değiştiren ek özelliklerin tanımlanmasına da olanak sağlar. Bir kaynak başvurusunun özelliklerinden biri **res-auth**. Bu özelliğin değeri, kurumsal uygulamanın, kaynak başvurusu eşlemelerinin IBM MQ ile bağlantı yaratırken (bir kimlik doğrulama diğer adı tanımlandıysa) ya da uygulamanın kendi kullanıcı adını ve parolasını belirtiyorsa, bağlantı üreticisinin kimlik doğrulama diğer adını kullanıp kullanmayacağını belirtir.

Bu özelliğin varsayılan değeri *Uygulama*' dir. This means that the user name and password that are flowed down to the queue manager, when a JMS connection is created, is determined by the application itself. Kaynak başvurusu eşlemelerinin olduğu bağlantı üreticisinin kimlik doğrulama diğer adı kullanılmıyor.

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 ayarlandıysa, varsayılan kullanıcı kimliği aşağıya IBM MQ olarak aktarılır. 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 ayarlandıysa, yönteme geçirilen kullanıcı adı ve parola IBM MQ' a gönderilir.

Bir bağlantı oluştururken kaynak başvuru eşlemlerinin eşlendiği bağlantı üreticisinde tanımlanan kimlik doğrulama diğer adını kullanmak için, **res-auth** özelliğini *Taşıyıcı* değerine ayarlamanız gerekir. Bir uygulama JMS bağlantısı yarattığında, `createConnection` çağrısı bir kullanıcı adı ve parola belirtse de kimlik doğrulama diğer adı ayrıntıları kullanılır.

## Dolaylı arama kullanılırken kimlik doğrulama diğer adını geçersiz kılma

Bir uygulama, **res-auth** özelliği *Taşıyıcı* olarak ayarlanmış bir kaynak başvurusu kullanıyorsa, JMS bağlantıları oluşturulduğunda kullanılan kimlik doğrulama diğer adını geçersiz kılabilirsiniz.

To override the authentication alias, the resource reference needs to include an extra property called **authDataAlias**, that maps to an existing authentication alias that has already been created in the application server environment into which the application will be deployed. Bu özelliği, IBM tarafından sağlanan Rational araçları kullanılarak yaratılan herhangi bir kaynak başvuruunda belirtebilirsiniz.

Bu yöntemi kullanarak, dolaylı olarak aranmış bir JMS bağlantı üreticisi kullanıyorsanız, farklı bir kimlik doğrulama diğer adı kullanabilirsiniz. Belirtilen kimlik doğrulama diğer adı yoksa, kurumsal uygulama kurulduktan sonra yeni bir ad belirlenebilir. Ek bilgi için WebSphere Application Server ürün belgelerindeki *Resource references* başlıklı konuya bakın.

### WebSphere Application Server 8.5 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 kümeleri kullanılırken ileti odaklı Bean 'ler için iş yükü dengelemesi

Bir WebSphere Application Server 7.0 ve 8.0 kümesinde konuşlandırılan ve IBM WebSphere MQ ileti alışverişi sağlayıcısı normal kipinde çalışacak şekilde yapılandırılmış ileti odaklı Bean uygulamaları kullanılırken, küme üyelerinden biri iletilerin çoğunluğunu işliyor. İletilerin işlenmesini birden çok küme üyesinde dağıtmak için küme üyelerinin iş yükünü dengeleyebilirsiniz.

IBM WebSphere MQ 7.0 introduced a new feature called **Asynchronous consume**, which allows applications to consume messages asynchronously from a queue using APIs called **MQCB** and **MQCTL**.

Message driven bean applications running inside of WebSphere Application Server 7.0 and 8.0, that use IBM WebSphere MQ messaging provider normal mode will automatically make use of this feature. When the applications start up, they will set up an asynchronous consumer on the JMS destination that they have been configured to monitor by calling **MQCB**. The **MQCTL** API is then called to indicate that the application is ready to receive messages from the JMS destination.

İletiyi yönlendirilen Bean uygulamaları bir WebSphere Application Server kümesine konuşlandırıldığında, her küme üyesi, ileti odaklı bean 'in iletiler için izlediği JMS hedefi için zamanuyumsuz bir tüketici ayarlar. JMS hedefini barındıran IBM WebSphere MQ 7.0 kuyruk yöneticisi, süreç için JMS hedefinde uygun bir ileti olduğunda küme üyeye bildirilmekten sorumludur.

IBM WebSphere MQ 7.0.1 Fix Pack 6 öncesinde kuyruk yöneticileri, JMS Hedefinde zamanuyumsuz tüketicisini ayarlayacak ilk küme üyesini kabul eder. Bu küme üyesi, JMS hedefine uygun bir ileti geldiğinde bildirim gönderilecek ilk üye olacaktır. Daha sonra, iletiyi yönlendirilen Bean uygulamasını başlatmak için ilk küme üyesi, JMS hedefine gelen uygun iletilerin çoğunluğunu işleyecek.

WebSphere Application Server bir IBM WebSphere MQ 7.0.1 Fix Pack 6 ya da daha sonraki bir kuyruk yöneticisine bağlandığında, bir JMS hedefine gelen iletiler, o JMS hedefine kaydedilmiş olan tüm zamanuyumsuz tüketicilere eşit olarak dağıtılır. Bir WebSphere Application Server 7.0 ve IBM MQ 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** özelliğini yapılandırma

## IBM MQ Üstbilgileri paketini kullanma

IBM MQ Üstbilgileri paketi, bir iletinin IBM MQ üstbilgilerini işlemek için kullanabileceğiniz bir yardımcı arabirimler ve sınıflar kümesi sağlar. Genellikle, komut sunucusunu kullanarak (Programlanabilir Komut Biçimi (PCF) iletileri kullanarak) yönetim hizmetleri gerçekleştirmek istediğinizde IBM MQ Üstbilgileri paketini kullanırsınız.

### Bu görev hakkında

IBM MQ Üstbilgileri 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' nin her ikisi için de kullanabilirsiniz:

- IBM MQ classes for Java (ayrıca IBM MQ Base Java olarak da adlandırılır).
- IBM MQ classes for Java Message Service (IBM MQ classes for JMS, also referred to as IBM MQ JMS).

IBM MQ Base Java applications typically manipulate MQMessage objects, and the Headers support classes can directly interact with these objects, since they natively understand the IBM MQ Base Java interfaces.

IBM MQ JMS' ta, bir iletinin bilgi yükü genellikle bir Dize ya da bayt dizisi nesnesidir ve bu nesne `DataInput` ve `DataOutput` akışlarıyla işlenebilir. IBM MQ Üstbilgileri paketi, bu veri akışlarıyla etkileşim kurmak için kullanılabilir ve IBM MQ JMS uygulamaları tarafından gönderilen ve alınan tüm MQ iletilerini işleme almak için uygundur.

Bu nedenle, IBM MQ Üstbilgileri paketi IBM MQ Base Java paketine başvurular içerse de, bu paket IBM MQ JMS uygulamaları içinde de kullanılmak üzere tasarlanmıştır ve Java Platform, Enterprise Edition (Java EE) ortamlarında kullanıma uygundur.

IBM MQ Üstbilgileri paketini kullanabilmenin tipik bir yolu, örneğin, denetim iletilerini Programlanabilir Komut Biçiminde (PCF) işletebilmenize yardımcı olur. Örneğin, aşağıdaki herhangi bir nedenden dolayı:

- Bir IBM MQ kaynağına ilişkin ayrıntılara erişmek için.
- Bir kuyruğun derinliğini izlemek için.
- Bir kuyruğa erişimi engellemek için.

By using PCF messages with the IBM MQ JMS API, this kind of administration of application-centric resources can be performed from within Java EE applications without having to resort to using the IBM MQ Base Java API.

### Yordam

- IBM MQ classes for Java ile ilgili ileti üstbilgilerini işlemek üzere IBM MQ Headers paketini kullanmak için bkz. [“IBM MQ classes for Java ile kullanma” sayfa 490.](#)
- IBM MQ classes for JMS ile ilgili ileti üstbilgilerini işlemek üzere IBM MQ Headers paketini kullanmak için bkz. [“IBM MQ classes for JMS ile kullanma” sayfa 491.](#)

## IBM MQ classes for Java ile kullanma

IBM MQ classes for Java uygulamaları IBM MQ classes for Java arabirimlerini yerel olarak anladıkları için, genellikle `MQMessage` nesnelerini yönlendirir ve bu nesnelerle Üstbilgiler destek sınıfları doğrudan bu nesnelerle etkileşimde bulunabilir.

## Bu görev hakkında

IBM MQ provides some sample applications that demonstrate how to use the IBM MQ Headers package with the IBM MQ Base Java API (IBM MQ classes for Java).

Örneklerde iki şey var:

- Bir denetim işlemini gerçekleştirmek ve yanıt iletisini ayrıştırmak için bir PCF iletisinin nasıl yaratılacağı.
- IBM MQ classes for Javakomutunu kullanarak bu PCF iletisinin nasıl gönderileceği.

Kullandığınız platforma bağlı olarak bu örnekler, IBM MQ kurulumunuzun `samples` ya da `tools` dizininde bulunan `pcf` dizini altında kurulur (bkz. [“IBM MQ classes for Java için kuruluş dizinleri” sayfa 328](#)).

## Yordam

1. Bir denetim işlemi gerçekleştirmek için bir PCF iletisi yaratın ve yanıt iletisini ayrıştırın.
2. IBM MQ classes for Javakomutunu kullanarak bu PCF iletisini gönderin.

## İlgili kavramlar

[“IBM MQ ileti üstbilgilerinin IBM MQ classes for Java ile işlenmesi” sayfa 354](#)

Farklı ileti üstbilgisi tiplerini gösteren Java sınıfları sağlanır. İki yardımcı sınıf da sağlanır.

[“PCF iletilerinin IBM MQ classes for Java ile işlenmesi” sayfa 359](#)

Java sınıfları PCF tarafından yapılandırılmış iletiler yaratmak ve ayrıştırmak ve PCF isteklerinin gönderilmesini kolaylaştırmak ve PCF yanıtlarını toplamak için sağlanır.

## IBM MQ classes for JMS ile kullanma

IBM MQ Üstbilgilerini IBM MQ classes for JMS ile birlikte kullanmak için, IBM MQ classes for Java ile aynı temel adımları gerçekleştirdiniz. The PCF message can be created and the response parsed in exactly the same way by using the IBM MQ Headers package and the same sample code as for IBM MQ classes for Java.

## Bu görev hakkında

IBM MQ API 'yı kullanarak bir PCF iletisi göndermek için, ileti bilgi yükü bir JMS Bytes Message 'a yazılmalı ve standart JMS API 'ları kullanılarak gönderilmelidir. Tek dikkat edilmesi gereken, iletinin bir JMS RFH2 ya da MQMD ' deki belirli değerleri içeren başka bir üstbilgi içermemesi gerekir.

Bir PCF iletisi göndermek için aşağıdaki adımları tamamlayın. PCF iletisinin yaratıldığı yol ve bilgiler yanıt iletisinden çıkarıldığında, IBM MQ classes for Java ile aynı şekilde bulunur (bkz. [“IBM MQ classes for Java ile kullanma” sayfa 490](#)).

## Yordam

1. SYSTEM.ADMIN.COMMAND.QUEUE' i temsil eden bir JMS Kuyruk Hedefi oluşturun.

IBM MQ JMS uygulamaları, PCF iletilerini SYSTEM.ADMIN.COMMAND.QUEUE(Kuyruk) ve bu kuyruğu gösteren bir JMS Hedefi (JMS Hedef) nesnesine erişmeniz gerekir. Hedef, aşağıdaki özellikler kümesine sahip olmalıdır:

```
WMQ_MQMD_WRITE_ENABLED = YES
WMQ_MESSAGE_BODY = MQ
```

WebSphere Application Server kullanıyorsanız, bu özellikleri hedefte özel özellikler olarak tanımlamanız gerekir.

Hedef programsal 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 deęerlerini ieren bir JMS Baytları iletisine dnştrn.

Bir JMS Byte iletisi yaratılmalıdır ve PCF iletisi yazılmalıdır. Bir yanıt kuyruęu yaratılması gerekiyor, ancak bunun belirli bir ayarı olmaması gerekiyor.

Ařaęıdaki rnek kod paracıęı, JMS Byte İletisi nasıl yaratılacaęı ve bir com.ibm.mq.headers.pcf.PCFMessage nesnesi nasıl yazılacaęı gsterilmektedir. PCFMessage nesnesi (pcfCmd) daha nce IBM MQ Headers paketi kullanılarak oluřturulmuř. (PCFMessage 'ı yklemek iin paketin com.ibm.mq.headers.pcf.PCFMessageolduęunu 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 gnderin ve standart JMS API ' larını kullanarak yanıtı alın.

## 4. Yanıt iletisini iřlenmek zere bir PCF iletisine dnştrn.

Yanıt iletisini almak ve bunu bir PCF iletisi olarak iřlemek iin ařaęıdaki 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 iletisi” sayfa 128

JMS iletileri bir stbilginin, zelliklerden ve bir gvdeden oluřur. JMS , beř ileti gvdesi tipini tanımlar.

IBM i

## Setting up IBM MQ on IBM i with Java and JMS

This collection of topics gives an overview of how you set up and test IBM MQ with Java and JMS on IBM i using CL commands or the qshell environment.

**Not:** IBM MQ 8.0, ldap.jar, jndi.jar ve jta.jar , JDK ' nin bir parasıdır.

### CL komutlarının kullanılması

Ayarladığınız CLASSPATH, MQ temel Java, JMS ieren JMS ve JNDI iermeyen JMS sınımaları iindir.

/home/Userprofile dizininiz altında bir .profile dosyası kullanmayacaksanız, aşağıdaki sistem düzeyi ortam değişkenlerini ayarlamamız gerekir. You can check to see if they are set using the **WRKENVVAR** command.

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 ayarlanmadıysa, aşağıdaki işlemi gerçekleştirin:

```
ADDENVVAR ENVVAR(CLASSPATH)
VALUE('.:QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar
:QIBM/ProdData/mqm/java/lib/connector.jar:QIBM/ProdData/mqm/java/lib
:QIBM/ProdData/mqm/java/samples/base
:QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar
:QIBM/ProdData/mqm/java/lib/jms.jar
:QIBM/ProdData/mqm/java/lib/providerutil.jar
:QIBM/ProdData/mqm/java/lib/fscontext.jar:') LEVEL(*SYS)
```

4. QIBM\_MULTI\_YIVCID ayarlanmadıysa, aşağıdaki komutu verin:

```
ADDENVVAR ENVVAR(QIBM_MULTI_THREADED) VALUE('Y') LEVEL(*SYS)
```

5. QIBM\_USE\_DESCRIPTOR\_STDIO ayarlanmamış ise, şu komutu verin:

```
ADDENVVAR ENVVAR(QIBM_USE_DESCRIPTOR_STDIO) VALUE('I') LEVEL(*SYS)
```

6. QSH\_REDIRECTION\_TEXTDATA ayarlanmadıysa, aşağıdaki komutu verin:

```
ADDENVVAR ENVVAR(QSH_REDIRECTION_TEXTDATA) VALUE('Y') LEVEL(*SYS)
```

## Qshell Ortamının Kullanılması

If you use the QSHELL environment, you can set up a .profile in your /home/Userprofile directory. Ek bilgi için, Qshell Interpreter (qsh) belgelerine bakın.

.profile' da aşağıdakileri belirtin. CLASSPATH deyiminin tek bir satırda olması ya da \ karakterini kullanarak farklı hatlara ayrılmış olması gerektiğini unutmayın.

```
CLASSPATH=.:QIBM/ProdData/mqm/java/lib/com.ibm.mq.jar: \
/QIBM/ProdData/mqm/java/lib/connector.jar: \
/QIBM/ProdData/mqm/java/lib: \
/QIBM/ProdData/mqm/java/samples/base: \
/QIBM/ProdData/mqm/java/lib/com.ibm.mqjms.jar: \
/QIBM/ProdData/mqm/java/lib/jms.jar: \
/QIBM/ProdData/mqm/java/lib/providerutil.jar: \
/QIBM/ProdData/mqm/java/lib/fscontext.jar:
HOME=/home/XXXXX
LOGNAME=XXXXX
PATH=/usr/bin:
QIBM_MULTI_THREADED=Y QIBM_USE_DESCRIPTOR_STDIO=I
QSH_REDIRECTION_TEXTDATA=Y
TERMINAL_TYPE=5250
```

QMQMJAVA kitaplığının, **DPLIBL** komutunu vererek kitaplık listesinde yer aldığından emin olun.

Listede QMQMJAVA kitaplığı yoksa, şu komutu kullanarak ekleyin: **ADDLIBLE LIB (QMQMJAVA)**

## 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 sinama 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 ögesini 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(tm) Message Service 5.300
Installation Verification Test

Creating a QueueConnectionFactory
Creating a Connection
Creating a Session
Creating a Queue
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
JMSType: null
JMSDeliveryMode: 2
JMSExpiration: 0
JMSPriority: 4
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f94040403ccf041f0000c012
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. **WRKMQLSR** komutunu vererek, dinleyicinin doğru kapı için başlatıldığını doğrulayın.
4. **STRQSH** komutunu vererek komut satırından qshell ögesini 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(tm) Message Service 5.300  
Installation Verification Test
```

```
Creating a QueueConnectionFactory  
Creating a Connection  
Creating a Session  
Creating a Queue  
Creating a QueueSender  
Creating a QueueReceiver  
Creating a TextMessage  
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE  
Reading the message back again
```

```
Got message:  
JMS Message class: jms_text  
JMSType: null  
JMSDeliveryMode: 2  
JMSExpiration: 0  
JMSPriority: 4  
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f94040403ccf041f0000d012  
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:QMOM  
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 sınıyor

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 Server için 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 ögesini 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 nesnelere (*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(tm) Message Service Administration

InitCtx>
InitCtx>
InitCtx>
InitCtx>
InitCtx>
InitCtx>
Stopping MQSeries classes for Java(tm) Message Service Administration

+ Administration done; tidying up files
+ Done!
$
> IVTRun -url file:///tmp/mqjms -icf com.sun.jndi.fscontext.RefFSContextFactory

5724-H72 (c) Copyright IBM Corp. 2011, 2024. All Rights Reserved.
MQSeries classes for Java(tm) Message Service
Installation Verification Test

Using administered objects, please ensure that these are available
```

```
Retrieving a QueueConnectionFactory from JNDI
Creating a Connection
Creating a Session
Retrieving a Queue from JNDI
Creating a QueueSender
Creating a QueueReceiver
Creating a TextMessage
Sending the message to SYSTEM.DEFAULT.LOCAL.QUEUE
Reading the message back again

Got message:
JMS Message class: jms_text
JMSType: null
JMSDeliveryMode: 2
JMSExpiration: 0
JMSPriority: 4
JMSMessageID: ID:c1d4d840c5d3c3d9e3d7f1f94040403ccf041f0000e012
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
$
```

## Bir Maven havuzu kullanarak Java uygulama geliřtirmesi

When developing a Java application for IBM MQ, by using a Maven repository to automatically install dependencies, you do not need to explicitly install anything before using IBM MQ interfaces.

### Maven Merkezi Havuzu

Maven, uygulamaları oluşturmak için kullanılan bir araçtır ve ayrıca, uygulamanızın erişmek isteyebileceği yapay nesnelere tutmak için bir havuz sağlar.

Maven Havuzu (ya da Merkezi Havuz), JAR dosyaları gibi dosyaların, bilinen bir adlandırma mekanizmasıyla kolayca keşfedilen farklı sürümlere sahip olmasını sağlayan bir yapıya sahiptir. Daha sonra oluşturma araçları, uygulamanızın bağımlılıklarını dinamik olarak çekmek için bu adları kullanabilir. Uygulamanızın tanımlamasında, Maven 'i bir oluşturma aracı olarak kullanırken, POM dosyası olarak adlandırıldığında, yalnızca bağımlılıkları ve oluşturma süreci orada ne yapacağınızı bilir.

### IBM MQ istemci dosyaları

Copies of the IBM MQ Java client interfaces are available in the Central Repository under the `com.ibm.mq` GroupId. Hem `com.ibm.mq.allclient.jar` 'ı (genellikle bağımsız programlar için kullanılır) hem de `wmq.jmsra.rar` 'ı (Java EE uygulama sunucularında kullanım için) bulabilirsiniz. `allclient.jar`, hem IBM MQ classes for JMS hem de IBM MQ classes for Java' yi içerir.

**Önemli:** IBM MQ yeniden yerelleştirilebilir JAR dosyasını içeren bir uygulama oluşturmak için Apache Maven Assembly Plugin *jar-with-dependencies* biçiminin kullanılması desteklenmez.

Maven komutu tarafından işlenen bir `pom.xml` dosyasında, aşağıdaki örneklerde gösterildiği gibi, bu JAR dosyaları için bağımlılıklar ekliyorsunuz:

- 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>9.1.3.0</version>
</dependency>
```

- Java EE kaynak bağıdaştırıcısını kullanmak için:

```
<dependency>
<groupId>com.ibm.mq</groupId>
<artifactId>wmq.jmsra</artifactId>
<version>9.1.3.0</version>
</dependency>
```

For an example of a simple project in Eclipse to run a JMS project, see the IBM Developer article [Developing Java applications for MQ just got easier with Maven](#).

## C++ uygulamaları geliştirilmesi

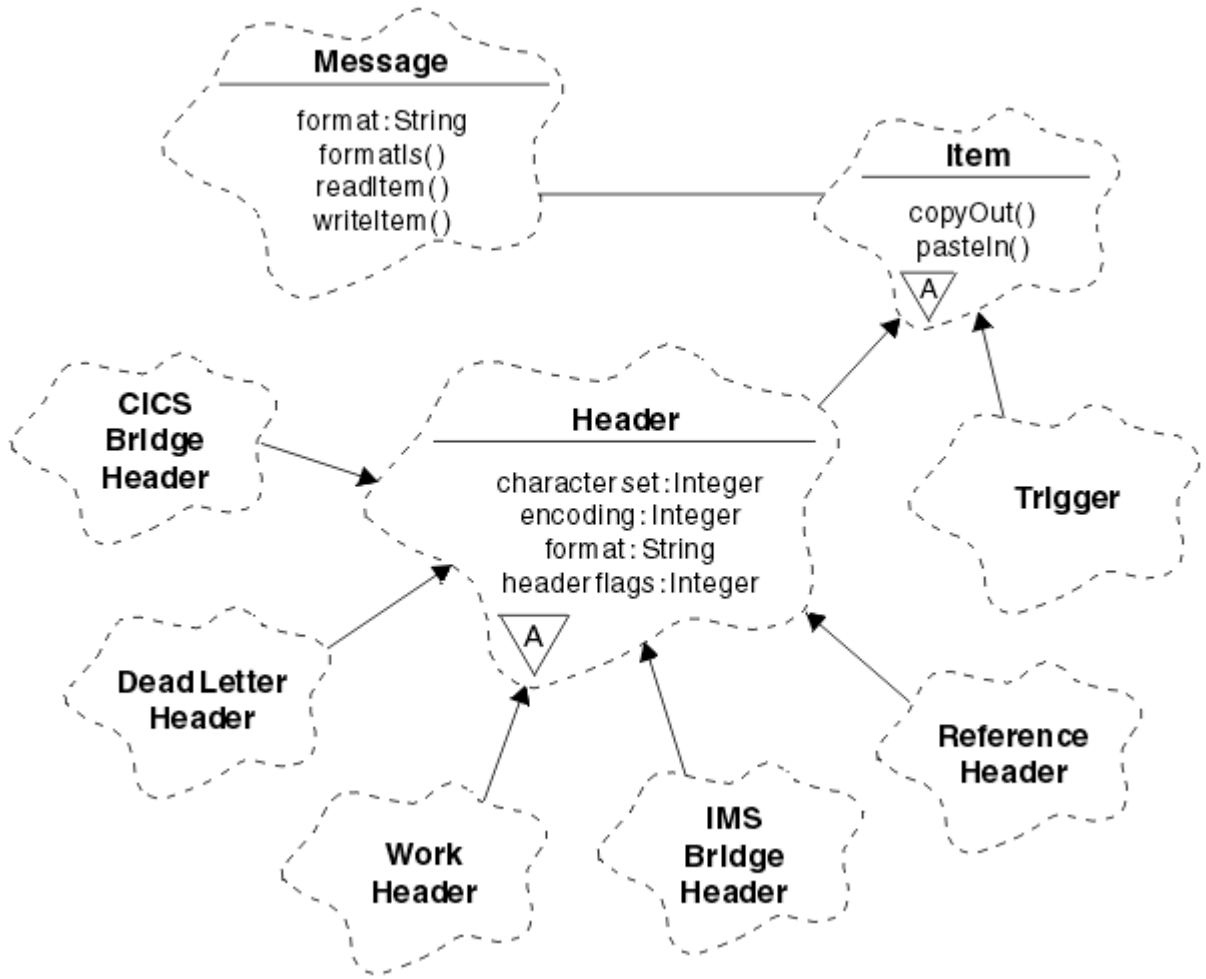
IBM MQ provides C++ classes equivalent to IBM MQ objects and some additional classes equivalent to the array data types. Bu, MQI aracılığıyla olmayan bazı özellikleri sağlar.

IBM WebSphere MQ 7.0, IBM MQ programlama arabirimlerinde yapılan geliştirmeler C++ sınıflarına uygulanmaz.

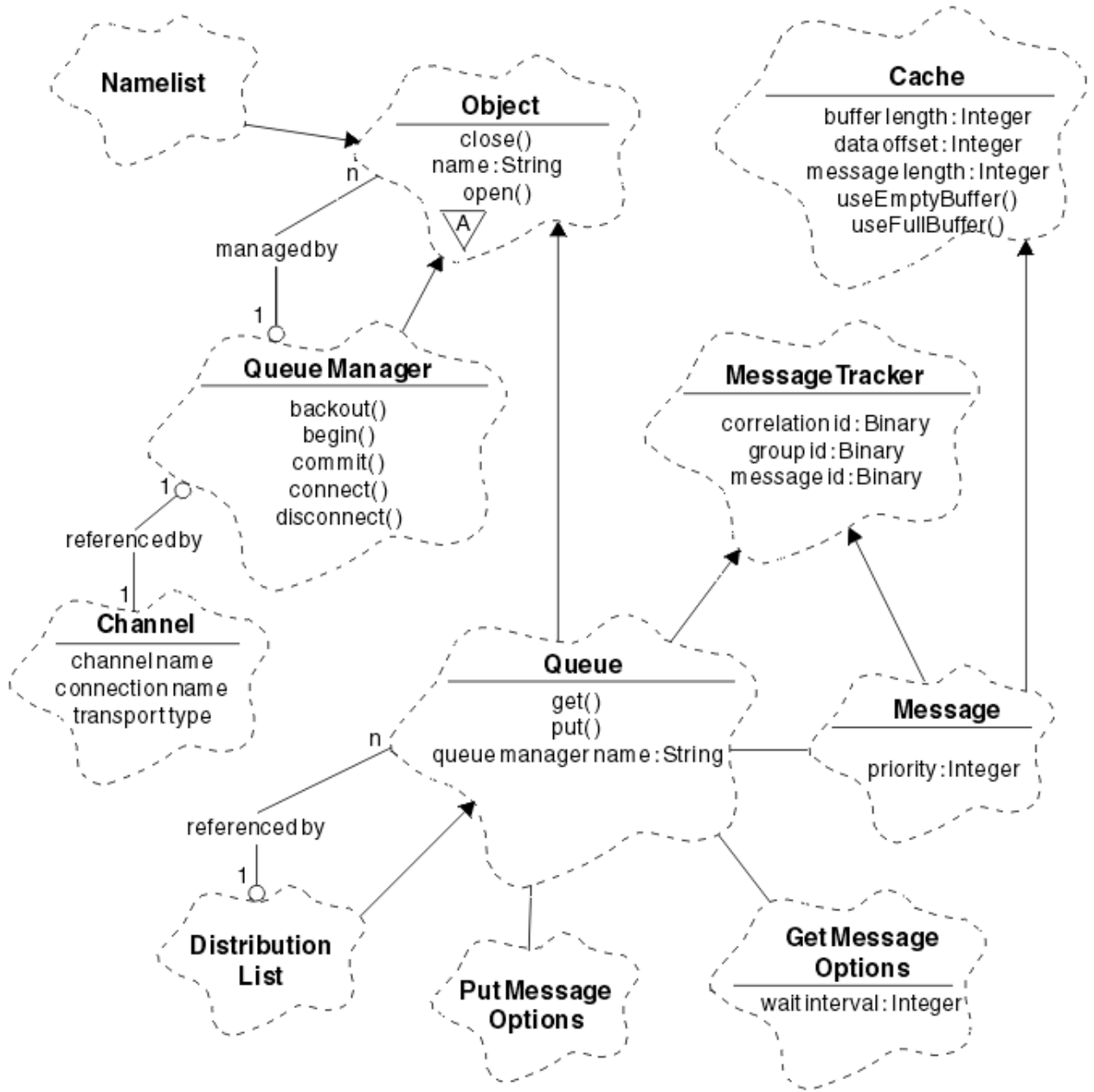
IBM MQ C++ aşağıdaki özellikleri sağlar:

- IBM MQ veri yapılarının otomatik olarak kullanıma hazırlanması.
- Yalnızca zaman içi kuyruk yöneticisi bağlantısı ve kuyruk açma.
- Örtülü kuyruk kapatma ve kuyruk yöneticisi bağlantısı kesiyor.
- -Ölü harf başlığı iletimi ve makbumu.
- IMS köprü üstbilgisi iletimi ve girişi.
- İleti üstbilgisi iletimine ve girişine başvuruda bulunun.
- İleti girişini tetikle.
- CICS bridge üstbilgi iletimi ve girişi.
- İş üstbilgisi iletimi ve girişi.
- İstemci kanalı tanımlaması.

The following Booch class diagrams show that all the classes are broadly parallel to those IBM MQ entities in the procedural MQI (for example using C) that have either handles or data structures. Tüm sınıflar `ImqError` sınıfından devralır (bkz. [ImqError C++ sınıfı](#)); bu, her nesneyle bir hata koşulunun ilişkilendirilmesini sağlar.



Şekil 56. IBM MQ C++ sınıfları (öğe işleme)



Şekil 57. IBM MQ C++ sınıfları (kuyruk yönetimi)

Booch sınıf çizgelerini doğru yorumlamak için aşağıdaki kurallardan haberdar olun:

- Yöntemler ve dikkate değer öznitelikler *sınıf* adının altında gösterilir.
- Bir bulut içindeki küçük bir üçgen *soyut sınıf* anlamına gelir.
- *Edinme* , üst sınıfa bir ok ile gösterilir.
- Bulutlar arasında süslenmemiş bir çizgi, sınıflar arasında bir *işbirliği ilişkisi* anlamına gelir.
- Sayı ile dekore edilen bir satır, iki sınıf arasındaki *gönderisel ilişki* anlamına gelir. Bu sayı, belirli bir ilişkiye herhangi bir anda katılabilecek nesnelerin sayısını gösterir.

Kuyruk yönetimi sınıflarının C++ yöntemi imzalarında aşağıdaki sınıflar ve veri tipleri kullanılır (bkz. Şekil 57 sayfa 502 ). ve öge işleme sınıfları (bkz. Şekil 56 sayfa 501 ):

- ImqBinary sınıfı (bkz. [ImqBinary C++ sınıfı](#) ), MQBYTE24 gibi bayt dizilerini kapsüller.
- **typedef imzalanmamış char ImqBoolean** olarak tanımlanan ImqBoolean veri tipi.
- ImqString sınıfı (bkz. [ImqString C++ sınıfı](#) ); MQCHAR64 gibi karakter dizilerini sarmalayan bir sınıf.

Veri yapılarına sahip varlıklar, uygun nesne sınıfları içinde alt toplamlardır. Tek tek veri yapısı alanları (bkz. [C++ ve MQI çapraz başvurusu](#)) yöntemlerle erişilir.

Tutamaçları olan varlıklar `ImqObject` sınıf sıradüzeninin altında gelir (bkz. `ImqObject C++ sınıfı`) ve MQI 'ya kapsüllenmiş arabirimleri sağlar. Bu sınıfların nesnelere, yordamsal MQI 'ye göre gerekli olan yöntem çağrılarının sayısını azaltan akıllı davranış sergiler. Örneğin, kuyruk yöneticisi bağlantılarını gerektiği şekilde kurabilir ve atabilir ya da uygun seçeneklerle bir kuyruk açabilir, daha sonra bu bağlantıları kapatabilirsiniz.

`ImqMessage` sınıfı (bkz. `ImqMessage C++ sınıfı`) MQMD veri yapısını kapsüller ve aynı zamanda kullanıcı verileri ve *öğeler* için bir tutma noktası görevi görür (bkz. [“C++ dilinde ileti okunuyor”](#) sayfa 512) önbelleğe alınmış arabellek olanakları sağlar. Kullanıcı verileri için değişmez uzunluklu arabellekler sağlayabilir ve arabelleği birçok kez kullanabilirsiniz. Arabelleğindeki veri miktarı, bir kullanımdan diğerine farklılık gösterebilir. Diğer bir seçenek olarak, sistem, esnek uzunluktaki bir arabellek sağlayabilir ve bu arabelleği yönetebilir. Arabelleğin büyüklüğü (iletilerin alınması için kullanılabilir tutar) ve gerçekte kullanılan miktar (iletim için bayt sayısı ya da gerçekte alınan bayt sayısı) önemli noktalar haline gelir.

### **İlgili kavramlar**

[Teknik genel bakış](#)

[“C++ örnek programları”](#) sayfa 503

İletileri almayı ve yerleştirmeyi göstermek için dört örnek program sağlanır.

[“C++ dilindeki önemli noktalar”](#) sayfa 507

Bu konu derlemi, Message Queue Interface (MQI) olanağını kullanan uygulama programlarını yazarken göz önünde bulundurmanız gereken C++ dili kullanımı ve kurallarıyla ilgili konuları ayrıntılarıyla içerir.

[“C++ dilinde ileti verileri hazırlanıyor”](#) sayfa 511

İleti verileri, sistem ya da uygulama tarafından sağlanabilen bir arabellekte hazırlanır. Her iki yöntemde de bazı avantajlar var. Bir arabelleğin kullanılmasına ilişkin örnekler verilmiştir.

[“IBM MQ için uygulama geliştirilmesi”](#) sayfa 5

İletileri göndermek ve almak, kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için uygulamalar geliştirebilirsiniz. IBM MQ , birçok farklı dil ve çerçeve içinde yazılmış uygulamaları destekler.

### **İlgili başvurular**

[“Building IBM MQ C++ programs”](#) sayfa 517

Desteklenen derleyicilerin URL 'si, C++ programlarını ve örneklerini IBM MQ platformlarında 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.





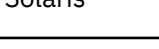

Örnek programlar şunlardır:

- MERHABA DÜNYA (`imqwrlld.cpp`)
- SPUT (`imqspud.cpp`)
- SGET (`imqsgud.cpp`)
- DPUT (`imqdput.cpp`)

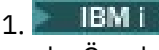

Örnek programlar, [Çizelge 71 sayfa 504](#) içinde gösterilen dizinlerde bulunur.

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini gösterir.

Çizelge 71. Örnek programların yeri

Çevre	Kaynağı içeren dizin	Oluşturulan dizini içeren dizin programlar
 AIX	MQ_INSTALLATION_PATH/samp	MQ_INSTALLATION_PATH/samp/bin/ia
 IBM i	/QIBM/ProdData/mqm/samp/	(bkz. not “1” sayfa 504 )
 Linux	MQ_INSTALLATION_PATH/samp	Yok
 Solaris	MQ_INSTALLATION_PATH/samp	MQ_INSTALLATION_PATH/ samp/bin/as
 Windows	MQ_INSTALLATION_PATH\tools\cplus\ samples	MQ_INSTALLATION_PATH\tools\cplus\ örnekler \bin\vn (bkz. not “2” sayfa 504 )
 z/OS	thlqual.SCSQCPPS	

#### Notlar:

-  IBM i için ILE C++ derleyicisi kullanılarak oluşturulan programlar QMQM kitaplığında yer alır. Örnek dosyalar /QIBM/ProdData/mqm/samp içinde bulunur.
-  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. “Building C++ programs on Windows” sayfa 523.

## Örnek program HELLO WORLD (imqwrlld.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 çağrılarında yararlanarak birkaç yöntem çağrısı kullanır.

### z/OS dışındaki tüm platformlarda

IBM MQ ile sunucu bağlantısı kullanıyorsanız, aşağıdaki yordamlardan birini izleyin:

- Var olan varsayılan kuyruğu kullanmak için, SYSTEM.DEFAULT.LOCAL.QUEUE, herhangi bir parametre geçirmeden **imqwrlld** programını çalıştırın
- Devingen olarak atanmış geçici bir kuyruk kullanmak için, **imqwrlld** komutunu varsayılan model kuyruğunun adını ( SYSTEM.DEFAULT.MODEL.QUEUE).

IBM MQ ile istemci bağlantısı kullanıyorsanız, aşağıdaki yordamlardan birini izleyin:

- MQSERVER ortam değişkenini ayarlayın (ek bilgi için [MQSERVER](#) kısmına bakın) ve **imqwrlldc** 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 **imqwrlldc** 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, adlandırılmış bir kuyruktan iletileri alır ve bu kuyruktan iletileri 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. Run **imqspu**ts *kuyruk-adi*.
2. Konsolda metin satırları yazın. Bu satırlar, belirlenen kuyruğa ileti olarak yerleştirilir.
3. Girişi sonlamak için boş bir satır girin.
4. Tüm satırları almak ve bunları konsolda görüntülemek için **imqsgets** *kuyruk-adi* komutunu çalıştırın.

**z/OS** Ek bilgi için “C++ programlarını z/OS Batch, RRS Batch ve CICS üzerinde oluşturma” sayfa 525 başlıklı konuya bakın.

## Açıkz/OS

**z/OS**

1. Örnek JCL **imqspu**rolanağını kullanarak toplu iş oluşturun ve çalıştırın. İletiler, SYSIN veri kümesinden okunuyor.
2. Örnek JCL **imqsg**etrolanağını kullanarak toplu iş oluşturun ve çalıştırın. İletiler kuyruktan alınır ve SYSPRINT veri kümesine gönderilir.

## Örnek program DPUT (imqdput.cpp)

Bu C++ örnek programı, iletileri iki kuyruktan oluşan bir dağıtım listesine koyar.

DPUT, ImqDistributionListe sınıfının kullanımını gösterir (bkz. [ImqDistributionList C++ sınıfı](#) ). Bu örnek z/OS üzerinde desteklenmez.

1. İki adlandırılmış kuyruğa ileti yerleştirmek için **imqdlets** *queue-name-1 queue-name-2* komutunu çalıştırın.
2. Bu kuyruklardan gelen iletileri almak için **imqsgets** *queue-name-1* ve **imqsgets** *queue-name-2* komutunu çalıştırın.

## C++ dilindeki önemli noktalar

Bu konu derlemi, Message Queue Interface (MQI) olanağını kullanan uygulama programlarını yazarken göz önünde bulundurmanız gereken C++ dili kullanımı ve kurallarıyla ilgili konuları ayrıntılarıyla içerir.

## C++ Üstbilgi dosyaları

Header files are provided as part of the definition of the MQI, to help you write IBM MQ application programs in the C++ language.

Bu üstbilgi dosyaları aşağıdaki tabloda özetlenmiştir.

Çizelge 72. C/C++ üstbilgi kütükleri	
Dosya adı	İçindekiler
IMQI.HPP	C++ MQI Sınıfları ( CMQC.H ve IMQTYPE.H)
IMQTYPE.H	<b>ImqBoolean</b> veri tipini tanımlar
CMQC.H	MQI veri yapıları ve bildirge değişmezleri

Uygulamaların taşınabilirliğini artırmak için, **#include** ön işlemcisi yönergesinde üstbilgi dosyasının adını küçük harfli olarak kodlayın:

```
#include <imqi.hpp> // C++ classes
```

## C++ yöntemleri ve öznitelikleri

Yöntem adları karışık durumda. Parametrelere ve dönüş değerlerine ilişkin çeşitli noktalar geçerlidir. Öznitelikler, set kullanılarak erişilir ve yöntemleri uygun olarak alır.

*const* olan yöntemlerin parametreleri yalnızca giriş içindir. Bir işaretçi (\*) ya da başvuru (&) de içinde olmak üzere imzaları olan parametreler referans olarak iletilir. Bir işaretçi ya da başvuru içermeyen dönüş değerleri, değer temelinde geçirilir; döndürülen nesnelere durumunda bunlar, çağırının sorumluluğu olan yeni varlıklardır.

Bazı yöntem imzaları, belirtilmediyse, varsayılan olarak kabul edilen öğeleri içerir. Bu tür öğeler her zaman imzaların sonunda olur ve eşittir işaretiyle (=) belirtilir; eşittir işareti, öğe atılırsa geçerli olan varsayılan değeri belirtir.

Bu sınıflardaki tüm yöntem adları küçük harfle başlayarak büyük ve küçük harfe karıştırılır. Bir yöntem adı içindeki ilk dışında her sözcük, büyük harfle başlar. Kısaltmaların anlamı geniş bir şekilde anlaşılmadıkça kullanılmaz. Kullanılan kısaltmalar arasında *tanıtıcı* (kimlik için) ve *eşitleme* (eşitleme için) vardır.

Nesne özniteliklerine, set ve get yöntemleri kullanılarak erişilir. Set yöntemi, *set* sözcüğüyle başlar; Get (alma) yönteminin öneki yoktur. Bir öznitelik *set okunur*ise, herhangi bir ayarlama yöntemi yoktur.

Öznitelikler, nesne yapısı sırasında geçerli durumlarla başlatılır ve bir nesnenin durumu her zaman tutarlıdır.

## C++ dilinde veri tipleri

Tüm veri tipleri C **typedef** deyiimiyle tanımlanır.

**ImqBoolean** tipi, IMQTYPE.H içinde **işaretsiz karakter** olarak tanımlanır ve TRUE ve FALSE değerlerine sahip olabilir. You can use **ImqBinary** class objects in place of **MQBYTE** arrays, and **ImqString** class objects in place of **char \***. Çoğu yöntem, depolama yönetimini kolaylaştırmak için **char** ya da **MQBYTE** işaretçileri yerine nesnelere döndürür. Tüm dönüş değerleri çağırının sorumluluğu haline gelir ve döndürülen bir nesne durumunda, saklama alanı silme işlemi kullanılarak atılabilir.

## C++ dilinde ikili dizgileri işleme

İkili veriler dizgileri, **ImqBinary** sınıfının nesnesi olarak bildirilir. Bu sınıfın nesnelere, bilinen C işlemlerini kullanarak kopyalanabilir, karşılaştırılabilir ve bu işlemler kullanılarak ayarlanabilirler. Örnek kod sağlanıyor.

Aşağıdaki kod örneği, ikili bir dizilimin 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 dizilimlerini işleme

Karakter verileri genellikle, bir dönüştürme işlemi kullanılarak **char \*** tipine dönüştürülebilen **ImqString** sınıf nesnelere döndürülür. ImqString sınıfı, karakter dizgilerinin işlenmesine yardımcı olacak yöntemler içerir.

Karakter verileri kabul edildiğinde ya da MQI C++ yöntemleri kullanılarak döndürüldüğünde, karakter verileri her zaman boş değerle sonlandırılır ve herhangi bir uzunluğa sahip olabilir. However, certain limits are imposed by IBM MQ that might result in information being truncated. Depolama yönetimini kolaylaştırmak için, karakter verileri genellikle **ImqString** sınıf nesnelere döndürülür. Bu nesnelere,

sağlanan dönüştürme işlecini kullanarak **char \*** 'a dönüştürülebilirler ve **char \*** ' un gerekli olduğu birçok durumda *salt okunur* için kullanılır.

**Not:** Bir **ImqString** sınıf nesnesinden **char \*** dönüştürme sonucu boş değerli olabilir.

Although C functions can be used on the **char \***, there are special methods of the **ImqString** class that are preferable; **işletmen uzunluğu** ( ) is the equivalent of **strlen** and **depolama** ( ) indicates the memory allocated for the character data.

## C++ dilinde nesnelerin ilk durumu

Tüm nesneler, özniteliklerine yansıtılan tutarlı bir başlangıç durumuna sahiptir. İlk değerler sınıf tanımlarında tanımlanır.

## C++ ' den C++ Kullanılması

C işlevlerini bir C++ programından kullandığınızda, uygun üstbilgileri içerin.

Aşağıdaki örnek, bir C++ programına dahil edilen `string.h` ' i göstermektedir:

```
extern "C" {
#include <string.h>
}
```

## C++ notasyonlu kuralları

Bu örnekte, yöntemlerin nasıl çağrılacağı ve parametrelerin bildirileceği gösterilmektedir.

Bu kod örneği şu yöntemleri ve deęiřtirgeleri kullanır: **ImqBoolean ImqQueue::get ( ImqMessage & msg )**

Deęiřtirgeleri řu řekilde 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 yerine getirmek için, birden çok işlem örtük olarak *tam zamanında*ile oluşabilir. Bu örtülü işlemler, bağlantı, açma, yeniden açma, kapatma ve bağlantı kesme işlemleridir. Sınıf özniteliklerini kullanarak, bağlantı denetimi ve açık örtük davranışı denetleyebilirsiniz.

## Baęlan

Bir `ImqQueueManager` nesnesi, MQI ' ya yönelik herhangi bir çağrıyla sonuçlanan herhangi bir yöntem için otomatik olarak baęlanır (bkz. [C++ ve MQI çapraz başvurusu](#) ).

## Au00e7

Bir MQGET, MQINQ, MQPUT ya da MQSET çağrısına neden olan herhangi bir yöntem için otomatik olarak bir ImqObject nesnesi açılır. Bir ya da daha fazla ilgili **açma seçeneği** değeri belirtmek için **openFor** yöntemini kullanın.

## Yeniden aç

ImqObject , nesnenin zaten açık olduğu bir MQGET, MQINQ, MQPUT ya da MQSET çağrısına neden olan herhangi bir yöntem için otomatik olarak yeniden açılır; ancak, var olan **açık seçenekler** , MQI çağrısının başarılı olmasına izin vermek için yeterli değildir. Bu nesne, geçici olarak MQCO\_NONE için geçici bir **kapatma seçenekleri** değeri kullanılarak kapatıldı. İlgili bir öğe eklemek için **openFor** yöntemini kullanın**seçeneği açın**.

Yeniden açma, belirli durumlarda sorunlara neden olabilir:

- Geçici bir dinamik kuyruk kapatıldığında yok edilir ve hiçbir zaman yeniden açılmayabilir.
- Dışlayıcı giriş (belirtik olarak ya da varsayılan olarak) için açılan bir kuyrukta, kapatma ve yeniden açma işlemi sırasında fırsat penceresindeki diğer kişiler tarafından erişilebilir.
- Bir kuyruk kapatıldığında göz atma konumu kaybedilir. Bu durum kapanmayı ve yeniden açmayı önlemez, ancak MQGMO\_BROWSE\_FIRST komutu yeniden kullanılmadıkça, imlecin sonraki kullanımını önler.
- Bir kuyruk kapatıldığında, alınan son iletinin bağlamı kaybedilir.

Bu koşullardan herhangi biri ortaya çıkarsa ya da öngörülebiliyorsa, bir nesne açılmadan önce (belirtik ya da örtük olarak) yeterli **açık seçenekleri** belirttik olarak ayarlayarak yeniden açılmamayı önleyebilirsiniz.

Karmaşık kuyruk işleme durumları için açık olarak **açık seçenekler** ' in ayarlanması daha iyi performans sağlar 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 edilmişse.

## Bağlantıyı kes

ImqQueueYöneticisi, bağlantının artık geçerli olmadığı herhangi bir noktada otomatik olarak kesilir; örneğin, bir ImqObject bağlantı başvurusu kesilirse ya da bir ImqQueueManager nesnesi yok edilirse.

## C++ dilinde ikili ve karakter dizilimleri

ImqString sınıfı, geleneksel *char \** veri biçimini sarsalıyor. ImqBinary sınıfı, ikili bayt dizisini sarsalıyor. Karakter verilerini ayarlayan bazı yöntemler verilerin kesilmesine neden olabilir.

Methods that set character ( **char \*** ) data always take a copy of the data, but some methods might truncate the copy, because certain limits are imposed by IBM MQ.

ImqString sınıfı (bkz. [ImqString C++ sınıfı](#) ) geleneksel **char \*** ' ı kapsüller ve aşağıdakiler için destek sağlar:

- Karşılaştırma
- Bitiştirme
- Kopyalama
- Integer-to-text ve text-to-integer dönüştürme
- Simge (sözcük) çıkartma
- Büyük harf çevirisi

ImqBinary sınıfı (bkz. [ImqBinary C++ sınıfı](#) ) Rasgele büyüklerin ikili bayt dizilerini sarmalıyor. Özellikle, aşağıdaki öznitelikleri tutmak için kullanılır:

- **muhasabe simgesi** (MQBYTE32)
- **bağlantı etiketi** (MQBYTE128)
- **İlinti tanıtıcısı** (MQBYTE24)
- **tesis simgesi** (MQBYTE8)
- **grup tanıtıcısı** (MQBYTE24)
- **eşgörünüm tanıtıcısı** (MQBYTE24)
- **ileti tanıtıcısı** (MQBYTE24)
- **ileti simgesi** (MQBYTE16)
- **işlem eşgörünümü tanıtıcısı** (MQBYTE16)

Bu özniteliklerin bulunduğu yer, aşağıdaki sınıfların nesnelere aittir:

- ImqCICSBridgeHeader (bkz. [ImqCICSBridgeHeader C++ sınıfı](#) )
- ImqGetMessageOptions (bkz. [ImqGetMessageOptions C++ sınıfı](#) )
- ImqIMSBridgeHeader (bkz. [ImqIMSBridgeHeader C++ sınıfı](#) )
- ImqMessageİzleyici (bkz. [ImqMessageTracker C++ sınıfı](#) )
- ImqQueueManager (bkz. [ImqQueueManager C++ sınıfı](#) )
- ImqReferenceÜstbilgisi (bkz. [ImqReferenceHeader C++ sınıfı](#) )
- ImqWorkÜstbilgisi (bkz. [ImqWorkHeader C++ sınıfı](#) )

ImqBinary sınıfı, karşılaştırma ve kopyalama desteği de sağlar.

## C++ dilinde desteklenmeyen işlevler

IBM MQ C++ sınıfları ve yöntemleri IBM MQ altyapısından bağımsızdır. Bu nedenle, bazı platformlarda desteklenmeyen bazı işlevler sunabilirler.

If you try to use a function on a platform on which it is not supported, the function is detected by IBM MQ but not by the C++ language bindings. IBM MQ , hatayı diğer herhangi bir MQI hatası gibi programınıza bildirir.

## C++ dilinde ileti alışverişi

Bu konular, C + + içinde iletilerin nasıl hazırlanacağını, okunacağını ve yazılacağını ayrıntılarıyla içerir.

### C++ dilinde ileti verileri hazırlanıyor

İleti verileri, sistem ya da uygulama tarafından sağlanabilen bir arabellekte hazırlanır. Her iki yöntemde de bazı avantajlar var. Bir arabelleğin kullanılmasına ilişkin örnekler verilmiştir.

Bir ileti gönderdiğinizde, ileti verileri ilk olarak bir ImqCache nesnesi tarafından yönetilen bir arabellekte hazırlanır (bkz. [ImqCache C++ sınıfı](#) ). Bir arabellek her ImqMessage nesnesiyle ilişkilendirilmiş (kalıtım temelinde) (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 hazırlanmış veri alanlarını doğrudan kullanabileceği için, birçok durumda veri kopyalamanın gerekli olmamasını sağlar. Bu dezavantaj, belirtilen arabelleğin değişmez uzunlukta olması.

Arabellek yeniden kullanılabilir ve iletilmeden önce **setMessageLength** yöntemi kullanılarak iletilen bayt sayısı her seferinde kullanılabilir kılınabilir.

Sistem tarafından otomatik olarak sağlandığında, kullanılabilir bayt sayısı sistem tarafından yönetilir ve veriler, örneğin ImqCache **write** yöntemi ya da ImqMessage **writeItem** yöntemi kullanılarak ileti arabelleğiyle kopyalanabilir. İleti arabelleği gereksinmeye göre büyür. Arabellek büyüdükçe, önceden yazılan veri kaybı olmaz. Büyük ya da çok parçalı bir ileti sıralı parçalarda yazılabilir.

Aşağıdaki örneklerde, basitleştirilmiş ileti gönderileri gösterilmektedir.

### 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 verileri kullanın.

```
char szBuffer[ 24 ] = "Hello world" ;  
msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );  
msg.setFormat( MQFMT_STRING );  
msg.setMessageLength( 12 );
```

### 3. Verileri kullanıcı tarafından sağlanan bir arabelleğe kopyala

```
char szBuffer[ 12 ];  
msg.useEmptyBuffer( szBuffer, sizeof( szBuffer ) );  
msg.setFormat( MQFMT_STRING );  
msg.write( 12, "Hello world" );
```

### 4. Sistem tarafından sağlanan bir arabelleğe veri kopyalama

```
msg.setFormat( MQFMT_STRING );  
msg.write( 12, "Hello world" );
```

### 5. Nesneleri sistem tarafından sağlanan arabelleğe nesneleri kullanarak kopyalama (ileti biçimini ve içeriğin yanı sıra içerik)

```
ImqString strText( "Hello world" );  
msg.writeItem( strText );
```

## C++ dilinde ileti okunuyor

Arabellek, uygulama ya da sistem tarafından sağlanabilir. Verilere doğrudan arabellekten erişilebilir ya da sırayla okunabilirler. Her ileti tipine eşdeğer bir sınıf vardır. Örnek kod verilmiştir.

Veri alınırken, uygulama ya da sistem uygun bir ileti arabelleği sağlayabilir. Aynı arabellek, hem birden çok iletim için, hem de belirli bir ImqMessage nesnesi için birden çok giriş için kullanılabilir. İleti arabelleği otomatik olarak sağlanırsa, alınan veri uzunluğunu sığdırmak için büyür. Ancak, uygulama tarafından sağlanan bir ileti arabelleği, alınan verileri tutmak için yeterli olmayabilir. Daha sonra, ileti girişi için kullanılan seçeneklere bağlı olarak kesilme 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 okunabilmektedir. Bu durumda, veri göstergesi gelen verilerin bir sonraki baytı adreslenir ve veri göstergesi ve veri uzunluğu her okunaca güncellenmektedir.

**Öğeler** , sırayla ve ayrı olarak işlenmesi gereken, ileti arabelleğindeki tüm kullanıcı alanında bulunan bir iletinin parçalarıdır. Normal kullanıcı verileri dışında, bir öge ölü harf üstbilgisi ya da tetikleme ileti olabilir. Öğeler her zaman ileti biçimleriyle ilişkilendirilir; ileti biçimleri her zaman öğelerle ilişkilendirilir **değildir** .

Tanınabilir bir IBM MQ ileti biçimine karşılık gelen her öge için bir nesne sınıfı vardır. Bir tane ölü harf üstbilgisi ve tetikleyici bir mesaj için bir tane var. Kullanıcı verileri için bir nesne sınıfı yok. Yani, tanınabilir biçimler tükendikten sonra, kalan kısmı işleme uygulama programına bırakılır. Kullanıcı verilerine ilişkin sınıflar, ImqItem sınıfı belirtilerek yazılabilir.



Aşağıdaki örnekte, kullanıcı verilerinden önce hayali bir durumda olabilecek bir dizi olası öge dikkate alan bir ileti alındı örneği gösterilmektedir. Öge olmayan kullanıcı verileri, tanımlanabilen ögeler sonrasında ortaya çıkan her şey olarak tanımlanır. Otomatik arabellek (varsayılan değer), ileti verilerinin rasgele bir 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 ilişkili 8 karakterlik biçim adını gösteren bir sabittir ve uygulama tarafından tanımlanır.

UserClass , ImqItem sınıfından türetilir (bkz. [ImqItem C++ sınıfı](#) ) ve sanal **copyOut** ve **pasteIn** yöntemlerini o sınıftan uygular.

The next two examples show code from the ImqDeadLetterHeader class (see [ImqDeadLetterHeader C++ sınıfı](#) ). İlk örnekte, özel kapsüllenmiş ileti- *yazı* kodu gösterilir.

```

// 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 örnekte, özel olarak kapsüllenmiş ileti- *okuma* kodu gösterilir.

```

// 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 depolama alanı *uçucu* olur. Yani, arabellek verileri her **get** yöntemi çağırısından sonra farklı bir fiziksel yerde tutulabilir. Bu nedenle, her zaman arabellek verilerine başvurulsa, ileti verilerine erişmek için **bufferPointer** ya da **dataPointer** yöntemlerini kullanın.

Bir programın ileti verilerini almak için sabit bir alanı bir kenara koymasını isteyebilirsiniz. Bu durumda, **get** yöntemini kullanmadan önce **useEmptyBuffer** yöntemini çağırın.

Sabit olmayan, otomatik olmayan alan, iletileri büyüklük üst sınırına sınırlar. Bu nedenle, `ImqGetMessageOptions` nesnesinin `MQGMO_ACCEPT_TRUNCATED_MSG` seçeneğini göz önünde bulundurmanız önemlidir. Bu seçenek belirlenmezse (varsayılan değer), `MQRC_TRUNCATED_MSG_FAILED` neden kodu beklenebilir. Bu seçenek belirlenirse, uygulamanın tasarımına bağlı olarak `MQRC_TRUNCATED_MSG_KABUL` edilir neden kodu beklenebilir.

Sonraki örnekte, iletilerin nasıl alınabileceği, 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 ;

```

In this code fragment, the buffer can always be addressed directly, with *pszBuffer*, as opposed to using the **bufferPointer** method. Ancak, genel amaçlı erişim için **dataPointer** yönteminin kullanılması daha iyi olur. Uygulama ( `ImqCache` sınıf nesnesi değil), kullanıcı tanımlı (otomatik olmayan) bir arabelleği atmalıdır.

**Dikkat:** Boş değerli bir gösterge ve **useEmptyArabellek** ile sıfır uzunluklu, sıfır uzunluklu değişmez uzunluktaki bir arabelleğinin beklenebileceği gibi bir değer göstermiyor. Bu birleşim, önceki kullanıcı tanımlı arabelleği yoksayma isteği olarak yorumlanır ve bunun yerine otomatik arabellek kullanımına döndür.

## C++ dilinde ölü-mektup kuyruğuna bir ileti yazılıyor

Ölü-mektup kuyruğuna bir ileti yazmak için kullanılan örnek program kodu.

Çok bölümlü bir iletinin tipik bir durumu, bir ölü-harf üstbilgisi içerir. İşlenemeyen bir iletiden gelen veriler, dead-letter 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 köprüsine ileti yazma

IMS köprüsine bir ileti yazmak için kullanılan örnek program kodu.

IBM MQ - IMS köprüsine gönderilen iletiler özel bir üstbilgi kullanabilir. IMS köprüsü üstbilgisine önek olarak olağan 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 ' e bir ileti yazılıyor

CICS bridge' e bir ileti yazmak için kullanılan örnek program kodu.

Messages sent to IBM MQ for z/OS using the CICS bridge require a special header. CICS bridge üstbilgisinde düzenli ileti verileri önekli olur.

```
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 iş üstbilgisiyle ileti yazma

z/OS Workload Manager tarafından yönetilen bir kuyruğa ilişkin ileti yazılmasına ilişkin örnek program kodu.

Messages sent to IBM MQ for z/OS, which are destined for a queue managed by the z/OS Workload Manager, require a special header. İş üstbilgisinin başında düzenli ileti verisi var.

```
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 );
```

## Building IBM MQ C++ programs

Desteklenen derleyicilerin URL 'si, C++ programlarını ve örneklerini IBM MQ platformlarında derlemek, bağlamak ve çalıştırmak için kullanılacak komutlarla birlikte listelenir.

Desteklenen her altyapıya ve IBM MQ sürümüne ilişkin derleyicilerin bir listesi için bkz. [IBM MQ](#).

IBM MQ C++ programınızı derlemek ve bağlamak için gereken komut, kuruluş ve gereksinmelerinize bağlıdır. Aşağıdaki örneklerde, bazı derleyicilerin bazıları için tipik derleme ve bağlantı oluşturma komutları, IBM MQ ' un varsayılan kuruluşu olan platformlarda varsayılan kurulumdur.

**AIX**

## Building C++ programs on AIX

XL C Enterprise Edition derleyicisini kullanarak AIX üzerinde IBM MQ C++ programları oluşturun.

### İstemci

*MQ\_INSTALLATION\_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

#### 32 bitlik iş parçacıklı uygulama

```
xlC -o imqsputc_32 imqspcut.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib -limqc23ia -limqb23ia -lmqic
```

#### 32 bitlik iş parçacıklı uygulama

```
xlC_r -o imqsputc_32_r imqspcut.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib -limqc23ia_r -limqb23ia_r -lmqic_r
```

#### 64 bitlik iş parçacıklı uygulama

```
xlC -q64 -o imqsputc_64 imqspcut.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -limqc23ia -limqb23ia -lmqic
```

#### 64 bitlik iş parçacıklı uygulama

```
xlC_r -q64 -o imqsputc_64_r imqspcut.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -limqc23ia_r -limqb23ia_r -lmqic_r
```

### Sunucu

*MQ\_INSTALLATION\_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

#### 32 bitlik iş parçacıklı uygulama

```
xlC -o imqspcut_32 imqspcut.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib -limqs23ia -limqb23ia -lmqm
```

#### 32 bitlik iş parçacıklı uygulama

```
xlC_r -o imqspcut_32_r imqspcut.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib -limqs23ia_r -limqb23ia_r -lmqm_r
```

#### 64 bitlik iş parçacıklı uygulama

```
xlC -q64 -o imqspcut_64 imqspcut.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -limqs23ia -limqb23ia -lmqm
```

#### 64 bitlik iş parçacıklı uygulama

```
xlC_r -q64 -o imqspcut_64_r imqspcut.cpp -qchars=signed -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -limqs23ia_r -limqb23ia_r -lmqm_r
```

**IBM i**

## Building C++ programs on IBM i

ILE C++ derleyicisini kullanarak IBM i üzerinde IBM MQ C++ programları oluşturun.

IBM ILE C++ for IBM i , C++ programları için yerli bir derleyicidir. Aşağıdaki yönergelerde, *Merhaba Dünya!* u kullanarak IBM MQ C++ uygulamaları yaratmak için bu derleyicinin nasıl kullanılacağı açıklanmaktadır. Örnek olarak IBM MQ örnek programı.

1. Install the ILE C++ for IBM i compiler as directed in the *Önce beni oku!* Ürünle birlikte gönderilen elkitabı.
2. QCXXN kitaplığının kitaplık listenizde yer aldığından emin olun.
3. HELLO WORLD örnek programı yarat:
  - a. Modül yarat:

```
CRTCPMOD MODULE(MYLIB/IMQWRLD) +
SRCSTMF('/QIBM/ProdData/mqm/samp/imqwrlld.cpp') +
INCDIR('/QIBM/ProdData/mqm/inc') DFTCHAR(*SIGNED) +
TERASPACE(*YES)
```

C++ örnek programlarının kaynağı /QIBM/ProdData/mqm/samp içinde bulunabilir ve içerme dosyaları /QIBM/ProdData/mqm/inc içinde yer alır.

Diğer bir seçenek olarak, kaynak kitaplıkta SRCFILE(QCPPSRC/LIB) SRCMBR(IMQWRLD) adlı kitaplıkta bulunabilir.

- b. Bu programı, bir program nesnesi oluşturmak için IBM MQ tarafından sağlanan hizmet programlarıyla bağlayın:

```
CRTPGM PGM(MYLIB/IMQWRLD) MODULE(MYLIB/IMQWRLD) +
BNDSRVPGM(QMQM/IMQB23I4 QMQM/IMQS23I4)
```

Yıvli 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 Building C++ programs on Linux

GNU g++ derleyicisini kullanarak Linux üzerinde IBM MQ C++ programları oluşturun.

### System p

`MQ_INSTALLATION_PATH` , IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder.

### İstemci: System p

#### 32 bitlik iş parçacıklı uygulama

```
g++ -m32 -o imqspc32 imqspc.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 bitlik iş parçacıklı uygulama

```
g++ -m32 -o imqspc_r32 imqspc.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 bitlik iş parçacıklı 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 bitlik 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 bitlik iş parçacıklı 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 bitlik 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 bitlik iş parçacıklı 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 bitlik 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 temsil eder.

## İstemci: IBM Z

### 32 bitlik iş parçacıklı 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 bitlik 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 bitlik iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsputc.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 bitlik iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsputc.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 bitlik iş parçacıklı uygulama

```
g++ -m31 -fsigned-char -o imqsputc_32 imqsputc.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 bitlik iş parçacıklı uygulama

```
g++ -m31 -fsigned-char -o imqsputc_32_r imqsputc.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 bitlik iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64 imqsputc.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 bitlik iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqsputc.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 temsil eder.

### İstemci: x86-64 (32 bit)

#### 32 bitlik iş parçacıklı uygulama

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsputc.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 bitlik iş parçacıklı uygulama

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsputc.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 bitlik iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64 imqspcut.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 bitlik iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqsputc_64_r imqspcut.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 bitlik iş parçacıklı uygulama

```
g++ -m32 -fsigned-char -o imqspcut_32 imqspcut.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 bitlik iş parçacıklı uygulama

```
g++ -m32 -fsigned-char -o imqspcut_32_r imqspcut.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 bitlik iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqspcut_64 imqspcut.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 bitlik iş parçacıklı uygulama

```
g++ -m64 -fsigned-char -o imqspcut_64_r imqspcut.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
```

## **Solaris** Building C++ programs on Solaris

Sun ONE derleyicisini kullanarak Solaris üzerinde IBM MQ C++ programları oluşturun.

### SPARC

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

### Müşteri: SPARC

#### 32 bit uygulama

```
CC -xarch=v8plus -mt -o imqspcut_32 imqspcut.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as
-lmqic -lsocket -lnsl -ldl
```

## 64 bit uygulama

```
CC -xarch=v9 -mt -o imqsputc_64 imqspcut.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as  
-limqb23as  
-lmqic -lsocket -lnsl -ldl
```

## Sunucu: SPARC

### 32 bit uygulama

```
CC -xarch=v8plus -mt -o imqspcut_32 imqspcut.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as  
-lmqm -lsocket -lnsl -ldl
```

### 64 bit uygulama

```
CC -xarch=v9 -mt -o imqspcut_64 imqspcut.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as  
-limqb23as  
-lmqm -lsocket -lnsl -ldl
```

## x86-64

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

## İstemci: x86-64

### 32 bit uygulama

```
CC -xarch=386 -mt -o imqspcut_32 imqspcut.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as  
-lmqic -lsocket -lnsl -ldl
```

### 64 bit uygulama

```
CC -xarch=amd64 -mt -o imqspcut_64 imqspcut.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as  
-limqb23as  
-lmqic -lsocket -lnsl -ldl
```

## Sunucu: x86-64

### 32 bit uygulama

```
CC -xarch=386 -mt -o imqspcut_32 imqspcut.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as  
-lmqm -lsocket -lnsl -ldl
```

### 64 bit uygulama

```
CC -xarch=amd64 -mt -o imqspcut_64 imqspcut.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as  
-limqb23as  
-lmqm -lsocket -lnsl -ldl
```

## Building C++ programs on Windows

Microsoft Visual Studio C++ derleyicisini kullanarak Windows üzerinde IBM MQ C++ programları oluşturun.



**Uyarı:** IBM MQ tarafından gönderilen kitaplıklar dinamik kitaplıklardır ve statik kitaplıklar değildir. IBM MQ , yalnızca derleme süresi sırasında kullanabileceğiniz "import libraries" olarak bilinen bir şey sağlar. Yürütme ortamı için, dinamik kitaplıkları kullanmanız gerekir.

From IBM MQ 8.0.0 Fix Pack 4, IBM MQ ships redistributable clients, containing libraries required for running IBM MQ applications. Bu kitaplıklar, istemci uygulamalarıyla paketlenmiş ve yeniden dağıtılabilir. Daha fazla bilgi için bkz. [Redistributable clients on Windows](#).

32 bit uygulamalarla kullanım için kitaplık (.lib) dosyaları ve dll dosyaları *MQ\_INSTALLATION\_PATH/Tools/Lib* 'te kurulur. 64 bit uygulamalarıyla kullanılacak dosyalar *MQ\_INSTALLATION\_PATH/Tools/Lib64* 'ta kurulur. *MQ\_INSTALLATION\_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

## İstemci

```
cl -MD imqspcut.cpp /Feimqspcutc.exe imqb23vn.lib imqc23vn.lib
```

## Sunucu

```
cl -MD imqspcut.cpp /Feimqspcut.exe imqb23vn.lib imqs23vn.lib
```

## Evrensel C yürütme ortamının kurulması

Windows 8.1 ya da Windows Server 2012 R2 kullanıyorsanız, evrensel C çalışma zamanı güncellemesini (Universal CRT) Microsoft' den kurmalısınız. Bu çalışma zamanı, Windows 10 ve Windows Server 2016 'nın bir parçası olarak dahil edilir.

Universal CRT güncellemesi Microsoft update KB3118401' dir. C : \Windows\System32 dizininizde ucrtbase.dll adlı bir dosya için arama yaparak bu güncelleştirmeye sahip olup olmadığınızı denetleyebilirsiniz. Yoksa, güncellemeyi şu Microsoft sayfasından karşıdan yükleyebilirsiniz: <https://www.catalog.update.microsoft.com/Search.aspx?q=kb3118401>.

Attempting to run an IBM MQ program, or a program you compile yourself using Microsoft Visual Studio 2017, without the runtime installed results in errors such as the following error:

```
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 yürütme ortamları sağlama

Bir IBM MQ programını Microsoft Visual Studio 2012 komutunu kullanarak derlediyseniz, IBM MQ kuruluş programının Microsoft Visual Studio 2012 C/C++ yürütme ortamlarını kurmadığından haberdar olun. Önceki IBM MQ sürümünüz aynı bilgisayara kurulduysa, bu kuruluştan Microsoft Visual Studio 2012 çalışma zamanları kullanılabilir.

Ancak, Microsoft Visual Studio 2012 kullanılarak oluşturulmuş ve önceki bir IBM MQ sürümü kurulu olmayan bir program kullanıyorsanız, aşağıdaki şeylerden birini yapmanız gerekir:

- **Microsoft Visual C++ Redistributable for VisualStudio 2017 (32 and 64-bit versions)** dosyasını karşıdan yükleyin ve Microsoft' den kurun.
- Recompile your program with Microsoft Visual Studio 2017, or another Microsoft Visual Studio level for which the runtimes are installed.

## Microsoft Visual Studio 2015 derleyicisi kullanılarak oluşturulan C++ istemci kitaplıkları

IBM MQ , Microsoft Visual Studio 2015 C++ derleyicisi ve Microsoft Visual Studio 2017 C++ derleyicisi ile oluşturulmuş C++ istemci kitaplıkları sağlar.

Both 32-bit and 64-bit versions of the IBM MQ C++ libraries are provided. 32 bit kitaplıklar bin\vs2015 klasörü altına kurulur ve 64 bit kitaplıkları bin64\vs2015 klasörlerinin altına kurulur.

Varsayılan olarak IBM MQ , Microsoft Visual Studio 2017 kitaplıklarını kullanacak şekilde yapılandırılır. To use the Microsoft Visual Studio 2015 libraries you must set the MQ\_PREFIX\_VS\_LIBRARIES environment variable to MQ\_PREFIX\_VS\_LIBRARIES=vs2015 before you install IBM MQ, or before you use the **setmqenv** or **setmqinst** command.

## Using differently named IBM MQ C++ libraries

IBM MQ , farklı adı taşıyan bazı ek C++ istemci kitaplıkları sağlar. Bu kitaplıklar, Microsoft Visual Studio 2015 ve Microsoft Visual Studio 2017 C++ derleyicileriyle oluşturulmuştur. Bu kitaplıklar, aynı zamanda Microsoft Visual Studio 2017 C++ derleyicisi ile oluşturulmuş var olan C++ kitaplıklarına ek olarak sağlanır. Bu ek IBM MQ C++ kitaplıklarının adları farklı olduğundan, IBM MQ C++ kullanılarak oluşturulmuş ve ürünün Microsoft Visual Studio 2017 ve önceki sürümleriyle derlenmiş olan IBM MQ C++ uygulamalarını aynı bilgisayarda çalıştırabilirsiniz.

Ek Microsoft Visual Studio 2017 kitaplıklarında aşağıdaki adlar bulunur:

- imqb23vnvs2017.dll
- imqc23vnvs2017.dll
- imqs23vnvs2017.dll
- imqx23vnvs2017.dll

Ek Microsoft Visual Studio 2015 kitaplıklarında aşağıdaki adlar bulunur:

- imqb23vnvs2015.dll
- imqc23vnvs2015.dll
- imqs23vnvs2015.dll
- imqx23vnvs2015.dll

Bu kitaplıkların hem 32 bitlik, hem de 64 bitlik sürümleri sağlanır. 32 bit kitaplıklar bin klasörü altına kurulur ve 64 bit kitaplık bin64 klasörü altına kurulur. İlgili içe aktarma kitaplıkları, Tools\lib ve Tools\lib64 dizinlerinin altına kurulur.

If your application uses imq\*vs2015.lib files, you must compile it using the Microsoft Visual Studio 2015 compiler. Microsoft Visual Studio 2015 ile derlenen IBM MQ C++ uygulamalarını ya da aynı bilgisayarda ürünün önceki bir sürümüyle derlenmiş olan uygulamaları çalıştırmak için, PATH ortam değişkenine aşağıdaki örneklerde gösterildiği gibi örnek olarak örnek eklenmelidir:

- 32 bit uygulamalar için:

```
SET PATH=installation_folder\bin\vs2015;%PATH%
```

- 64 bit uygulamalar için:

```
SET PATH=installation_folder\bin64\vs2015;%PATH%
```

## İlgili başvurular

[Windows: IBM MQ 8.0' daki değişiklikler](#)

## C++ programlarını z/OS Batch, RRS Batch ve CICSüzerinde oluşturma

Build IBM MQ C++ programs on z/OS for the Batch, RRS batch or CICS environments and run the sample programs.

IBM MQ for z/OS ' in desteklediği ortamlardan üç tanesi için C++ programları yazabilirsiniz:

- Toplu

- RRS toplu işi
- CICS

### **Derleme, bağlantı önle ve bağlantı**

C++ kaynak kodunuzu derleyerek, önlemeye ve bağlantı oluşturmaya göre bir z/OS uygulaması yaratın.

IBM MQ C++ for z/OS , IBM C++ for z/OS diline ilişkin z/OS DLL ' ler olarak uygulanır. DLL ' ler kullanarak, sağlanan tanım yardımcı destelerini, ön bağlantı sırasında derleyici çıktısıyla birleştirin. Bu, linker ' ın çağrılarınızı IBM MQ C++ üyesi işlevleriyle denetlemesini sağlar.

**Not:** Üç çevrenin her biri için üç takım yan desteler vardır.

Bir IBM MQ for z/OS C++ uygulaması oluşturmak için JCL oluşturun ve çalıştırın. Aşağıdaki yordamı kullanın:

1. Uygulamanız CICS altında çalışıyorsa, programınızdaki CICS komutlarını çevirmek için CICS tarafından sağlanan yordamı kullanın.

In addition, for CICS applications you need to:

- a. SCSQLOAD kitaplığını DFHRPL birleştirmeye ekleyin.
- b. SCSQPROC kitaplığındaki IMQ4B100 adlı üyeyi kullanarak CSQCAT1 CEDA grubunu tanımlayın.
- c. CSQCAT1 ürününü kurun.

2. Nesne kodunu üretmek için programı derleyin. Derlemeniz için JCL, derleyicinin kullanabileceği ürün verileri tanımlama dosyalarını içeren deyimler içermelidir. Veri tanımları aşağıdaki IBM MQ for z/OS kitaplıklarında sağlanır:

- **thlqual.SCSQC370**
- **thlqual.SCSQHPPS**

/cxx derleyici seçeneğini belirttiğinizden emin olun.

**Not:** **thlqual** adı, z/OS üzerindeki IBM MQ kuruluş kitaplığının üst düzey niteleyicidir.

3. Pre-link the object code created in step “2” [sayfa 526](#), including the following definition sivedecks, which are supplied in **thlqual.SCSQDEFS**:

- a. Toplu iş için imqqs23dm ve imqb23dm
- b. RRS toplu işi için imqqs23dr ve imqb23dr
- c. CICS için imqqs23dc ve imqb23dc

Bunlar, ilgili DLL ' ler.

- a. Toplu iş için imqqs23im ve imqb23im
- b. RRS toplu işi için imqqs23ir ve imqb23ir
- c. CICS için imqqs23ic ve imqb23ic

4. Link-edit the object code created in step “3” [sayfa 526](#), to produce a load module, and store it in your application load library.

Toplu ya da RRS toplu iş programlarını çalıştırmak için, STEPLIB ya da JOBLIB veri kümesi bitişirmesine ilişkin **thlqual.SCSQAUTH** ve **thlqual.SCSQLOAD** kitaplıklarını ekleyin.

Bir CICS programını çalıştırmak için, önce sistem denetimcinizden CICS programını IBM MQ programı ve işlemi olarak tanımlamanız gerekir. Daha sonra her zamanki gibi çalıştırabilirsiniz.

### **Örnek programları çalıştır**

Programlar, “C++ örnek programları” [sayfa 503](#) içinde açıklanmıştır.

Örnek uygulamalar yalnızca kaynak biçimiyle sağlanır. Dosyalar şunlardır:

Çizelge 73. z/OS örnek program dosyaları

Örnek	Source program (in library thlqual.SCSQCPPS)	JCL ( thlqual.SCSQPROckitaplığı)
MERHABA DÜNYA	imqwrlld	imqwrlldr
SPUT	imqspud	imqspudr
SGET	imqsget	imqsgetr

Örnekleri çalıştırmak için, herhangi bir C++ programıyla birlikte bunları derleyin ve bağlayın (bkz. “C++ programlarını z/OS Batch, RRS Batch ve CICSüzerinde oluşturma” sayfa 525 ). Toplu iş oluşturmak ve çalıştırmak için sağlanan JCL ' yi kullanın. Başlangıçta, JCL ' yi uyarlayarak, bu açıklamayla birlikte verilen açıklamaları izleyerek uyarlamalısınız.

## z/OS Building C++ programs on z/OS UNIX System Services

Unix System Services için z/OS üzerinde IBM MQ C++ programları oluşturun.

To build an application under the UNIX System Services shell, you must give the compiler access to the IBM MQ include files (located in thlqual.SCSQC370 and hlqual.SCSQHPPS ), and link against two of the DLL sidedecks (located in thlqual.SCSQDEFS ). At runtime, the application needs access to the IBM MQ data sets thlqual.SCSQLOAD, thlqual.SCSQAUTH, and one of the language specific data sets, such as thlqual.SCSQANLE<sup>6</sup>.

### Derleniyor

1. Örneği TSO **oput** komutunu kullanarak dosya sistemine kopyalayın ya da FTP ' yi kullanın. Bu örnekte, örneği /u/fred/sampleadlı bir dizine kopyaladığınız ve adı imqwrlld .cppolarak adlandırdığınız varsayılmıştır.
2. UNIX System Services kabuğunda oturum açın ve örneği yerleştirdiğiniz dizine geçin.
3. C++ derleyicisini, giriş olarak DLL yardımcı programı ve .cpp dosyalarını kabul edebilmesi için 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 destesiyle bağlar; bunun yerine RRS toplu iş desteleri kullanılabilir. \ karakteri, komutu birden çok satırda bölmek için kullanılır. Bu karakteri girmeyin; komutu tek bir satır olarak girin:

```
/u/fred/sample:> c++ -o imqwrlld -I "'thlqual.SCSQC370'" \
-I "'thlqual.SCSQHPPS'" imqwrlld.cpp \
"'thlqual.SCSQDEFS(IMQS23DM)'" "'thlqual.SCSQDEFS(IMQB23DM)'"
```

TSO **oput** komutuna ilişkin ek bilgi için [z/OS UNIX System Services Command Reference](#)adlı sayfaya bakın.

C++ programlarını oluşturmayı kolaylaştırmak için make yardımcı programını da kullanabilirsiniz. Burada, HELLO WORLD C++ örnek programını oluşturmak için bir örnek makefile yer alıyor. Derleme ve bağlantı aşamalarını ayırır. Set up the environment as in step “3” sayfa 527 before running make.

```
flags = -I "'thlqual.SCSQC370'" -I "'thlqual.SCSQHPPS'"
decks = "'thlqual.SCSQDEFS(IMQS23DM)'" "'thlqual.SCSQDEFS(IMQB23DM)'"

imqwrlld: imqwrlld.o
c++ -o imqwrlld imqwrlld.o $(decks)
```

<sup>6</sup> UNIX sistem hizmetini üç ortamda ( “C++ programlarını z/OS Batch, RRS Batch ve CICSüzerinde oluşturma” sayfa 525 ) çalıştırmak için “Nesne kodunda önceden bağlantı oluşturma ” altında listelenen tüm yardımcı güvertelerle bağlantı kurabilirsiniz.

```
imqwrld.o: imqwrld.cpp
c++ -c -o imqwrld $(flags) imqwrld.cpp
```

Make kullanımı hakkında daha fazla bilgi için [z/OS UNIX System Services Programming Tools](#) başlıklı konuya bakın.

### Çalışıyor

1. UNIX System Services kabuğunda oturum açın ve örneği oluşturduğunuz 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
```

## .NET uygulamalarının geliştirilmesi

IBM MQ classes for .NET , .NET programlama çerçevesinde yazılmış bir programın IBM MQ ile IBM MQ MQI client arasında bağlantı kurmasını ya da bir IBM MQ sunucusuna doğrudan bağlanmasını sağlar.

If you have applications which use Microsoft .NET Framework and want to take advantage of the facilities of IBM MQ, you must use IBM MQ classes for .NET. Daha fazla bilgi için bkz [“kurmaIBM MQ classes for .NET Framework”](#) sayfa 534.

**V 9.1.1** IBM MQ 9.1.1, IBM MQ , Windows ortamlarındaki uygulamalar için .NET Core ' i destekler. Daha fazla bilgi için bkz [“kurmaIBM MQ classes for .NET Standard”](#) sayfa 530.

**V 9.1.2** IBM MQ 9.1.2, IBM MQ , Linux ortamlarındaki uygulamalar için .NET Core ' i destekler.

**V 9.1.4** IBM MQ 9.1.4' tan IBM MQ .NET yönetilen uygulamaları, kümelenmiş kuyruk yöneticilerindeki bağlantıları otomatik olarak dengelemeyi başarır. .NET Framework ve .NET Standard kitaplıkları desteklenmektedir. Daha fazla bilgi için bkz. [Uniform kümeleri ve Otomatik uygulama dengeleme](#).

Nesne yönelimli IBM MQ .NET arabirimi, MQI fiillerini kullanmak yerine nesne yöntemlerini kullandığından, MQI arabiriminden farklıdır.

Yordamsal IBM MQ uygulama programlama arabirimi, aşağıdaki listede yer alan fiiller etrafında oluşturulmuştur:

```
MQCONN, MQDISC, MQOPEN, MQCLOSE,
MQINQ, MQSET, MQGET, MQPUT, MQSUB
```

Bu fiillerin tümü, bir parametre olarak, üzerinde çalışacakları IBM MQ nesnesinin bir tanıtıcısı olarak alır. Because .NET is object-oriented, the .NET programming interface turns this round. Programınız, bu nesnelere ilgili yöntemler çağırarak işlem yapmak istediğiniz IBM MQ nesnelere oluşur. Programları .NET tarafından desteklenen herhangi bir dilde yazabilirsiniz.

Yordamsal arabirimi kullanırken, MQDISC çağrısını ( *Hconn*, CompCode, Reason) kullanarak bir kuyruk yöneticisinden bağlantınızı kesilir; burada *Hconn* , kuyruk yöneticisine ilişkin bir tanıtıcıdır.

.NET arabiriminde, kuyruk yöneticisi, MQQueueManagersınıfı bir nesle temsil edilir. Bu sınıftaki Disconnect () yöntemini çağırarak kuyruk yöneticisinden bağlantıyı kesmenizi sağlar.

```
// 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 MQ ile etkileşimli çalışabilmesini sağlayan bir sınıf kümesidir. Bunlar, uygulamanızın kullandığı (kuyruk yöneticileri, kuyruklar, kanallar ve iletiler gibi) IBM MQ ' un çeşitli bileşenlerini temsil eder. Bu sınıfların ayrıntıları için [IBM MQ .NET sınıflarına ve arabirimlerine](#) bakın.

Yazdığınız uygulamaları derlemeden önce, bir .NET Framework ürününün kurulu olması gerekir. IBM MQ classes for .NET ve .NET Framework ürününün kurulmasına ilişkin yönergeler için bkz. [“kurma IBM MQ classes for .NET Framework”](#) sayfa 534.

### **İlgili kavramlar**

[“IBM MQ classes for .NET ' u kuyruk yöneticisine bağlama seçenekleri”](#) sayfa 529

There are three modes of connecting IBM MQ classes for .NET to a queue manager. Gereksinimlerinize en uygun bağlantı tipini göz önünde bulundurun.

[“IBM MQ .NET programlarının yazılması ve konuşlandırılması”](#) sayfa 547

To use IBM MQ classes for .NET to access IBM MQ queues, you write programs in any language supported by .NET containing calls that put messages onto, and get messages from, IBM MQ queues.

[“IBM MQ için uygulama geliştirilmesi”](#) sayfa 5

İletileri göndermek ve almak, kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için uygulamalar geliştirebilirsiniz. IBM MQ , birçok farklı dil ve çerçeve içinde yazılmış uygulamaları destekler.

### **İlgili görevler**

[Troubleshooting IBM MQ .NET problems](#)

[“IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi”](#) sayfa 1292

IBM MQ için Microsoft Windows Communication Foundation (WCF) özel kanalı, WCF istemcileri ve hizmetleri arasında ileti gönderir ve iletileri alır.

### **İlgili başvurular**

[Teknik genel bakış](#)

## **IBM MQ classes for .NET olanağını kullanmaya başlama**

IBM MQ classes for .NET , .NET programlama çerçevesinde yazılmış bir programın IBM MQ ile IBM MQ MQI client arasında bağlantı kurmasını ya da bir IBM MQ sunucusuna doğrudan bağlanmasını sağlar.

### **IBM MQ classes for .NET ' u kuyruk yöneticisine bağlama seçenekleri**

There are three modes of connecting IBM MQ classes for .NET to a queue manager. Gereksinimlerinize en uygun bağlantı tipini göz önünde bulundurun.

### **İstemci bağ tanımları bağlantısı**

IBM MQ classes for .NET ' u IBM MQ MQI client olarak kullanmak için, IBM MQ sunucusu makinesinde ya da ayrı bir makinede IBM MQ MQI client ile birlikte kurabilirsiniz. İstemci bağ tanımları bağlantısı XA ya da XA dışı 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 ' yı kullanır. This provides better performance for IBM MQ applications than using network connections.

To use the bindings connection, you must install IBM MQ classes for .NET on the IBM MQ server.

## Yönetilen istemci bağlantısı

Bu kipte yapılan bir bağlantı, yerel ya da uzak bir makinede çalışan bir IBM MQ istemcisine IBM MQ istemcisi olarak bağlanır.

Bu kipteki IBM MQ classes for .NET bağlantısı .NET tarafından yönetilen kodda kalır ve yerel hizmetlere çağrılar yapmaz. Yönetilen kodla ilgili daha fazla bilgi için Microsoft belgelerine bakın.

Yönetilen istemciyi kullanmak için bir dizi sınırlama vardır. Bunlarla ilgili daha fazla bilgi için bkz. “Yönetilen istemci bağlantıları” sayfa 548.

### Linux Windows V 9.1.1 kurmaIBM MQ classes for .NET Standard

Örnekler de dahil olmak üzere IBM MQ 9.1.1, IBM MQ classes for .NET Standard'den Windows' ta IBM MQ ile kurulum. V 9.1.2 IBM MQ 9.1.2, IBM MQ classes for .NET Standard , Linux platformlarında da kullanılabilir. There is a prerequisite of Microsoft.NET Core for IBM MQ classes for .NET Standard.

## Önkoşullar ve kuruluş

IBM MQ classes for .NET Standard ' un en son sürümü, *Java ve .NET Messaging ve Web Hizmetleri* özelinde standart IBM MQ kurulumunun bir parçası olarak varsayılan olarak kurulur.

IBM MQ 9.1.1saatinde, IBM MQ classes for .NET Standard yalnızca Windows üzerinde kullanılabilir.

V 9.1.2 IBM MQ 9.1.2' tan IBM MQ classes for .NET Standard , Linux platformlarında ve Windowsplatformlarında da kullanılabilir.

IBM MQ classes for .NET Standardkomutunu çalıştırmak için, Microsoft .NET Core' i kurmalısınız.

Windows For more information about prerequisites and installation on Windows:

- See [IBM MQ classes for .NETgereksinimleri](#), for the prerequisite software to run IBM MQ classes for .NET Standard.
- Kuruluş yönergeleri için bkz. [Installing IBM MQ server on Windows](#) ya da [Installing an IBM MQ client on Windows systems](#) .

V 9.1.2 Linux For more information about prerequisites and installation on Linux:

- See [IBM MQ classes for .NETgereksinimleri](#), for the prerequisite software to run IBM MQ classes for .NET Standard.
- Rpm kuruluş yönergeleri için bkz. [Installing an IBM MQ Client on Linux systems](#).
- Linux Ubuntu için, Debian paketlerini kullanarak, bkz. [Linux systems üzerinde IBM MQ istemcisi kurulması](#).

V 9.1.4 From IBM MQ 9.1.4, the IBM MQ classes for .NET Standard library, `amqmdnetstd.dll`, is available for downloading from the NuGet repository. Daha fazla bilgi için bkz. “[IBM MQ classes for .NET Standard dosyasını NuGet havuzundan yükleme](#)” sayfa 533.

## amqmdnetstd.dll kitaplık

Windows IBM MQ 9.1.1' tan `amqmdnetstd.dll` kitaplığı, Windowsüzerinde .NET Standard desteği için kullanılabilir. Kaynak dosyalar da içinde olmak üzere örnek uygulamalar da sağlanır; bkz. “[.NETiçin örnek uygulamalar](#)” sayfa 534.

V 9.1.2 Linux IBM MQ 9.1.2'ta `amqmdnetstd.dll` kitaplığı da Linux' ta kullanılabilir. Bir IBM MQ istemcisi Linuxüzerine kurulu olduğunda, kitaplık `&MQINSTALL_PATH%/lib64` path içine kurulur. .NET örnekleri, `&MQINSTALL_PATH%/samp/dotnet/samples/cs/core/base` içinde bulunur.

Any library built against the Microsoft .NET Standard specification can be used for developing .NET Framework applications as well as .NET Core applications.



**Uyarı:** .NET Framework için amqmdnet.dll kitaplığı hala sağlanıyor, ancak bu kitaplık sabitlendi; başka bir özellik de bu kitaplığa tanıtılacak.

En son özelliklerden herhangi biri için amqmdnetstd.dll kitaplığına geçiş yapmak gerekir. Ancak, amqmdnet.dll kitaplığını IBM MQ 9.1 Long Term Support ya da Continuous Delivery yayın düzeylerinde kullanmaya devam edebilirsiniz.

## dspmqr command

IBM MQ 9.1.1' tan, .NET Core bileşenine ilişkin sürüm ve oluşturma bilgilerini görüntülemek için **dspmqr** komutunu kullanabilirsiniz.

## IBM MQ classes for .NET Framework ve IBM MQ classes for .NET Standard özellikleri

Aşağıdaki çizelgede, IBM MQ 9.1.1 for IBM MQ classes for .NET Framework ve IBM MQ classes for .NET Standard ürünlerinde geçerli olan özellikler listelenmektedir.

Çizelge 74. IBM MQ classes for .NET Framework ve IBM MQ classes for .NET Standard özellikleri arasındaki farklar		
Özellik	IBM MQ classes for .NET Framework	IBM MQ classes for .NET Standard
Sınıf Adları (API'ler)	Tüm sınıflar her ağda aynı kalır.	Tüm sınıflar her ağda aynı kalır.
İşletim Sistemi	Windows	Windows Dockerized konteynerleri V 9.1.2 Linux V 9.1.3 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, .NET Standard içinde desteklenmez. app.config yerine ortam değişkenlerini kullanmanız gerekir.
Takip edin	İzlemeyi etkinleştirmek için <b>strmqtrc</b> komutu ve yeniden dağıtılabilir istemciler için app.config dosyası kullanılır.	<b>strmqtrc</b> komutu. Yeniden dağıtılabilir pakete ilişkin izlemeyi etkinleştirmek için ortam değişkenleri kullanılmalıdır. IBM MQ .NET için, <b>MQDOTNET_TRACE_ON</b> ortam değişkeni yeniden dağıtılabilir istemciler için izlemeyi etkinleştirmek için 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.
İletim Kipleri	Yönetilen, Yönetilmeyen ve Bağ Tanımları	Yönetilen

Çizelge 74. IBM MQ classes for .NET Framework ve IBM MQ classes for .NET Standard özellikleri arasındaki farklar (devamı var)

Özellik	IBM MQ classes for .NET Framework	IBM MQ classes for .NET Standard
TLS	Windows anahtar deposu, sertifikaları saklamak için kullanılır.	<p><b>Windows</b> Windows üzerinde, 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.</p> <p><b>V 9.1.2</b> <b>Linux</b></p> <p>Linux üzerinde, sertifikaları kurmak için X509Store sınıfını kullanmanız ve .NET Core sertifikalarını aşağıdaki konuma kurmanız önerilir: ".dotnet/corefx/cryptography/x509stores".</p>
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 ' ler):

Etki alanı düzeyinde kimlik doğrulaması, IBM MQ .NET Standard sınıfları için desteklenmez. Oturum açan kullanıcı kimliği, kimlik doğrulaması için kullanılır.

In addition to the environment variable **MQDOTNET\_TRACE\_ON** used to enable trace on IBM MQ .NET Standard, other environment variables including **MQERRORPATH**, **MQLOGLEVEL**, **MQSERVER**, and so on, that are used for IBM MQ classes for .NET Framework, can be used for IBM MQ classes for .NET Standard as well. Bu değişkenler, hem IBM MQ classes for .NET Standard hem de IBM MQ classes for .NET Framework için aynı şekilde çalışır.

IBM MQ classes for .NET Standard içinde, **MQDOTNET\_TRACE\_ON** ortam değişkeni, IBM MQ izleme dizininin kullanılabilir olup olmadığını denetler. İzleme dizini kullanılabiliriyorsa, izleme kütüğü izleme dizininde yaratılır. Ancak, IBM MQ kurulu değilse, izleme dosyası yürürlükteki çalışma dizinine kopyalanır.

See “Bağımsız IBM MQ .NET istemcisinin kullanılması” sayfa 578 for more information on the variables you use for trace, including **MQTRACEPATH** and **MQTRACELEVEL**.

## MacOS üzerinde IBM MQ .NET Core uygulamalarının geliştirilmesi

**V 9.1.3**

From IBM MQ 9.1.3, IBM MQ .NET Core applications can be developed on MacOS.

IBM MQ .NET kitaplıkları, MacOS araç takımıyla birlikte paketlenmez; bu nedenle bunları Windows ya da Linux IBM MQ istemcisinden MacOS' a kopyalamanız gerekir. Daha sonra bu kitaplıkları, MacOS üzerinde IBM MQ .NET Core uygulamaları geliştirmek için kullanabilirsiniz.

Bu uygulamalar geliştirildikten sonra, Windows ya da Linux ortamlarında da desteklenebilir.

### İlgili kavramlar

“kullanma IBM MQ classes for XMS .NET Standard” sayfa 587

XMS ürününü Microsoft .NET Standard ile nasıl kullanacağınız ve IBM MQ classes for XMS .NET Framework ile IBM MQ classes for XMS .NET Standard arasındaki farklar. IBM MQ classes for XMS .NET Standard için Microsoft .NET Core önkoşulu vardır.

Linux Windows V 9.1.4 **IBM MQ classes for .NET Standard dosyasını NuGet**

### havuzundan yükleme

IBM MQ 9.1.4' den IBM MQ classes for .NET Standard , .NET Developers tarafından kolayca kullanılabilmesi için NuGet havuzundan yüklenebilir.

### Bu görev hakkında

NuGet , .NET da 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.

IBM MQ 9.1.4 olanağından, amqmdnetstd.dll kitaplığını içeren IBMMQDotnetClient NuGet paketini, tüm paket yazarları ve tüketiciler tarafından kullanılan merkezi paket havuzu olan NuGet Gallery 'den yükleyebilirsiniz.

IBMMQDotnetClient paketini karşıdan yüklemenin üç yolu vardır:

- Microsoft Visual Studio komutunu 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 dağıtılabılır pakete gelince, **MQDOTNET\_TRACE\_ON** ortam değişkenini kullanarak izlemeyi etkinleştirebilirsiniz.

### Yordam

- Microsoft Visual Studio içindeki Package Manager UI 'sini kullanarak IBMMQDotnetClient paketini karşıdan yüklemek için aşağıdaki adımları tamamlayın:

- a) .NET projesini farenin sağ düğmesiyle tıklattığınız ve **NuGet Packages** öğesini seçin.
- b) **Göz At** etiketini tıklattığınız ve “IBMMQDotnetClient” için arama yapın.
- c) Paketi seçin ve **Kur** düğmesini tıklattığınız.

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, IBM MQDotnetClient paketini <https://www.nuget.org/packages/IBM MQDotnetClient> adresinden yükleyin.

### İlgili görevler

“Downloading IBM MQ classes for XMS .NET Standard from the NuGet repository” sayfa 590  
From IBM MQ 9.1.4, the IBM MQ classes for XMS .NET Standard are available for downloading from the NuGet repository, so that they can be easily consumed by .NET Developers.

### İlgili başvurular

[IBM MQ Client for .NET lisans bilgileri](#)

## Windows kurma IBM MQ classes for .NET Framework

Örnekler de içinde olmak üzere IBM MQ classes for .NET Framework, IBM MQ ile birlikte kurulur. Windows üzerinde Microsoft .NET Framework ile ilgili bir önkoşul vardır.

IBM MQ classes for .NET Framework ' un en son sürümü, *Java ve .NET Messaging ve Web Hizmetleri* özelinde standart IBM MQ kurulumunun bir parçası olarak varsayılan olarak kurulur. Kuruluş yönergeleri için bkz. [Installing IBM MQ server on Windows](#) ya da [Installing an IBM MQ client on Windows systems](#).

**V 9.1.0** From IBM MQ 9.1, to run IBM MQ classes for .NET Framework you must install Microsoft .NET Framework V4.5.1 or later.

Microsoft .NET Framework V3.5 ile derlenen var olan uygulamalar, uygulamanın app.config dosyasına şu biçim imi eklenerek yeniden derlenmeden çalıştırılabilir:

```
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.1"/>
  </startup>
</configuration>
```

**Not:** IBM MQ ürününü kurmadan önce Microsoft .NET Framework V4.5.1 ya da üstü kurulu değilse, IBM MQ ürünü hatasız olarak devam eder, ancak IBM MQ classes for .NET kullanılabilir durumda değildir. If the .NET Framework is installed after you install IBM MQ, then the IBM MQ .NET assemblies must be registered by running the *WMQInstallDir\bin\amqiRegisterdotNet.cmd* script, where *WMQInstallDir* is the directory where IBM MQ is installed. Bu komut dosyası, gerekli düzenekleri Global Assembly Cache (GAC) içine kurar. Alınan eylemleri kaydetmek için %TEMP% dizininde oluşturulan *amqi\*.log* dosyaları kümesi. .NET , daha önceki bir sürümden v4.5.1 ya da daha yüksek bir sürümüne yükseltilmişse, *amqiRegisterdotNet.cmd* komut kütüğünü yeniden çalıştırmak gerekli değildir; örneğin, .NET V3.5.

Birden çok kuruluş ortamında, IBM MQ classes for .NET 'u bir destek paketi olarak önceden kurduysanız, destek paketini ilk kaldırmadığınız sürece IBM MQ ' ı 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 534](#).

For information about using the IBM MQ custom channel for the Microsoft WCF with .NET, see [“IBM MQ ile Microsoft Windows Communication Foundation uygulamalarının geliştirilmesi” sayfa 1292](#)

### İlgili kavramlar

“kurma IBM MQ classes for .NET Standard” sayfa 530

Örnekler de dahil olmak üzere IBM MQ 9.1.1, IBM MQ classes for .NET Standard'den Windows' ta IBM MQ ile kurulur. **V 9.1.2** IBM MQ 9.1.2, IBM MQ classes for .NET Standard , Linux platformlarında da kullanılabilir. There is a prerequisite of Microsoft .NET Core for IBM MQ classes for .NET Standard.

## .NET için örnek uygulamalar

Kendi .NET uygulamalarınızı çalıştırmak için, örnek uygulamalar yerine uygulama adınızı yerine koyarak, doğrulama programlarına ilişkin yönergeleri kullanın.

Aşağıdaki örnek uygulamalar sağlanır:

- Bir put iletisi uygulaması
- İleti alma uygulaması
- 'merhaba dünya' uygulaması
- Bir yayınlama/abone olma uygulaması
- İleti özelliklerini kullanan bir uygulama

Bu örnek uygulamaların tümü C# dilinde sağlanır ve bazıları C++ dilinde ve Visual Basic 'te de sağlanır. Uygulamaları .NET tarafından desteklenen herhangi bir dilde yazabilirsiniz.

#### **"put message" program SPUT (nmqsput.cs, mmqsput.cpp, vmqsput.vb)**

Bu program, adı belirtilen bir kuyruğa nasıl ileti konacağını gösterir. Programda üç parametre vardır:

- Kuyruğun adı (gerekli), örneğin, SYSTEM.DEFAULT.LOCAL.QUEUE
- Kuyruk yöneticisinin adı (isteğe bağlı)
- Bir kanalın tanımlaması (isteğe bağlı); örneğin, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Kuyruk yöneticisi adı verilmezse, kuyruk yöneticisi varsayılan olarak varsayılan yerel kuyruk yöneticisine varsayılan değer olarak ayarlanır. Bir kanal tanımlandıysa, bu, MQSERVER ortam değişkeniyle aynı biçime sahiptir.

#### **"İleti al" program SGET (nmqsget.cs, mmqsget.cpp, vmqsget.vb)**

Bu program, adı belirtilen kuyruktan nasıl ileti alacağını gösterir. Programda üç parametre vardır:

- Kuyruğun adı (gerekli), örneğin, SYSTEM.DEFAULT.LOCAL.QUEUE
- Kuyruk yöneticisinin adı (isteğe bağlı)
- Bir kanalın tanımlaması (isteğe bağlı); örneğin, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Kuyruk yöneticisi adı verilmezse, kuyruk yöneticisi varsayılan olarak varsayılan yerel kuyruk yöneticisine varsayılan değer olarak ayarlanır. Bir kanal tanımlandıysa, bu, MQSERVER ortam değişkeniyle aynı biçime sahiptir.

#### **"Merhaba Dünya" programı (nmqwrlid.cs, mmqwrlid.cpp, vmqwrlid.vb)**

Bu program, bir iletinin nasıl yerleştireceğini ve nasıl alacağını gösterir. Programda üç parametre vardır:

- Bir kuyruğun adı (isteğe bağlı); örneğin, SYSTEM.DEFAULT.LOCAL.QUEUE YA DA SYSTEM.DEFAULT.MODEL.QUEUE
- Kuyruk yöneticisinin adı (isteğe bağlı)
- Bir kanal tanımlaması (isteğe bağlı); örneğin, SYSTEM.DEF.SVRCONN/TCP/hostname(1414)

Kuyruk adı verilmezse, ad varsayılan olarak SYSTEM.DEFAULT.LOCAL.QUEUE. Kuyruk yöneticisi adı verilmezse, kuyruk yöneticisi varsayılan olarak varsayılan yerel kuyruk yöneticisine varsayılan değer olarak ayarlanır.

#### **"Publish/subscreen" programı (MQPubSubSample.cs)**

Bu program, IBM MQ yayınlama/abone olma olanağını nasıl kullanacağını gösterir. Yalnızca C# içinde sağlanır. Programın iki değiştirgesi vardır:

- Kuyruk yöneticisinin adı (isteğe bağlı)
- Bir kanal tanımlaması (isteğe bağlı)

#### **"İleti özellikleri" programı (MQMessagePropertiesSample.cs)**

Bu program ileti özelliklerinin nasıl kullanılacağını gösterir. Yalnızca C# içinde sağlanır. Programın iki değiştirgesi vardır:

- Kuyruk yöneticisinin adı (isteğe bağlı)
- Bir kanal tanımlaması (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 temsil eder.

### C#

```
MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqswrld.cs
MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqspu.cs
MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\nmqsgt.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\mmqspu.cpp
MQ_INSTALLATION_PATH\Tools\dotnet\samples\mcp\mmqsgt.cpp
```

### Visual Basic

```
MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqswrld.vb
MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqspu.vb
MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\vmqsgt.vb
MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqswrld.vb
MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqspu.vb
MQ_INSTALLATION_PATH\Tools\dotnet\samples\vb\xmqsgt.vb
```

## Örnek uygulamaların oluşturulması

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 çizgi 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\bldmcpamp.bat
```

bldmcpamp.bat dosyası, bu örnek programı oluşturmak için gerekli olan her örnek için bir çizgi içerir:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

Bu uygulamaları Microsoft Visual Studio 2003/.NET SDKv1.1.üzerinde derlemek istiyorsanız, derleme komutunu aşağıdaki şekilde değiştirin:

```
cl /clr:oldsyntax MQ_INSTALLATION_PATH\bin mmqwrld.cpp
```

şu ürünü geçir

```
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 çizgi içerir:

```
vbc /r:System.dll /r: MQ_INSTALLATION_PATH\bin\amqmdnet.dll /out:vmqwrld.exe vmqwrld.vb
```

## IBM MQ komutunu Microsoft .NET Core ile kullanma için örnekler

### V 9.1.1

IBM MQ 9.1.1, IBM MQ , Windows Core for IBM MQ .NET uygulamaları için .NET Core olanağını destekler. Örnekler de içinde olmak üzere IBM MQ classes for .NET Standard, standart IBM MQ kurulumunun bir parçası olarak varsayılan olarak kurulur.

IBM MQ .NET için örnek uygulamalar, &MQINSTALL\_PATH&/samp/dotnet/samples/cs/core/base için kurulur. Ayrıca, örnekleri derlemek için kullanılacak bir komut dosyası da sağlanır.

Sağlanan build.bat dosyalarını kullanarak örnekleri oluşturabilirsiniz. Windows üzerinde aşağıdaki konumdaki her bir örnek için bir build.bat vardır:

- MQ\tools\dotnet\samples\cs\core\base\SimpleGet
- MQ\tools\dotnet\samples\cs\core\base\SimplePut

### V 9.1.2

### Linux

From IBM MQ 9.1.2, IBM MQ also supports .NET Core for applications in Linux environments.

For more information about using IBM MQ with Microsoft .NET Core, see [“kurma IBM MQ classes for .NET Standard” sayfa 530.](#)

## Kuyruk yöneticinizin TCP/IP istemci bağlantılarını kabul etmek için yapılandırılması

İstemcilerden gelen bağlantı isteklerini kabul etmek için bir kuyruk yöneticisi yapılandırın.

### Bu görev hakkında

Bu kısımda, bir kuyruk yöneticisinin TCP/IP istemci bağlantılarını kabul edecek şekilde yapılandırılmasına ilişkin temel adımlar açıklanmaktadır. 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 alınmadığından, yalnızca bir kum havuzu ortamında kullanılmak üzere tasarlanmıştır. Bir üretim sistemi için, TLS ya da bir güvenlik çıkışı kullanmayı düşünün. Ek bilgi için [Securing IBM MQ](#) başlıklı konuya bakın.

2. Dinleyici başlatma:

```
runmqclsr -t tcp [-m qmname ] [-p portnum ]
```

**Not:** Köşeli ayrıçlar isteğe bağlı deęiřtirgeleri belirtir; varsayılan kuyruk yöneticisi için *qname* gerekli deęildir ve varsayılan deęer (1414) kullanıyorsanız, *kapı-no* kapı numarası gerekli deęildir.

## .NETiçinde dağıtımli hareketler

Dağıtımli hareketler ya da genel hareketler, istemci uygulamalarının bir işlemdaki iki ya da daha çok aęa baęlı sisteme çeřitli veri kaynaklarını içermesine olanak saęlar.

Dağıtımli işlemlerde, bir hareket yöneticisi iki ya da daha fazla kaynak yöneticisi arasındaki hareketi düzenler ve yönetir.

Hareketler tek faz ya da iki aşamalı kesinleştirme işlemi olabilir. Tek aşamalı kesinleştirme, hareket ve iki aşamalı kesinleştirme işleminde yalnızca bir kaynak yöneticisinin katılacağı bir işlemdir ve harekette yer alan birden çok kaynak yöneticisi vardır. İki aşamalı kesinleştirme işleminde, hareket yöneticisi, tüm kaynak yöneticilerinin kesinleştirilmeye hazırlanıp hazırlanmadığını denetlemek için bir hazırlama çağrısı gönderir. Tüm kaynak yöneticilerinden alındı bildirimini alındığında, kesinleştirme çağrısı yayınlanır. Ters durumda, işlemin tamamında geriye işleme gerçekleşir. Daha fazla ayrıntı için bkz. [Hareket yönetimi ve destek](#). Kaynak yöneticileri, harekete katılmalarının işlem yöneticilerini bilgilendirmelidir. Kaynak yöneticisi, katılımının hareket yöneticisine bilgi veriyorsa, hareket kesinleştirme ya da geri alma işlemi olduğunda, kaynak yöneticisi hareket yöneticisinden geri çağrılar alır.

IBM MQ .NET sınıfları, yönetilmeyen ve sunucu baę tanımları kipi baęlantılarında dağıtımli hareketleri önceden destekler. In these modes, IBM MQ .NET classes delegates all its calls to C extended transaction client, which manages the transaction processing on behalf of .NET.

IBM MQ.NET classes now support distributed transactions in managed mode where IBM MQ .NET Classes uses System.Transactions namespace for the distributed transactions support. System.Transactions altyapısı, IBM MQdahil olmak üzere tüm kaynak yöneticilerinde başlatılan işlemleri destekleyerek işlemsel programlama basit ve verimli hale getirir. IBM MQ .NET uygulaması, .NET örtük hareket programlama ya da belirtik hareket programlama modeli kullanarak ileti alabilir ve alabilir. Örtük işlemlerde, hareket sınırları, ne zaman kesinleştirileceğine, geriye işlenmeye (belirtik işlemler için) karar veren uygulama programı tarafından yaratılır ya da işlemi tamamlar. Belirtik işlemlerde, kesinleřtirmek, geri almak ve işlemi tamamlamak isteyip istemediđinizi belirtik olarak belirtmeniz gerekir.

IBM MQ.NET , hareket yöneticisi olarak Microsoft dağıtımli hareket eşgüdümçüsü (MS DTC) kullanır ve birden çok kaynak yöneticisi arasında işlemi düzenler ve yönetir. Kaynak yöneticisi olarak IBM MQ kullanılır. TLS ' yi XA hareketleriyle kullanamayadığınızın unutmayın. CCDT ' yi kullanmanız gerekir. Daha fazla bilgi için [TLS kanallarıyla genişletilmli işlemsel istemciyi kullanmabařlıklı konuya](#) bakın.

IBM MQ.NET , X/Open Distributed Transaction Processing (DTP) modelinden sonra gelir. X/Open Distributed Transaction Processing modeli, bir satıcı konsorsiyumu olan Open Group (Açık Grup) tarafından önerilen dağıtımli bir hareket işleme modesidir. Bu model, işlem işleme ve veritabanı etki alanlarında ticari satıcı firmaların çoęu arasında bir standarttır. Ticari işlem yönetimi ürünlerinin çoęu, X/DTP modelini destekler.

## Hareket kipleri

- [“.NET yönetilen kipinde dağıtımli hareketler” sayfa 539](#)
- [Yönetilmeyen kip için dağıtımli hareketler](#)

## Çeřitli senaryolarda işlemlerin koordine edilmesi

- Bir baęlantı birkaç harekette katılabilir, ancak herhangi bir zamanda yalnızca bir işlem etkindir.
- İşlem sırasında, MQQueueManager.Baęlantı kesme çağrısı onurlandırılır. Bu durumda, işlemin geri işlenmesi istenir.
- Bir işlem sırasında, MQQueue.Close ya da MQTopic.Close çağrısı kabul edilir. Bu durumda, işlemin geri istenmesi istenir.
- Hareket sınırları, işlemin ne zaman kesinleştirileceğine, geri alınmasına (belirtik işlemler için) ya da işlemin tamamlanmasını (örtük işlemler için) karar veren uygulama programı tarafından yaratılır.

- Bir kuyruğa ya da konu çağrısına çağrı göndermeden ya da çağrı göndermeden önce istemci uygulaması beklenmeyen bir hatayla başarısız olursa, hareket geriye işlenir ve bir MQException yayınlanır.
- MQCC\_FAILED neden kodu bir Kuyruk ya da Konu çağrısı sırasında döndürülürse ya da çağrıya çağrılırsa, neden kodu ile bir MQException yayınlanır ve hareket işlenir. Bir hazırlama çağrısı hareket yöneticisi tarafından önceden verildiyse, IBM MQ .NET , işlemi zorla geri döndürerek hazırlama isteğini geri döndürür. Daha sonra, hareket yöneticisi DTC, geçerli ortam işlemlerindeki tüm kaynak yöneticileriyle birlikte yürürlükteki çalışma için geriye işleme neden olur.
- Birden çok kaynak yöneticisi içeren bir işlem sırasında, bazı ortam nedeni Koma ya da Alma çağrısının süresiz olarak askıda kalma sürmesine neden olursa, hareket yöneticisi öngörülme süreye kadar bekler. Bu süre geçtikten sonra, yürürlükteki ortam hareketlerindeki tüm kaynak yöneticileriyle tüm geçerli çalışmaların geriye işlenmesine neden olur. Hazırlık aşamasında bu süresiz bekleme gerçekleşirse, hareket yöneticisi zamanaşımına uğrayabilir ya da işlemin geriye işlendiği durumlarda, kaynak üzerinde belirsiz bir çağrı yayınlayabilir.
- İşlemleri kullanan uygulamaların SYNC\_POINT altına ileti koyması ya da iletileri almaları gerekir. Bir ileti koyma ya da alma çağrısı, SYNC\_POINT altında olmayan bir işlemsel bağlam altında yayınlanırsa, çağrıya MQRC\_UNIT\_OF\_WORK\_NOT\_STARTED neden koduyla başarısız olur.

## Microsoft.NET System.Transactions ad alanı kullanılarak Yönetilen ve Yönetilmeyen Müşteri hareketi desteği arasındaki davranış farkları

Nested Transactions have a TransactionScope inside another TransactionScope

- IBM MQ .NET fully Managed Client, iç içe geçmiş TransactionScope' u destekler
- IBM MQ .NET yönetilmeyen istemci, iç içe geçmiş TransactionScopeögesini desteklemiyor

System.Transactions' dan Bağımlı İşlemler

- IBM MQ .NET fully managed client does support the dependent transactions facility provided by System.Transactions.
- IBM MQ .NET unmanaged client does not support the dependent transactions facility provided by System.Transactions.

## Ürün Örnekleri

Yeni ürün örnekleri SimpleXAPutve SimpleXAGet , WebSphere

MQ\tools\dotnet\samples\cs\basealtında kullanılabilir. Örnekler, SystemTransactions ad alanını kullanarak Dağıtılmış Hareketler altında MQPUT ve MQGET kullanılmasını gösteren C# uygulamalarıdır. Bu örneklerle ilgili daha fazla bilgi için bkz. [“Creating simple put and get messages within a TransactionScope”](#) sayfa 542

## .NET yönetilen kipinde dağıtılmış hareketler

IBM MQ .NET sınıfları, yönetilen kipteki dağıtılmış işlemler desteği için System.Transactions ad alanını kullanır. Yönetilen kipte, MS DTC, bir harekette listelenen tüm sunucularda dağıtılmış hareketleri koordine eder ve yönetir.

IBM MQ .NET classes provide an explicit programming model based on the System.Transactions.Transaction class and an implicit programming model using the System.Transactions.TransactionScope, class where the transactions are automatically managed by the infrastructure.

## Örtük İşlem

Aşağıdaki kod parçası, bir IBM MQ .NET uygulamasının .NET örtük hareket programlamasını kullanarak bir iletiyi nasıl yerleştirdiğini açıklamalı.

```
Using (TransactionScope scope = new TransactionScope ())
{
    Q.Put (putMsg,pmo);
    scope.Complete ();
}
```

```
Q.close();
qMgr.Disconnect();}
```

### Örtük hareketin kod akışına ilişkin açıklama

Kod, *TransactionScope* ögesini yaratır ve iletiyi kapsam altına yerleştirir. Daha sonra, işlemin tamamlanmasına ilişkin işlem koordinatörü bilgilendirmek için *Complete* (Tamamlandı) çağrısını çağırır. Artık hareket eşgüdümçüsü hareketi tamamlamak için *prepare* ve *commit* konularını yayınlar. Bir sorun saptanırsa, bir *geri alma* çağrılır.

### Belirtik İşlem

Aşağıdaki kod, bir IBM MQ .NET uygulamasının .NET belirtik hareket programlama modelini kullanarak iletileri nasıl yerleştirdiğini açıklamalı.

```
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 hareket kod akışının açıklaması

Kod parçası, *CommittableTransaction* sınıfını kullanarak hareket yaratır. Bu, o kapsam altına bir ileti koyar ve daha sonra, hareketi tamamlamak için belirtik olarak *commit* (kesinleştirme) çağrılarını çağırır. Herhangi bir sorun varsa *geriye işleme* çağrılır.

### .NET yönetilmeyen kipteki dağıtılmış hareketler

IBM MQ.NET classes support unmanaged connections (client) using extended transaction client and COM+/MTS as the transaction coordinator, using either implicit or explicit transaction programming model. Yönetilmeyen kipte, IBM MQ .NET sınıfları, .NETadına işlem işlemini yöneten C genişletilmiş hareket istemcisine tüm çağrılarını yetkilendirir.

Hareket işleme, bir dış hareket yöneticisi tarafından denetlenir ve hareket yöneticisinin API 'si denetimi altında genel iş birimi eşgüdümünün eşgüdümünü sağlar. MQBEGIN, MQCMIT ve MQBACK filleri kullanılmıyor. IBM MQ .NET sınıfları, bu desteği, yönetilmeyen taşıma kipi (C istemcisi) yoluyla gösterir. Bkz. [XA uyumlu hareket yöneticilerinin yapılandırılması](#)

MTS, CICS, Tuxedo ve diğer platformlarda mevcut olduğu şekilde Windows NT ' ta aynı özellikleri sağlamak için bir hareket işleme (TP) sistemi olarak gelişmiştir. MTS kurulduğunda, Microsoft Distributed Transaction Coordinator (MSDTC Distributed Transaction Coordinator) olarak adlandırılan Windows NT ' a ayrı bir hizmet eklenir. MSDTC , ayrı veri depolarına ya da kaynaklarına yayılan işlemleri koordine eder. Çalışmak için, her veri deposunun kendi özel kaynak yöneticisini gerçekleştirmesi gerekir.

IBM MQ , DTC XA çağrılarını IBM MQ(X/Open) çağrılarına eşlemeyi yönettiği bir arabirim (özel kaynak yöneticisi arabirimi) uygulayarak MSDTC ile uyumlu hale gelir. IBM MQ , bir kaynak yöneticisinin rolünü yürütür.

COM +, bir IBM MQdeğerine erişim gibi bir bileşen, bir işlem gerekli olduğunda, COM genellikle uygun MTS bağlam nesnesiyle birlikte denetler. Bir işlem gerekiyorsa, COM DTC ' yi bilgilendirir ve bu işlem için otomatik olarak integral bir IBM MQ işlemi başlatır. Daha sonra, COM, MQMTS yazılımıyla veri ile birlikte çalışır, iletileri doldurur ve gerektiği gibi iletiler elde eder. COM ' tan elde edilen nesne eşgörünümü, verilerdeki tüm işlemler sona erdikten sonra SetComplete ya da SetAbort yönteminden geçmektedir. When the application issues SetComplete, the call signals the DTC that the application has completed the

transaction and the DTC can go ahead with the two-phase commit process. The DTC then issues calls to MQMST which in turn issues calls to IBM MQ to commit or roll back the transaction.

## Yönetilmeyen istemciyi kullanarak bir IBM MQ .NET uygulaması yazma

COM + bağlamı içinde çalıştırmak için, bir .NET sınıfı Sistem tarafından edinilmelidir. EnterpriseServices.ServicedComponent. Hizmet verilen bileşenleri kullanan düzenekler oluşturmak için kurallar ve öneriler aşağıdaki gibi olur:

**Not:** Aşağıdaki adımlar yalnızca System.EnterpriseServices kipini kullanıyorsanız anlamdır.

- COM + içinde başlatılmakta olan sınıf ve yöntem için genel (iç sınıf yok, korunan ya da statik yöntemler yok) olmalıdır.
- Sınıf ve yöntem öznitelikleri: TransactionOption özniteliği, sınıfın hareket düzeyini belirtir; bu öznitelik, işlemlerin devre dışı bırakılıp bırakılmadığını, desteklendiğini ya da gerekli olup olmadığını gösterir. ExecuteUOW() yöntemindeki AutoComplete özniteliği, işlenmeyen bir kural dışı durum yayınlamazsa, COM + 'a hareketi kesinleştirmesini bildirir.
- Düzenekğin güçlü bir şekilde adlandırılması: Düzenekğin, Global Assembly Cache (GAC) içinde güçlü bir şekilde adlandırılması ve kayıtlı olması gerekir. Düzenek, GAC 'de kaydedildikten sonra açık olarak ya da tembel kayıt tarafından COM + içinde kayıtlı.
- COM + içinde bir düzenekğin kaydedilmesi: Düzenekğin COM istemcilerine açıklanabilmesini hazırlayın. Then create a type library by using the Assembly Registration tool, regasm.exe.

```
regasm UnmanagedToManagedXa.dll
```

- Düzenekği GAC gacutil /i UnmanagedToManagedXa.dlliçine kaydedin.
- Register the assembly in COM+ by using the .NET services installer tool, regsvcs.exe. regasm.exe:tarafından yaratılan tip kitaplığına bakın.

```
Regsvcs /appname:UnmanagedToManagedXa /tlb:UnmanagedToManagedXa.tlb UnmanagedToManagedXa.dll
```

- Düzenek GAC 'ye konuşlandırılır ve daha sonra tembel kayıt tarafından COM + 'da kayıtlı olur. The .NET framework takes care of the registration after the code is run for the first time.

COM + ile System.EnterpriseServices modelini ve System.Transactions işlevini kullanarak örnek kod akışı aşağıdaki kısımlarda 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 komutunu kullanarak kod akışı örneği

```

[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();
}
}

```

### ***Creating simple put and get messages within a TransactionScope***

Ürün örnek C# uygulamaları IBM MQ içinde bulunur. Bu basit uygulamalar, bir TransactionScope içinde ileti koymanın ve iletilerin yerleştirilmesini gösterir. Görevin sonunda, bir kuyruktan ya da konudan iletiler yerleştirebilir ve iletiler alabilirsiniz.

### **Başlamadan önce**

XA İşlemleri için MSDTC hizmeti çalışıyor ve etkinleştirilmelidir.

### **Bu görev hakkında**

Örnek, yalın bir uygulamadır ( SimpleXAPut ve SimpleXAGet). SimpleXAPut ve SimpleXAGet programları, IBM MQ içinde kullanılabilir olan C# uygulamalarıdır. SimpleXAPut demonstrates using MQPUT, under Distributed Transactions using SystemTransactions namespace. SimpleXAGet demonstrates using MQGET, under Distributed Transactions using SystemTransactions namespace.

SimpleXAPut , MQ\tools\dotnet\samples\cs\base içinde bulunur

### **Yordam**

Uygulamalar, tools\dotnet\samples\cs\base\binkomutundan komut satırı değiştirgelerile 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]
```

parametrelerin bulunduğu yer:

#### **-destinationURI**

Bu, kuyruk ya da konu olabilir. Bir kuyruk için, queue://queueName olarak ve bir konunun topic://topicNameolarak belirtilmesini belirtin.

#### **-host**

Bu ad, localhost (yerel anasistem) ya da bir IP adresi gibi bir anasistem adı olabilir.

#### **-port**

Kuyruk yöneticisinin çalışmakta olduğu kapı.

#### **-channel**

Kullanılmakta olan bağlantı kanalı. Varsayılan değer SYSTEM.DEF.SVRCONN

#### **-transaction**

Hareket sonucu; örneğin, kesinleştirme ya da geriye işleme gibi.

#### **-mode**

Örneğin, yönetilen ya da yönetilmeyen taşıma kipi.

#### **-numberOfMsgs**

İletilerin 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' taki işlemlerin kurtarılıyor**

Bu bölümde, yönetilen kip kullanılarak IBM MQ .NET XA içindeki işlemlerin kurtarılabilmesinin ele alınmıştır.

### **Genel Bakış**

Dağıtımli hareket işleme işlemlerinde işlemler başarıyla tamamlanabilir. Ancak, bir işlemin birçok nedenden dolayı başarısız olabileceği senaryolar da olabilir. Bu nedenler arasında bir sistem arızası, donanım hatası, ağ hatası, yanlış ya da geçersiz veriler, uygulama hataları ya da doğal ya da insan yapımı olağanüstü durumlar bulunabilir. İşlem başarısızlıklarının önlenmesi olanaklı değildir. Dağıtılmış hareket sisteminin bu hataları işleme yeteneğine sahip olması gerekir. Hata ortaya çıktığında hataları saptayabilir ve düzeltebilmelidir. Bu işlem, İşlem Kurtarma olarak bilinir.

Dağıtımli hareket işleme işleminin önemli bir yönü, eksik ya da belirsiz hareketlerin kurtarılması. Kurtarılması, belirli bir işlemin İş Birimi bölümü kurtarılincaya kadar kilitli tutulmak üzere çalıştırılmalıdır. Microsoft.NET from its System.Transactions class library provides the option for recovering incomplete/in-doubt transactions. This recovery support expects Resource Manager to maintain the transaction logs and run the recovery when in need.

## Kurtarma Modeli

Microsoft .NET hareket kurtarma modelinde, Transaction Manager (System.Transactions ya da Microsoft Distributed Transaction Coordinator (MS DTC) ya da her ikisi), başlatılır, koordinatlar ve hareket kurtarma işlemini denetler. OLE Tx İletişim Kuralı ( Microsoft XA iletişim kuralı) tabanlı Kaynak Yöneticileri, DTC ' yi sürücü, koordinat ve bunların kurtarılması için yapılandırma seçeneklerini sağlar. Bunu yapmak için, Kaynak Yöneticileri 'nin XA\_Switch 'i MS DTC ile yerel arabirim kullanılarak kaydettirmesi gerekir.

XA\_Switch provides the entry points of XA functions like xa\_start, xa\_end, and xa\_recover in the Resource Manager to the Distributed Transaction Coordinator.

### Microsoft Distributed Transaction Coordinator (DTC) olanağını kullanarak kurtarma:

Microsoft Distributed Transaction eşgüdümçüsü, iki tür kurtarma işlemi sağlar.

#### Soğuk Kurtarma

Bir XA kaynak yöneticisine bağlantı açıksa, 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.

#### Sıcak Kurtarma

XA kaynak yöneticisi ya da ağ başarısız olduğu için, hareket yöneticisi ile XA kaynak yöneticisi arasındaki bağlantı başarısız olursa, hareket yöneticisi devam ederse, çalışırken kurtarma işlemi gerçekleştirilir. Başarısızlığın ardından, hareket yöneticisi belirli aralıklarla XA kaynak yöneticisine yeniden bağlanmayı dener. Bağlantı yeniden kurulduğunda, hareket yöneticisi XA kurtarma işlemini başlatır.

System.Transactions ad alanı, hareket yöneticisi olarak MS DTC ' ye dayalı olarak yönetilen dağıtılmış hareketlerin yönetilen somutlamasını sağlar. MS DTC ' nin yerel arabirimiyle aynı özellikleri sağlar, ancak tam olarak yönetilen ortamdır. Tek fark, işlem kurtarma işlemleriyle ilgilidir. System.Transactions , Kaynak Yöneticilerinin kurtarmayı kendi kendilerine kullanmasını ve daha sonra, İşlem Yöneticileri (MS DTC) ile koordinat etmesini bekler. Resource Manager , belirli bir tamamlanmamış işlemin kurtarılması için sormalı ve İşlem Yöneticisi bu işlemin gerçek sonucuna dayalı olarak onu ve koordinatları kabul eder.

#### IBM MQ .NET için işlem kurtarma işlemi

Bu bölümde, dağıtılmış işlemlerin IBM MQ .NET sınıflarıyla nasıl kurtarılabildiği ele alınmıştır.

## Genel Bakış

Tamamlanmamış bir hareketi kurtarmak için, kurtarma bilgilerinin gerekli olduğunu kabul edin. İşlem kurtarma bilgileri, kaynak yöneticileri tarafından depolamak üzere günlüğe kaydedilmelidir. IBM MQ .NET sınıfları, benzer bir yolu izler. İşlem kurtarma bilgileri, SYSTEM.DOTNET.XARECOVERY.QUEUE.

IBM MQ .NET ' da hareket kurtarma işlemi, iki aşamalı bir işlemdir.

1. İşlem kurtarma bilgilerinin günlüğe kaydedilmesi.
  - Hazırlama aşaması sırasında, her işlem için, kurtarma bilgilerini içeren kalıcı bir ileti SYSTEM.DOTNET.XARECOVERY.QUEUE.
  - Kesinleştirme çağrısı başarılı olursa, ileti silinir.
2. Bir Monitor uygulaması WmqDotnetXAMonitor kullanılarak işlem kurtarılıyor.
  - WmqDotnetXAMonitor is a .NET managed application that processes messages in SYSTEM.DOTNET.XARECOVERY.QUEUE and recovers incomplete transactions

MCA iletiyi hedef kuyruğa koyamıyorsa, özgün iletiyi içeren bir kural dışı durum raporu oluşturur ve bunu, özgün iletide belirtilen yanıtlama kuyruğuna gönderilecek bir iletim kuyruğuna koyar. (Yanıtlama kuyruğu, MCA ile aynı kuyruk yöneticisiyse, ileti bir iletim kuyruğuna değil, doğrudan o kuyruğa konadır.)



## SYSTEM.DOTNET.XARECOVERY.QUEUE

Bu, tamamlanmamış hareketlere ilişkin işlem kurtarma bilgilerini içeren bir sistem kuyruğudur. Bu kuyruk, bir kuyruk yöneticisi yaratıldığında yaratılır.

**Not:** SYSTEM.DOTNET.XARECOVERY.QUEUE KUYRUK

### WMQDotnetXAMonitor Uygulaması

IBM MQ .NET XA Monitor uygulaması, verili bir kuyruk yöneticisini izler ve varsa, tamamlanmamış hareketleri kurtarır. Aşağıda eksik işlem olarak kabul edilir ve kurtarılır:

#### Tamamlanmamış Hareketler

- Hareket hazırlandıysa, ancak zamanaşımı süresi içinde COMMIT işlemi tamamlanmadıysa.
- Hareket hazırlandıysa, ancak IBM MQ kuyruk yöneticisi sona ermiş olabilir.
- İşlem hazırlandıysa, ancak İşlem Yöneticisi sona erdiyse.

Monitor uygulaması, IBM MQ .NET istemci uygulamanızın çalıştığı aynı sistemden çalıştırılmalıdır. Aynı kuyruk yöneticisine bağlanan birden çok sistemde çalışan uygulamalar varsa, izleme uygulaması tüm sistemlerden çalıştırılmalıdır. Her istemci makinenin, uygulamayı kurtarmak için çalışan bir izleme uygulaması varsa, her izleme programı, yürürlükteki izleme programının yerel MS DTC ' nin yeniden listeleyebilmesi ve tamamlanabilmesi için eşgüdümleme yaptığı işleme karşılık gelen iletiyi belirleyebilmelidir.

*IBM MQ .NET için işlem kurtarma kullanım senaryoları*

İşlemlerin kurtarılması gerekebileceği birkaç farklı kullanım senaryosu vardır.

- **Tek DTC ve tek kuyruk yöneticisi yönetim ortamı kullanan IBM MQ Uygulaması:** Bu kullanım durumunda, hareket altında kuyruk yöneticisine ve çalışan İş Birimini (UoW) bağlarken ve işlem başarısız olursa ve tamamlanamazsa, izleme uygulaması hareketi kurtarır ve işlemi tamamlar.

Bu kullanım durumunda, tek bir kuyruk yöneticisi hareketlerle ilişkilendirildiği için, izleme uygulamasının tek bir eşgörünümü çalışır.

- **Tek DTC ve tek kuyruk yöneticisi yönetim ortamını kullanan birden çok IBM MQ uygulaması:** Bu kullanım senaryosunda, tek DTC altında birden çok WMQ uygulaması vardır ve bunların tümü aynı kuyruk yöneticisine bağlanıyor ve hareketler altında UoW çalıştırılıyor.

Hareketler başarısız olursa ve tamamlanamazsa, izleme uygulaması bunları kurtarır ve tüm uygulamalarla ilgili işlemleri tamamlar.

Bu kullanım durumunda, hareketlerde bir kuyruk yöneticisi kullanıldığı için tek bir Monitor uygulaması çalışır.

- **Birden çok IBM MQ Uygulaması, birden çok DTCs, farklı Kuyruk Yöneticisi örnekleri:** Bu kullanım senaryosunda, farklı DTM ' ler altında birden çok WMQ uygulaması vardır (yani, her uygulama farklı bir makinede çalıştırılır) ve farklı kuyruk yöneticilerine bağlanıyor.

Hata oluşursa ve işlem tamamlanamazsa, Monitor uygulaması, DTC adresini saptamak için iletide yer alan TransactionManager' ın neresinde olduğunu denetler. TransactionManager, izleme programının çalışmakta olduğu DTC adresiyle eşleşiyorsa, kurtarma işlemini tamamlar; başka bir işlem, DTC ' ye karşılık gelen ileti bulununcaya kadar aramaya devam eder.

Bu kullanım durumunda, her istemcinin işlemlerde kullandığı kendi kuyruk yöneticisine sahip olduğu için, her istemci için (kullanıcı ya da bilgisayar) çalışan bir izleme programı uygulamasının tek bir eşgörünümü olacaktır.

- **Birden çok IBM MQ Uygulaması, birden çok DTCs, birden çok aynı Kuyruk Yöneticisi örneği:** Bu kullanım senaryosunda, farklı DTM ' ler altında birden çok WMQ uygulaması vardır (her bir uygulama farklı bir makinede çalıştırılır) ve tümü aynı kuyruk yöneticisine bağlanıyor.

If failure occurs and transaction becomes incomplete, monitor application verifies the TransactionManagerWhereabouts in the message to check if the DTC address and value match with

the DTC under which the monitor is running. Her iki değer de eşleşirse, kurtarma işlemi, DTC ' ye karşılık gelen iletiyi buluncaya kadar aramayı sürdürmeye devam eder.

Bu kullanım durumunda, her istemcinin hareketlerde kullandığı kendi kuyruk yöneticisi ilişkilendirmesi olduğu için, her istemci için (kullanıcı ya da bilgisayar) çalışan tek bir izleme programı eşgörünümü olacaktır.

- **Birden çok IBM MQ Uygulaması, tek DTC, farklı Kuyruk Yöneticisi örnekleri:** Bu kullanım durumunda, tek bir DTC altında birden çok WMQ uygulaması vardır (bir bilgisayarda, çalışan birden çok WMQ uygulaması vardır) ve farklı kuyruk yöneticilerine bağlanıyor.

Hareket başarısız olursa ve tamamlanamazsa, Monitor Application işlemi kurtarır.

Bu kullanım durumunda, her uygulamanın hareketlerde kullanılan kendi kuyruk yöneticisi olduğu ve her birinin kurtarılması gerektiği için, kuyruk yöneticisi olarak çalışan izleme uygulamasının birden çok eşgörünümü olacaktır.

**Not:** İzleme uygulaması arka planda çalışmıyorsa, uygulamayı başlatabilirsiniz.

*WMQDotnetXAMonitor uygulamasının kullanılması*

WMQDotnetXAMonitor uygulamasının el ile çalıştırılması gerekir. Her an başlatılabilir. You can start it when you see the messages on the SYSTEM.DOTNET.XARECOVERY.QUEUE or you can keep it running in the background before you do any transactional work with the applications that are written using IBM MQ .NET classes.

Monitor uygulamasını başlatmak için aşağıdaki komutu kullanın.

```
WmqDotnetXAMonitor.exe -m QueueManagerName -n ConnectionName -c ChannelName -i
```

Where

- **-m QueueManagerAdı**  
Kuyruk yöneticisi adı.  
İsteğe Bağlı
- **-n ConnectionName**  
Anasistem (kapı) biçiminde bağlantı adı. Bağlantı adı birden çok bağlantı adı içerebilir. Virgülle ayrılmış listelerde birden çok bağlantı adı verilmelidir; örneğin, localhost (1414), localhost (1415), localhost (1416). Monitor uygulaması, virgülle ayrılmış listede belirtilen bağlantı adlarının her biri için kurtarma işlemini çalıştırır.
- **-c ChannelName**  
Kanal adı.
- **-h**  
Buluşsal şube tamamlıyor.  
İsteğe Bağlı

İzleyici uygulaması aşağıdaki işlemleri gerçekleştirir:

1. SYSTEM.DOTNET.XARECOVERY.QUEUE (KUYRUK) 100 saniyenin bir aralığında.
2. Kuyruk derinliği sıfırdan büyükse, XA monitörü ileti kuyruğunu tarar ve iletinin tamamlanmamış işlem ölçütlerine uygun olup olmadığını denetler.
3. İletilerin herhangi biri tamamlanmamış işlem ölçütlerine uyarsa, monitörü dışarı çeker ve işlem kurtarma bilgilerini alır.
4. Daha sonra, kurtarma bilgilerinin yerel MS DTC ile ilgili olup olmadığını belirler. If evet ise, işlemi geri almak için devam eder. Ters durumda, sonraki iletiye göz atmak için geri döner.
5. Daha sonra, eksik işlemi kurtarmak için kuyruk yöneticisine çağrı yapar.

### *WmqDotNETXAMonitor uygulama yapılandırma dosyası ayarları*

Uygulamayı izlemek için, uygulama yapılandırma dosyası kullanılarak girişler de sağlanabilir. Örnek bir uygulama yapılandırma dosyası IBM MQ .NET ile birlikte teslim edilir. Bu dosya gereksinimlerinize göre değiştirilebilir.

Uygulama Yapılandırması dosyası, giriş değerlerini dikkate alırken en yüksek önceliği alır. Giriş değerleri hem komut satırı, hem de Uygulama Yapılandırması dosyasında sağlandıysa, uygulama yapılandırmasındaki değerler dikkate alınır.

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>
```

### *WmqDotNetXAMonitor Uygulama günlüğü*

Monitor Application, Monitor 'un ilerleme durumunu ve işlem kurtarma durumunu günlüğe kaydetmek için uygulama dizininde bir günlük kütüğü yaratır. Günlüğe kaydetme işlemi, kurtarma işlemi yürütülmekte olan yürürlükteki kuyruk yöneticisini göstermek için bağlantı adı ve kanal ayrıntılarıyla başlar.

Kurtarma işlemi başladıktan sonra, hareket kurtarma iletisinin MessageId , tamamlanmamış işlemin TransactionId ve Transaction Manager Eşgüdümleme başına işlemin gerçek sonucunun günlüğe kaydedileceği bir kez.

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**

To use IBM MQ classes for .NET to access IBM MQ queues, you write programs in any language supported by .NET containing calls that put messages onto, and get messages from, IBM MQ queues.

IBM MQ belgeleri yalnızca C#, C++ ve Visual Basic dillerinde bilgi içerir.

Bu konu grubunda, uygulamaların IBM MQ sistemleriyle etkileşimde bulunabilmelerine yardımcı olacak bilgiler sağlanmaktadır. Tek tek sınıflara ilişkin ayrıntılar için [IBM MQ .NET sınıfları ve arabirimleri](#) başlıklı konuya bakın.

## Bağlantı farkları

IBM MQ.NET programına ilişkin programınız, kullanmak istediğiniz bağlantı kiplerinde bazı bağımlılıklara sahiptir.

When IBM MQ classes for .NET are used as a managed client, there are a number of differences from a standard IBM MQ MQI client, as some features are not available to a managed client.

IBM MQ.NET determines which connection type to use from the settings that you specify for the connection name, channel name, the customization value `NMQ_MQ_LIB` and the property `MQC.TRANSPORT_PROPERTY`.

### Yönetilen istemci bağlantıları

When IBM MQ classes for .NET are used as a managed client, there are a number of differences from a standard IBM MQ MQI client.

Yönetilen bir istemci için aşağıdaki özellikler kullanılamaz:

- Kanal sıkıştırması
- Kanal çıkışı zincirleme

Bu özellikleri yönetilen bir istemciyle kullanmaya çalışırsanız, bu bir `MQException` döndürür. Hata, bağlantının istemci ucunda saptanırsa, `MQRC_ENVIRONMENT_ERROR` neden kodunu kullanır. Sunucu ucunda saptanırsa, sunucu tarafından döndürülen neden kodu kullanılır.

Yönetilmeyen bir istemci için yazılan kanal çıkışları işe yaramaz. Yönetilen istemci için özel olarak yeni çıkışlar yazmalısınız. `CCDT` (istemci kanal tanımlama çizelgesinde (`CCDT`) geçersiz kanal çıkışı belirtilmemesine dikkat edin.

Yönetilen kanal çıkışının adı en çok 999 karakter uzunluğunda olabilir. Ancak, kanal çıkış adını belirtmek için `CCDT` 'yi kullanırsanız, bu değer 128 karakterle sınırlıdır.

İletişim yalnızca TCP/IP üzerinde desteklenir.

`endmqm` komutunu kullanarak bir kuyruk yöneticisini durdurduğunuzda, .NET yönetilen istemcisine bir sunucu bağlantısı kanalı, sunucu bağlantısı kanallarının diğer istemcilere daha uzun süre kapanmasını sağlar.

Yönetilen IBM MQ sorun tanımlama programlarını kullanmak için `NMQ_MQ_LIB` 'i yönetilen olarak ayarladıysanız, `strmqtrc` komutunun `-i`, `-p`, `-s`, `-b` ya da `-c` parametrelerinin hiçbiri desteklenmektedir.

XA hareketlerini kullanan yönetimli bir .NET uygulaması bir z/OS kuyruk yöneticisiyle çalışmaz. Bir z/OS kuyruk yöneticisine bağlanmayı deneyen yönetilen bir .NET istemcisi bir hatayla başarısız oldu, `MQRENE` çağrısında `MQRC_UOW_ENLISTMENT_ERROR` (`mqrc=2354`). Ancak XA hareketlerini kullanan yönetilen bir .NET uygulaması, dağıtılmış kuyruk yöneticisiyle çalışacaktır.

### Kullanılacak bağlantı tipini tanımlama

Bağlantı tipi, bağlantı adı, kanal adı, uyarılama değeri `NMQ_MQ_LIB` ve özellik `MQC.TRANSPORT_PROPERTY`.

Bağlantı adını aşağıdaki gibi belirtebilirsiniz:

- Bir `MQQueueManager` oluşturucusuna açık bir şekilde:

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- By setting the properties `MQC.HOST_NAME_PROPERTY` and, optionally, `MQC.PORT_PROPERTY` in a hashtable entry on an `MQQueueManager` constructor:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Açık MQEnvironment değerleri olarak

```
MQEnvironment.Hostname
```

MQEnvironment.Port (isteğe bağlı).

- By setting the properties MQC.HOST\_NAME\_PROPERTY and, optionally, MQC.PORT\_PROPERTY in the MQEnvironment.properties hashtable.

Kanal adını aşağıdaki gibi belirtebilirsiniz:

- Bir MQQueueManager oluşturucusuna açık bir şekilde:

```
public MQQueueManager(String queueManagerName, MQLONG Options, string Channel,  
string ConnName)
```

```
public MQQueueManager(String queueManagerName, string Channel, string ConnName)
```

- MQC.CHANNEL\_PROPERTY , bir MQQueueManager oluşturucusuna ilişkin hashtable girişlerinde:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- Açık bir MQEnvironment değeri olarak

```
MQEnvironment.Channel
```

- MQC.CHANNEL\_PROPERTY , MQEnvironment.properties hashtable 'ında.

İletim özelliğini aşağıdaki gibi belirtebilirsiniz:

- MQC.TRANSPORT\_PROPERTY , bir MQQueueManager oluşturucusuna ilişkin hashtable girişlerinde:

```
public MQQueueManager(String queueManagerName, Hashtable properties)
```

- MQC.TRANSPORT\_PROPERTY , MQEnvironment.properties hashtable.

Aşağıdaki değerlerden birini kullanarak, gerek duyduğunuz bağlantı tipini seçin:

MQC.TRANSPORT\_MQSERIES\_BINDINGS -sunucu olarak bağlantı kur

MQC.TRANSPORT\_MQSERIES\_CLIENT -XA dışı istemci olarak bağlantı kurun

MQC.TRANSPORT\_MQSERIES\_XACLIENT -XA istemcisi olarak bağlan

MQC.TRANSPORT\_MQSERIES\_MANAGED -XA dışı yönetilen istemci olarak bağlan

NMQ\_MQ\_LIB uyarılama değerini, aşağıdaki çizelgede gösterildiği gibi, bağlantı tipini açık bir şekilde seçmek için ayarlayabilirsiniz.

NMQ_MQ_LIB değeri	Bağlantı tipi
mqic.dll	XA dışı bir istemci olarak bağlan
mqicxa.dll	XA istemcisi olarak bağlan
mqm.dll	Sunucu olarak ya da XA dışı bir istemci olarak bağlan
yönetilen	XA dışı yönetilen bir istemci olarak bağlan

**Not:** mqic32.dll ve mqic32xa.dll deęerleri, daha önceki yayınlarla uyumluluk sağlamak için mqic.dll ve mqicxa.dll eşanlamlıları olarak kabul edilir. Ancak, mqm.dll ve mqm.pdb , istemci paketinin yalnızca bir parçası olarak IBM WebSphere MQ 7.1 ' dan itibaren kullanılabilir.

Ortamınızda kullanılmayan bir bağlantı tipini seçerseniz, örneğin mqic32xa.dll deęerini belirtmiş ve XA desteęi yoksa, IBM MQ.NET kural dışı durum yayınlar.

NMQ\_MQ\_LIB ' nin "yönetilen" olarak ayarlanması, istemcinin yönetilen IBM MQ sorun tanılama sınamalarını, .NET veri dönüştürmesini ve dięer yönetilen düşük düzeyli IBM MQ işlevlerini kullanmasına neden olur.

NMQ\_MQ\_LIB için dięer tüm deęerler, .NET işleminin yönetilmeyen IBM MQ sorun tanılama sınamalarını ve veri dönüştürmesini ve yönetilmeyen dięer düşük düzeyli IBM MQ işlevlerini (sistemde bir IBM MQ MQI client ya da sunucu kurulu olduęu varsayılarak) kullanmasına neden olur.

IBM MQ.NET , bağlantı tipini aşıęıdaki gibi seçer:

1. MQC.TRANSPORT\_PROPERTY belirtildi, MQC.TRANSPORT\_PROPERTY.

Ancak, MQC.TRANSPORT\_PROPERTY - MQC.TRANSPORT\_MQSERIES\_MANAGED , istemci işleminin yönetiliyor olduęunu garanti etmez. Bu ayara rağmen, istemci aşıęıdaki durumlarda yönetilmiyor:

- Süreçteki başka bir iş parçasıęı MQC.TRANSPORT\_PROPERTY , MQC.TRANSPORT\_MQSERIES\_MANAGED.
- NMQ\_MQ\_LIB "yönetilen" olarak ayarlanmazsa, sorun tanılama sınamaları, veri dönüştürme ve dięer düşük düzeyli işlevler tam olarak yönetilmiyor (sistemde bir IBM MQ MQI client ya da sunucu kurulu olduęu varsayılıyor).

2. Bir bağlantı adı, kanal adı olmadan ya da bağlantı adı olmadan bir kanal adı belirlendiye, bir hata oluşur.

3. Hem bir bağlantı adı, hem de kanal adı belirtilmişse:

- NMQ\_MQ\_LIB, mqic32xa.dllolarak ayarlandıysa, XA istemcisi olarak bağlanır.
- Yönetilen olarak NMQ\_MQ\_LIB ayarlanmışsa, yönetilen istemci olarak bağlanır.
- Ters durumda, XA istemcisi olmayan bir istemci olarak bağlanır.

4. NMQ\_MQ\_LIB belirtilirse, bu, NMQ\_MQ\_LIB deęerine göre bağlanır.

5. Kurulu bir IBM MQ sunucusu varsa, sunucu olarak bağlanır.

6. Bir IBM MQ MQI client kuruluysa, XA dışı bir istemci olarak bağlanır.

7. Ters durumda, yönetilen istemci olarak bağlanır.

## **IBM MQ .NET proje şablonunu kullanma**

From IBM MQ 9.1.5, the IBM MQ .NET client offers you the ability to use a project template to assist you in developing your .NET Core applications.

### **Başlamadan önce**

Sisteminizde Microsoft Visual Studio 2017ya da daha sonraki bir sürüm ve .NET Core 2.1 bulunmalıdır.

.NET şablonunu şu şablondan kopyalamanız gerekir:

```
&MQ_INSTALL_ROOT%\tools\dotnet\samples\cs\core\base\ProjectTemplates\IBMMQ.NETClientApp.zip
```

dizini

```
&USER_HOME_DIRECTORY%\Documents\&Visual_Studio_Version%\Templates\ProjectTemplates
```

dizin, burada:

- &MQ\_INSTALL\_ROOT , kurulumunuzun kök dizidir.

- & `USER_HOME_DIRECTORY` , ana dizininidir.

Şablonu almak için Microsoft Visual Studio ' u durdurmanız ve yeniden başlatmanız gerekir.

## Bu görev hakkında

.NET proje şablonu, uygulamalarınızı geliştirmenize yardımcı olmak için kullanabileceğiniz bazı ortak kodları içerir. With the in-built code, you can connect to the IBM MQ queue manager, and perform a put or a get operation by simply modifying the properties in the in-built code.

## Yordam

1. Microsoft Visual Studio uygulamasını açın.
2. **Dosya**'yı tıklayın, ardından **Yeni** ' yi ve ardından **Proje** ' yi tıklayın.
3. *Create a new project window*(Yeni proje penceresi yarat) alanında IBM MQ .NET Client App (.NET Core) seçeneğini belirleyin ve **Next**düğmesini tıklayın.
4. *Yeni projenizi yapılandır* penceresinde, isterseniz projenizin *Proje adı* 'yı değiştirin ve .NET projesini yaratmak için **Yarat** ' ı tıklayın.

MQDotnetApp.cs , proje dosyasıyla birlikte oluşturulan dosyadır. Bu dosya, kuyruk yöneticisine bağlanan kodu içerir ve bir put ve alma işlemi gerçekleştirir.

Bağlantı özellikleri varsayılan değerlere ayarlanır:

- MQC.CONNECTION\_NAME\_PROPERTY , *localhost (1414)*olarak ayarlandı
- MQC.CHANNEL\_PROPERTY , *DOTNET.SVRCONN*değerine ayarlı

Kuyruk *Q1*olarak ayarlıdır ve bu özellikleri uygun şekilde değiştirebilirsiniz.

5. Uygulamayı derleyin ve çalıştırın.

## İlgili kavramlar

[IBM MQ bileşenleri ve özellikleri](#)

## İlgili başvurular

[.NET uygulama yürütme ortamı- Windows yalnızca](#)

## IBM MQ classes for .NET için yapılandırma dosyaları

Bir .NET istemci uygulaması, bir IBM MQ MQI client yapılandırma dosyasını kullanabilir ve yönetilen bağlantı tipini kullanıyorsanız, bir .NET uygulama yapılandırma dosyası kullanılabilir. Uygulama yapılandırma dosyasındaki ayarlar önceliğe sahiptir.

## İstemci yapılandırma dosyası

Bir IBM MQ classes for .NET istemcisi uygulaması, diğer herhangi bir IBM MQ MQI client ile aynı şekilde bir istemci yapılandırma dosyasını kullanabilir. Bu dosya genellikle *mqclient.ini*olarak adlandırılır, ancak farklı bir dosya adı belirleyebilirsiniz. İstemci yapılandırma dosyası hakkında daha fazla bilgi için bkz. [Yapılandırma dosyası kullanarak istemci yapılandırılması](#).

Bir IBM MQ MQI client yapılandırma dosyasında yalnızca aşağıdaki öznitelikler IBM MQ classes for .NET ile ilgilidir. Diğer öznitelikleri belirlerseniz, bu bir etki göstermez.

Çizelge 75. IBM MQ classes for .NET ile ilgili istemci yapılandırma dosyası öznitelikleri	
Stanza	Öznitelik
<a href="#">Kanallar</a>	CCSID
<a href="#">Kanallar</a>	ChannelDefinitionDizini
<a href="#">Kanallar</a>	ChannelDefinitionDosyası
<a href="#">Kanallar</a>	ReconDelay

Çizelge 75. IBM MQ classes for .NET ile ilgili istemci yapılandırma dosyası öznitelikleri (devamı var)

Stanza	Öznitelik
<a href="#">Kanallar</a>	DefRecon
<a href="#">Kanallar</a>	MQReconnectTimeout
<a href="#">Kanallar</a>	ServerConnectionParms
<a href="#">Kanallar</a>	Put1DefaultAlwaysSync
<a href="#">Kanallar</a>	PasswordProtection
<a href="#">ClientExitYolu</a>	ExitsDefaultYolu
<a href="#">ClientExitYolu</a>	ExitsDefaultPath64
<a href="#">MessageBuffer</a>	MaximumSize
<a href="#">MessageBuffer</a>	PurgeTime
<a href="#">MessageBuffer</a>	UpdatePercentage
<a href="#">TCP</a>	CIntRcvBufSize
<a href="#">TCP</a>	CIntSndBufSize
<a href="#">TCP</a>	IPAddressVersion
<a href="#">TCP</a>	KeepAlive

Uygun ortam değişkenini kullanarak bu özniteliklerden herhangi birini geçersiz kılabilirsiniz.

## Uygulama yapılandırma dosyası

If you are running with the managed connection type, you can also override the IBM MQ client configuration file and the equivalent environment variables using the .NET application configuration file.

.NET uygulaması yapılandırma dosyası ayarları yalnızca yönetilen bağlantı tipiyle çalıştırıldığında ve diğer bağlantı tiplerinde yoksayılr.

The .NET application configuration file and its format are defined by Microsoft for general use within the .NET framework, but the particular section names, keys and values mentioned in this documentation are specific to IBM MQ.

.NET uygulaması yapılandırma dosyasının biçimi, *bölmelersayıdır*. Her bir bölüm bir ya da daha çok *anahtar* içerir ve her anahtarın ilişkili bir *değer* vardır. Aşağıdaki örnekte, TCP/IP KeepAlive özelliğini denetlemek için bir .NET uygulaması yapılandırma dosyasında kullanılan kısımlar, 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 kısım adlarında ve anahtarlarında kullanılan anahtar sözcükler, istemci yapılandırma dosyasında tanımlı olan stanzalar ve özniteliklere ilişkin anahtar sözcüklerle tam olarak eşleşir.

The section <configSections> must be the first child element of the <configuration> element.

Ek 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 ortamını ayarlama

Bir kuyruk yöneticisine bağlanmak için istemci bağlantısını kullanmadan önce, IBM MQ ortamını ayarlamanız gerekir.

**Not:** Sunucu bağ tanımları kipinde IBM MQ classes for .NET kullanıldığında bu adım gerekli değildir.

.NET programlama arabirimi, NMQ\_MQ\_LIB uyarılma değerini kullanmanızı sağlar, ancak bir sınıf MQEnvironment (MQEnvironment) da içerir. Bu sınıf, aşağıdaki listede yer alan, bağlantı girişimi sırasında kullanılacak ayrıntıları belirtmenizi sağlar:

- Kanal adı
- Anasistem adı
- Kapı numarası
- Kanal çıkışları
- SSL parametreleri
- Kullanıcı kimliği ve parola

MQEnvironment sınıflarıyla ilgili tam bilgi için bkz. [MQEnvironment.NET sınıfı](#)

Kanal adını ve anasistem adını belirtmek için aşağıdaki kodu kullanın:

```
MQEnvironment.Hostname = "host.domain.com";
MQEnvironment.Channel = "client.channel";
```

Varsayılan olarak, istemciler 1414 numaralı kapıdaki bir IBM MQ dinleyicisine bağlanmayı dener. Farklı bir kapı belirtmek için şu kodu kullanın:

```
MQEnvironment.Port = nnnn;
```

## Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme

IBM MQ ortamını yapılandırdığınızda, bir kuyruk yöneticisine bağlanmaya hazırsınız.

Kuyruk yöneticisine bağlanmak için, MQQueueManager sınıfının yeni bir eşgörünümünü yaratın:

```
MQQueueManager queueManager = new MQQueueManager("qMgrName");
```

Kuyruk yöneticisinden bağlantıyı kesmek için, kuyruk yöneticisinden Disconnect yöntemini çağırın:

```
queueManager.Disconnect();
```

Kuyruk yöneticisine bağlanma girişimi sırasında kuyruk yöneticisine ilişkin sorgulama (inq) yetkinizin olması gerekir. Sorgu yetkisi olmadan, bağlantı kurma girişimi başarısız olur.

Disconnect yöntemini çağırırsanız, o kuyruk yöneticisi aracılığıyla eriştiğiniz tüm açık kuyruklar ve işlemler kapatılır. Ancak, bu kaynakları kullanmayı bitirdiğinizde, bu kaynakları açık bir şekilde kapatmak için iyi bir programlama uygulamasıdır. Kaynakları kapatmak için, her kaynakla ilişkilendirilmiş nesneyle ilgili Close yöntemini kullanın.

Kuyruk yöneticisiyle ilgili Commit ve Backout yöntemleri, yordamsal arabirimle birlikte kullanılan MQCMIT ve MQBACK çağrılarını değiştirir.

## Kuyruklara ve konulara erişilmesi

You can access queues and topics using methods of MQQueueManager or appropriate constructors.

Kuyruklara erişmek için, MQQueueManager sınıfının yöntemlerini kullanın. MQOD (nesne tanımlayıcı yapısı), bu yöntemlerin parametrelerine daraltılır. Örneğin, queueManageradlı bir MQQueueManager nesnesiyle gösterilen kuyruk yöneticisinde bir kuyruk açmak için aşağıdaki kodu kullanın:

```
MQQueue queue = queueManager.AccessQueue("qName",
                                           MQC.MQOO_OUTPUT,
                                           "qMgrName",
                                           "dynamicQName",
                                           "altUserId");
```

*seçenekler* parametresi, MQOPEN çağrısındaki Options parametresiyle aynıdır.

AccessQueue yöntemi, MQQueue sınıfını yeni bir nesne döndürür.

Kuyruğu kullanmayı bitirdiğinizde, aşağıdaki örnekteki gibi kapatmak için Close () yöntemini kullanın:

```
queue.Close();
```

IBM MQ .NET ile, MQQueue oluşturucusunu kullanarak bir kuyruk da yaratabilirsiniz. Parametreler, kullanılacak örnek MQQueueManager nesnesini belirten bir kuyruk yöneticisi değiştirgesinin eklenmesiyle, tam olarak accessQueue yöntemi ile 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 bir AccessTopic() yöntemini kullanın. Bu, sınıf MQTopic 'in yeni bir nesnesini döndürür. Konuyu kullanmayı bitirdiğinizde, bu konuyu kapatmak için MQTopic 'in Close () yöntemini kullanın.

Ayrıca, bir MQTopic oluşturucusu kullanarak bir konu da yaratabilirsiniz. Konular için bir dizi oluşturucular vardır; daha fazla bilgi için bkz. [MQTopic.NET sınıfı](#).

## İletilerin İşlenmesi

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 ya da konuları kuyruğa ya da konuya koyun. MQQueue ya da MQTopic sınıfının Get () yöntemini kullanarak kuyruklardan ya da konulardan ileti alın. Yordamsal arabirimden farklı olarak, burada MQPUT ve MQGET byte dizilerinin bayt dizilerinin olması, IBM MQ classes for .NET ' in MQMessage sınıfının somut örneklerini koymasına ve almalarına sağlar. MQMessage sınıfı, bu iletiyi açıklayan tüm MQMD (ileti tanımlayıcı) değiştirgeleriyle birlikte gerçek ileti verilerini içeren veri arabelleğinden sarkılanır.

Yeni bir ileti oluşturmak için, MQMessage sınıfının yeni bir eşgörünümünü yaratın ve ileti arabelleğindeki verileri yerleştirmek için WriteXXX yöntemlerini kullanın.

Yeni ileti eşgörünümü yaratıldığında, tüm MQMD parametreleri otomatik olarak varsayılan değerlerine ayarlanır ( [MQMD için ilk değerler ve dil bildirimleri](#) ' da tanımlandığı gibi). MQQueue yönteminin put () yöntemi, değiştirge olarak MQPutMessageSeçenekleri sınıfının bir örneğini de alır. Bu sınıf MQPMO yapısını temsil eder. Aşağıdaki örnek bir ileti yaratır ve bunu 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 olanağının get () yöntemi, kuyruktan yeni alınan iletiyi gösteren yeni bir MQMessage eşgörünümü döndürür. Ayrıca, parametre olarak MQGetMessageOptions sınıfının bir eşgörünümünü alır. Bu sınıf MQGMO yapısını temsil eder.

Get () yöntemi, gelen iletiye sığması için iç arabelleğindeki büyüklüğü otomatik olarak ayarlandığından, bir ileti büyüklüğü üst sınırı belirtmeniz gerekmez. Döndürülen iletteki verilere erişmek için MQMessage sınıfının ReadXXX yöntemlerini kullanın.

Aşağıdaki örnekte, kuyruktan iletinin nasıl alacağını gösterilmektedir:

```
// 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, *encoding* üye değişkenini ayarlayarak değiştirebilirsiniz.

*characterSet* adlı üye değişkenini ayarlayarak dizgileri okumak ve yazmak için kullanılacak karakter kümesini değiştirebilirsiniz.

Daha fazla ayrıntı için bkz. [MQMessage.NET sınıfı](#) .

**Not:** MQMessage WriteUTF() yöntemi, içerdiği Unicode baytların yanı sıra dizginin uzunluğunu da otomatik olarak kodlar. İletiniz başka bir .NET programı tarafından okunacağı zaman ( ReadUTF() kullanılarak), bu, dizgi bilgilerini göndermenin en basit yoludur.

### ***İleti özelliklerinin işlenmesi***

İleti özellikleri, iletileri seçmenize ya da üstbilgilerine erişmeden bir iletiyle ilgili bilgileri almanızı sağlar. MQMessage sınıfı, özellikleri almak ve ayarlamak için yöntemler içerir.

Bir uygulamanın işlenecek iletileri seçmesine izin vermek ya da MQMD ya da MQRFH2 üstbilgilerine erişmeden bir iletiyle ilgili bilgileri almak için ileti özelliklerini kullanabilirsiniz. Ayrıca, IBM MQ ve JMS uygulamaları arasında iletişimi de kolaylaştırır. IBM MQ'deki 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 yöntemlerinde Get \* Özelliği ve set yöntemlerinin adları şu biçim kümesi \* özelliğine sahiptir; 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 özelliği myproperty (bir karakter dizgisi) almak için, `message.GetStringProperty('myproperty')` çağrısını kullanın. İsteğe bağlı olarak bir özellik tanımlayıcısını iletilebileceğiniz gibi, IBM MQ tamamlanacaktır.

## Hataların işlenmesi

Handle errors arising from IBM MQ classes for .NET using try and catch blocks.

.NET arabirimindeki yöntemler bir tamamlanma kodu ve neden kodu döndürmez. Bunun yerine, bir IBM MQ çağrısından kaynaklanan tamamlanma kodu ve neden kodu her iki sıfır olmadığına bir kural dışı durum yayınlılar. This simplifies the program logic so that you do not have to check the return codes after each call to IBM MQ. Programınızın hangi noktalarda hata olma olasılığına karşı karar vereceğine karar verebilirsiniz. Bu noktalarda kodunuzu try ve catch blokları ile çevrebilirsiniz. Örneğin:

```
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 );
}
```

## Öznelik değerlerinin alınması ve ayarlanması

MQManagedObject, MQQueue ve MQQueueManager sınıfları, öznelik değerlerini almanıza ve ayarlamanıza izin veren yöntemler içerir. MQQueue için, yöntemlerin yalnızca, kuyruğu açtığınızda uygun sorgu ve küme işaretlerini belirttiğinizde işe yaradığını göz önünde bulundurun.

For common attributes, the MQQueueManager and MQQueue classes inherit from a class called MQManagedObject. Bu sınıf, Sorgula () ve Set () arabirimlerini tanımlar.

*Yeni* işlecini kullanarak yeni bir kuyruk yöneticisi nesnesi yarattığınızda, bu nesne otomatik olarak sorgulanmak üzere açılır. Bir kuyruk nesnesine erişmek için AccessQueue() yöntemini kullandığınızda, o nesne sorgulamak ya da ayarlama işlemleri için otomatik olarak açılmaz, bu durum bazı uzak kuyruklar tiplerinde sorunlara neden olabilir. Sorgulamak ve Ayarlama yöntemlerini kullanmak ve bir kuyrukta özellikleri ayarlamak için, AccessQueue() yönteminin openOptions değiştirilmesinde uygun olarak sorgu ve ayar işaretlerini belirlemeniz gerekir.

Sorgu 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 değiştirilmesine gerek yoktur. Aşağıdaki örnek, bir kuyruğun nasıl bir kuyruğun üzerinde nasıl yapılır gösterileceğini 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 öznelikleri sorgulanabilir. Özneliklerin bir alt kümesi, bir nesnenin özellikleri olarak gösterilir. Nesne özneliklerinin listesi için [Nesnelerin öznelikleribaşlıklı](#) konuya bakın. Nesne özellikleri için, uygun sınıf tanımına bakın.

## Çok iş parçacıklı programlar

.NET yürütme ortamı, doğal olarak çok iş parçacıklıdır. IBM MQ classes for .NET , bir kuyruk yöneticisi nesnesinin birden çok iş parçacığında paylaşılmasına izin verir, ancak hedef kuyruk yöneticisine tüm erişimin uyumlulaştırılmasını sağlar.

Bir kuyruk yöneticisine bağlanan ve başlatma sırasında kuyruk açan basit bir programı düşünün. Program ekranda tek bir düğme görüntüler. Bir kullanıcı bu düğmeyi tıklattığında, program kuyruktan bir ileti alır. Bu durumda, uygulamanın ilk kullanıma hazırlanması bir iş parçacığıda gerçekleşir ve düğme tuşuna yanıt olarak yürütülen kod, ayrı bir iş parçacığıda (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, ilgili bağlantı için devam etmekte olan diğer tüm çağrılar tamamlanincaya kadar engellenmiştir. Programınızdaki birden çok iş parçacığının aynı kuyruk yöneticisine eşzamanlı olarak erişmeniz gerekiyorsa, eşzamanlı erişim gerektiren her iş parçacığı için yeni bir MQQueueManager nesnesi yaratın. (Bu, her iş parçacığı için ayrı bir MQCONN çağrısı yayınlamaya eşdeğerdir.)

Varsayılan bağlantı seçenekleri MQC.MQCNO\_HANDLE\_SHARE\_NONE ya da MQC.MQCNO\_SHARE\_NO\_BLOCK , kuyruk yöneticisi artık uyumlulaştırılmadı.

## Using a client channel definition table with .NET

You can use a client channel definition table (CCDT) with the .NET classes for IBM MQ. CCDT ' nin yerini, yönetilen ya da yönetilmeyen bir bağlantı kullanıp kullanmamanıza bağlı olarak farklı şekillerde belirtiyorsunuz.

## XA dışı ya da XA yönetilmeyen istemci bağlantısı tipi

Yönetilmeyen bir bağlantı tipiyle CCDT ' nin yerini iki şekilde belirtebilirsiniz:

- Çizelgenin yer aldığı dizini belirtmek için MQCHLLIB ortam değişkenlerini ve çizelgenin dosya adını belirtmek için MQCHLTAB ' ı kullanın.
- İstemci yapılandırma dosyası kullanılıyor. CHANNELDEFINITION kısmında, çizelgenin bulunduğu dizini belirlemek için ChannelDefinitionDizinini ve dosya adını belirtmek için ChannelDefinitiondosyasını kullanın.

Konum hem istemci yapılandırma dosyasında, hem de ortam değişkenleri kullanılarak belirtilirse, ortam değişkenleri önceliğe sahip olur. Bu özelliği, istemci yapılandırma dosyasında standart bir yer belirtmek ve gerektiğinde ortam değişkenlerini kullanarak geçersiz kılmak için kullanabilirsiniz.

## Yönetilen istemci bağlantısı tipi

Yönetilen bir bağlantı tipiyle CCDT ' nin yerini üç şekilde belirtebilirsiniz:

- .NET uygulama yapılandırma dosyasını kullanma. CHANNELS kısmında, çizelgenin bulunduğu dizini belirlemek için ChannelDefinitionDizinini ve dosya adını belirtmek için ChannelDefinitiondosyasını kullanın.
- Çizelgenin yer aldığı dizini belirtmek için MQCHLLIB ortam değişkenlerini ve çizelgenin dosya adını belirtmek için MQCHLTAB ' ı kullanın.
- İstemci yapılandırma dosyası kullanılıyor. CHANNELDEFINITION kısmında, çizelgenin bulunduğu dizini belirlemek için ChannelDefinitionDizinini ve dosya adını belirtmek için ChannelDefinitiondosyasını kullanın.

Yer, bu şekilde birden fazla şekilde belirtilirse, ortam değişkenleri istemci yapılandırma dosyasına göre öncelik alır ve .NET Uygulama Yapılandırma Dosyası diğer iki yöntemden de öncelikli olarak önceliğe sahip olur. Bu özelliği, istemci yapılandırma dosyasında standart bir yer belirtmek ve gerektiğinde, ortam değişkenlerini ya da uygulama yapılandırma dosyasını kullanarak geçersiz kılmak için kullanabilirsiniz.

## Bir .NET uygulaması, hangi kanal tanımının kullanılacağını belirler

IBM MQ .NET istemci ortamında, kullanılacak kanal tanımlaması farklı şekillerde belirtilebilir. Kanal tanımlamasının birden çok belirtimi var olabilir. Bir uygulama, kanal tanımını bir ya da daha çok kaynaktan türetir.

Birden çok kanal tanımlaması varsa, kullanılan öğe aşağıdaki öncelik sırasına göre seçilir:

1. Properties specified on the MQQueueManager constructor, either explicitly or by including *MQC.CHANNEL\_PROPERTY* in the properties hashtable
2. Bir özellik *MQC.CHANNEL\_PROPERTY* , MQEnvironment.properties hashtable içinde
3. MQEnvironment 'daki Kanal özelliği
4. .NET uygulaması yapılandırma dosyası, bölüm adı KANALS, anahtar ServerConnectionParms (yalnızca yönetilen bağlantılar için geçerlidir)
5. *MQSERVER* ortam değişkeni
6. İstemci yapılandırma dosyası, stanza KANALLARI, Öznitelik ServerConnectionParms
7. İstemci kanal tanımlama çizelgesi (CCDT). CCDT ' nin yeri, .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 tanımlaması, uygulama tarafından sağlanan değerlerden alan tarafından alan temelinde oluşturulur. Bu değerler farklı arabirimler kullanılarak sağlanabilir ve her biri için birden çok değer bulunabilir. Alan değerleri, verilen öncelik sırasının ardından kanal tanımına eklenir:

1. MQQueueManager oluşturucuda *connName* değerinin değeri
2. MQQueueManager.properties HASH çizelgesinden özelliklerin değerleri
3. MQEnvironment.properties HASH çizelgesinden özellik değerleri
4. Değerler MQEnvironment alanları olarak ayarlandı (örneğin, MQEnvironment.Hostname, MQEnvironment.Port)

4-6 numaralı öğeler için, tüm kanal tanımlaması değer olarak sağlanır. Kanal tanımlamasındaki belirlenmemiş alanlar sistem varsayılanlarını alır. Diğer tanımlama yöntemlerinin ve alanlarının diğer yöntemlerinden herhangi bir değer bu belirtilerle birleştirilir.

7-9 numaralı öğeler için, tüm kanal tanımlaması CCDT ' den alınır. Kanal tanımlandığında belirtik olarak belirlenmeyen alanlar, sistem varsayılan değerlerini alır. Diğer tanımlama yöntemlerinin ve alanlarının diğer yöntemlerinden herhangi bir değer bu belirtilerle birleştirilir.

## Using channel exits in IBM MQ .NET

İstemci bağ tanımlarını kullanıyorsanız, diğer istemci bağlantılarında olduğu gibi kanal çıkışlarını da kullanabilirsiniz. Yönetilen bağ tanımlarını kullanırsanız, uygun bir arabirimi gerçekleştiren bir çıkış programı yazmanız gerekir.

### İstemci bağ tanımları

İstemci bağ tanımlarını kullanıyorsanız, kanal çıkışlarını Kanal çıkışlarında açıkladığı gibi kullanabilirsiniz. Yönetilen bağ tanımları için yazılan kanal çıkışlarını kullanamazsınız.

### Yönetilen bağ tanımları

Yönetilen bir bağlantı kullanırsanız, bir çıkışı gerçekleştirmek için uygun arabirimi gerçekleştiren yeni bir .NET sınıfı tanımlarsınız. IBM MQ paketinde üç çıkış arabirimi tanımlanır:

- MQSendExit
- MQReceiveExit



- MQSecurityExit

**Not:** Bu arabirimler kullanılarak yazılan kullanıcı çıkışları, yönetilmeyen ortamdaki kanal çıkışları olarak desteklenmez.

Aşağıdaki örnekte, üçünü gerçekleştiren bir sınıf tanımlanmaktadır:

```
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 nesne eşgörünümü geçirilir. Bu nesnelere, yordamsal arabirimde tanımlanan MQCXP ve MQCD yapılarını gösterir.

Bir gönderme çıkışı tarafından gönderilecek veriler ve bir güvenlik ya da alma çıkışta alınan veriler, çıkışa ilişkin parametreleri kullanarak belirtilir.

On entry, the data at offset *dataOffset* with length *dataLength* in the byte array *dataBuffer* is the data that is about to be sent by a send exit, and the data received in a security or receive exit. *dataMaxUzunluğu* parametresi uzunluk üst sınırını belirtir ( *dataOffset* 'tan) *dataBuffer*' taki çıkışa kullanılabilir. Not: Bir güvenlik çıkışı için, çıkışa ilk kez çağrılırsa ya da iş ortağı sonu veri göndermek üzere seçildiyse, *dataBuffer* için boş değer (null) olabilir.

Buna karşılık olarak, *dataOffset* ve *dataLength* değeri, .NET sınıflarının daha sonra kullanması gereken, döndürülen bayt dizisi içindeki görece konum ve uzunluk değerine işaret edecek şekilde ayarlanmalıdır. Gönderme çıkışı için bu, göndermesi gereken verileri gösterir ve bir güvenlik ya da alma çıkışı için, yorumlanacak veriler belirtilir. Çıkış, olağan durumda bir bayt dizisi döndürmelidir; kural dışı durumlar, veri göndermeyi seçebilecek bir güvenlik çıkışıdır ve INIT ya da TERM nedenleriyle çağrılan herhangi bir çıkıştan çıkılır. Bu nedenle yazılabilecek en basit çıkış biçimi *dataBuffer*' ı döndürmekten başka bir şey yapmamaktadır.

Olabilecek en basit çıkış gövdesi şunlardır:

```
{
    return dataBuffer;
}
```

## MQChannelDefinition sınıfı

Yönetilen .NET istemci uygulamasıyla belirtilen kullanıcı kimliği ve parola, istemci güvenlik çıkışa geçirilen IBM MQ .NET MQChannelDefinition sınıfında ayarlanır. Güvenlik çıkışı, kullanıcı kimliğini ve parolayı MQCD.RemoteUserIdentifier ve MQCD.RemotePassword alanları (bkz. [“Güvenlik çıkışı yazılıyor” sayfa 1005](#)).

### **Kanal çıkışlarının belirtilmesi (yönetilen istemci)**

MQQueueManager nesnesini yaratırken (MQEnvironment ya da MQQueueManager oluşturucuda) bir kanal adı ve bağlantı adı belirtirseniz, kanal çıkışlarını iki şekilde belirleyebilirsiniz.

Öncelik sırasıyla şunlar olur:

1. MQQueueManager oluşturucuda MQC.SECURITY\_EXIT\_PROPERTY, MQC.SEND\_EXIT\_PROPERTY ya da MQC.RECEIVE\_EXIT\_PROPERTY gruplama etiketleri (hashtable) özellikleri geçirilebilir.
2. MQEnvironment SecurityExit, SendExit ya da ReceiveExit özellikleri ayarlanıyor.

Bir kanal adı ve bağlantı adı belirtmezseniz, kanal tanımından kullanılacak kanal, istemci kanalı tanımlama çizelgesinden (CCDT) alınan kanal tanımlamalarından çıkar. Kanal tanımlamasında saklanan değerleri geçersiz kılmamız mümkün değildir. Kanal tanımlama çizelgelerine ilişkin ek bilgi için [İstemci kanal tanımlama çizelgesi](#) ve [“Using a client channel definition table with .NET” sayfa 559](#) başlıklı konuya bakın.

Her durumda, belirtim aşağıdaki biçimi kullanarak bir dizginin biçimini alır:

```
Assembly_name(Class_name)
```

*Sınıf\_adi* is the fully qualified name, including namespace specification, of a .NET class that implements the IBM.WMQ.MQSecurityExit, IBM.WMQ.MQSendExit or IBM.WMQ.MQReceiveExit interface (as appropriate). *Assembly\_name* , sınıfı barındıran düzeneğin dosya uzantısı da içinde olmak üzere tam olarak nitelenmiş konumdur. MQEnvironment ya da MQQueueManager özelliklerini kullanıyorsanız, dizilimin uzunluğu 999 karakterle sınırlıdır. Ancak, CCDT ' de kanal çıkış adı belirtildiyse, bu değer 128 karakterle sınırlıdır. Gerekliğinde, .NET istemci kodu dizgi belirtimini ayrıştırarak belirtilen sınıfın bir eşgörünümünü yükler ve yaratır.

### **Kanal çıkışı kullanıcı verilerinin belirtilmesi (yönetilen istemci)**

Kanal çıkışları, bunlarla ilişkilendirilmiş kullanıcı verilerine sahip olabilir. MQQueueManager nesnenizi yaratırken (MQEnvironment ya da MQQueueManager oluşturucuda) bir kanal adı ve bağlantı adı belirtirseniz, kullanıcı verilerini iki şekilde belirtebilirsiniz.

Öncelik sırasıyla şunlar olur:

1. MQQueueManager oluşturucuda MQC.SECURITY\_USERDATA\_PROPERTY, MQC.SEND\_USERDATA\_PROPERTY ya da MQC.RECEIVE\_USERDATA\_PROPERTY gruplama etiketleri (hashtable) özellikleri geçirilebilir.
2. MQEnvironment SecurityUserVerileri, SendUserVerileri ya da ReceiveUserVeri özellikleri ayarlanıyor.

Bir kanal adı ve bağlantı adı belirtmezseniz, kullanılacak çıkış kullanıcı veri değerleri, istemci kanal tanımlama çizelgesinden (CCDT) alınan kanal tanımlamasından gelir. Kanal tanımlamasında saklanan değerleri geçersiz kılmamız mümkün değildir. Kanal tanımlama çizelgelerine ilişkin ek bilgi için [İstemci kanal tanımlama çizelgesi](#) ve [“Using a client channel definition table with .NET” sayfa 559](#) başlıklı konuya bakın.

Her durumda, belirtim 32 karakterle sınırlı olan bir dizgidir.

## **.NET içinde otomatik istemci yeniden bağlantısı**

Beklenmeyen bir bağlantı sonu sırasında istemcinizin otomatik olarak bir kuyruk yöneticisine yeniden bağlanmasını sağlamanız gerekir.

Örneğin, kuyruk yöneticisi durdurursa ya da ağ ya da sunucu başarısız olursa, istemci beklenmedik bir şekilde kuyruk yöneticisinden bağlantısı kesilebilir.

Otomatik istemci yeniden bağlantısı olmadan, bağlantı başarısız olduğunda hata ortaya çıktı. Bağlantıyı yeniden kurmanıza yardımcı olması için hata kodunu kullanabilirsiniz.

Otomatik istemci yeniden bağlantı olanağını kullanan bir istemciye, yeniden bağlanabilir istemci adı verilir. Yeniden bağlanabilir bir istemci yaratmak için, kuyruk yöneticisine bağlanırken yeniden bağlanma seçenekleri çağrılan belirli seçenekleri belirtin.

İstemci uygulaması bir IBM MQ .NET istemciyse, kuyruk yöneticisi yaratmak için MQQueueManager sınıfını kullandığınızda, CONNECT\_OPTIONS\_PROPERTY için uygun bir değer belirterek, otomatik istemci yeniden bağlantı almayı tercih edebilir. CONNECT\_OPTIONS\_PROPERTY değerlerinin ayrıntıları için [Yeniden yapılandırma seçenekleri başlıklı konuya](#) bakın.

İstemci uygulamasının her zaman aynı adı taşıyan bir kuyruk yöneticisine, aynı kuyruk yöneticisine ya da istemci bağlantı çizelgesinde aynı QMNAME ile tanımlanmış kuyruk yöneticilerine bağlanıp bağlanmayacağını seçebilirsiniz (ayrıntılar için [CCDT 'de Kuyruk Yöneticisi Grupları 'a](#) 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 kurmasını sağlar.

### İlgili kavramlar

[IBM MQ.NET yönetilen istemci TLS desteği](#)

[Şifreleme güvenlik iletişim kuralları: TLS](#)

### Yönetilmeyen .NET istemcisi için TLS desteği

Yönetilmeyen .NET istemcisi için TLS desteği, C MQI ve GSKit 'i temel alır. TLS işlemlerini ve GSKit 'i işleyen C MQI, TLS güvenli yuva iletişim kurallarını uygular.

#### Yönetilmeyen .NET istemcisi için TLS 'nin etkinleştirilmesi

TLS yalnızca istemci bağlantıları için desteklenir. TLS 'yi etkinleştirmek için, kuyruk yöneticisiyle iletişim kurarken kullanılacak CipherSpec değerini belirtmelisiniz; bu, hedef kanaldaki CipherSpec ayarına uygun olmalıdır.

TLS 'yi etkinleştirmek için, MQEnvironment 'ın SSLCipherSpec durağan üye değişkenini kullanarak CipherSpec değerini belirtin. Aşağıdaki örnek, TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA:' un CipherSpec ile TLS gerektirecek şekilde ayarlanmış bir SVRCONN kanalına SECURE.SVRCONN.CHANNELadlı bir kanala bağlanır.

```
MQEnvironment.Hostname           = "your_hostname";
MQEnvironment.Channel            = "SECURE.SVRCONN.CHANNEL";
MQEnvironment.SSLCipherSpec     = "TLS_RSA_WITH_AES_128_CBC_SHA256";
MQEnvironment.SSLKeyRepository = "C:\mqm\key";
MQQueueManager qmgr = new MQQueueManager("your_q_manager");
```

CipherSpecslistesi için bkz. [Specifying CipherSpecs](#) .

The SSLCipherSpec property can also be set using the MQC.SSL\_CIPHER\_SPEC\_PROPERTY in the hash table of connection properties.

TLS 'yi kullanarak başarıyla bağlanmak için, istemci anahtar deposunun, kuyruk yöneticisi tarafından sunulan sertifikanın doğrulanabileceği Sertifika Yetkilisi kök sertifikaları zinciriyle birlikte ayarlanması gerekir. Benzer şekilde, SVRCONN kanalında SSLClientAuth MQSSL\_CLIENT\_AUTH\_REQUIRED olarak ayarlandıysa, istemci anahtar deposu, kuyruk yöneticisi tarafından güvenilen bir tanıtıcı özel sertifika içermeli.

#### Kuyruk yöneticisinin Ayırt Edici Adı 'nı kullanma

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 örneği belirtildi. Örneğin:

```
MQEnvironment.SSLPeerName = "CN=QMGR.*, OU=IBM, OU=WEBSPPHERE";
```

Ancak kuyruk yöneticisi, QMGR. ' un başında bir Common Name değeri olan bir sertifika sunarsa, bağlantının başarılı olmasına izin verir. ve en az iki Kuruluş Birimi adı, ilkinin IBM ve ikinci WEBSPPHERE olması gerekir.

The SSLPeerName property can also be set using the MQC.SSL\_PEER\_NAME\_PROPERTY in the hash table of connection properties. Eş adları ayarlamaya ilişkin ayırt edici adlar ve kurallara ilişkin ek bilgi edinmek için [Securing IBM MQ](#) belgesine bakın.

SSLPeerName ayarlandıysa, bağlantılar yalnızca geçerli bir örüntü olarak ayarlandıysa ve kuyruk yöneticisi eşleşen bir sertifika sunarsa başarılı olur.

#### *TLS kullanılırken hata işlerken hata oluştu*

TLS kullanan bir kuyruk yöneticisine bağlanırken IBM MQ classes for .NET tarafından aşağıdaki neden kodları yayınlanabilir:

#### **MQRC\_SSL\_NOT\_ALLOWED**

SSLCipherSpec özelliği ayarlıydı, 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**

SSLPeerName özelliğinde belirtilen ayırt edici ad (DN) kalıbı geçerli değil.

### ***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 en düşük .NET Framework düzeyi .NET Framework v3.5' dir. Şifre Algoritması desteğinin düzeyi, uygulamanın kullanmakta olduğu .NET Framework düzeyini temel alır:

- .NET Framework düzeylerine 3.5 ve 4.0dayalı uygulamalar için, kullanılabilir güvenli yuva protokolleri SSL 3.0 ve TSL 1.0' dır.
- For applications that are based on .NET Framework level 4.5, the available secure socket protocols are SSL 3.0, TLS 1.1 and TLS 1.2.

You might need to move applications that expect higher TLS protocol support to a later version of the framework as defined for Microsoft Security support in the .NET Framework.

Yönetilen .NET istemcisi için TLS desteğinin başlıca özellikleri aşağıdaki gibidir:

#### **TLS iletişim kuralı desteği**

.NET yönetilen istemcisine ilişkin TLS desteği, .NET SSLStream sınıfı aracılığıyla tanımlanır ve uygulamanın kullanmakta olduğu .NET Framework 'e bağlıdır. Daha fazla bilgi için bkz. [“Yönetilen .NET istemcisine ilişkin TLS iletişim kuralı desteği” sayfa 565.](#)

#### **CipherSpec desteği**

.NET yönetilen istemcisine ilişkin TLS ayarları, Microsoft.NET TLS steam 'larına uygun olarak bulunur. Daha fazla bilgi için bkz. [“Yönetilen .NET istemcisine ilişkinCipherSpec desteği” sayfa 565](#) ve [“Yönetilen .NET istemcisine ilişkinCipherSpec eşlemeleri” sayfa 567.](#)

#### **Anahtar havuzları**

İstemci tarafındaki anahtar havuzu bir Windows anahtar deposudur. Sunucu tarafı havuzu, havuzun bir Cryptographic Message Syntax (CMS) tipidir. Daha fazla bilgi için bkz. [“Yönetilen .NET istemcisine ilişkin anahtar havuzları” sayfa 568.](#)

## Sertifikalar

Bir istemci ile kuyruk yöneticisi arasında karşılıklı kimlik doğrulamayı uygulamak için kendinden imzalı TLS sertifikalarını kullanabilirsiniz. Daha fazla bilgi için bkz. [“Yönetilen .NET istemcisi için sertifikaları kullanma” sayfa 569.](#)

## SLAYICI ADI

.NET' ta, uygulamalar isteğe bağlı SSLPEERNAME özniteliğini kullanarak bir Ayırt Edici Ad (DN) örüntülerini belirtir. Daha fazla bilgi için bkz. [“SLAYICI ADI” sayfa 569.](#)

## FIPS uyumluluğu

Enabling FIPS programmatically is not supported by the Microsoft.NET Security library. FIPS etkinleştirilmesi, Windows Grup İlkesi ayarı tarafından denetlenir.

## NSA Suite B uyumluluğu

IBM MQ , RFC 6460 'ı gerçekleştirir. NSA Suite B için Microsoft.NET uygulaması 5430 'tür. Bu, .NET Framework 3.5 ' den itibaren desteklenir.

## Gizli anahtar ilk duruma getirme ya da yeniden görüşme

SSLStream sınıfı, diğer IBM MQ istemcileriyle tutarlılık sağlamak amacıyla, gizli anahtar ilk duruma getirme ya da yeniden görüşme özelliğini desteklemese de, .NET yönetilen istemcisi uygulamaların SSLKeyResetCount (Sayı) olarak ayarlanmasına izin verir. Daha fazla bilgi için bkz. [“Yönetilen .NET istemcisi için gizli anahtar sıfırlama ya da yeniden anlaşma” sayfa 570.](#)

## İptal denetimi

SSLStream sınıfı, sertifika zincirleme motoru tarafından otomatik olarak yapılan sertifika iptal denetimini destekler. Daha fazla bilgi için bkz. [“İptal denetimi” sayfa 570.](#)

## IBM MQ güvenlik çıkışı desteği

SSLStream sınıfı, IBM MQ güvenlik çıkışları için sınırlı destek sağlar. Querying local and remote certificates to get SSLPeerNamePtr(Subject DN) and SSLRemCertIssNamePtr (Issuer DN) is possible since this is supported in Microsoft.NET. Ancak, DNQ, UNSTRUCTUREDNAME ve UNSTRUCTUREADDRESS gibi öznitelikleri alma desteği yoktur, bu nedenle bu değerler çıkışlar kullanılarak alınmaz.

## Şifreleme donanımı desteği

Yönetilen .NET istemcisi için şifreleme donanımı desteklenmez.

## Yönetilen .NET istemcisine ilişkin TLS iletişim kuralı desteği

IBM MQ.NET TLS desteği, .NET SSLStream sınıfını temel alır.

**Not:** Yönetilen .NET istemcisine ilişkin TLS iletişim kuralı desteği, uygulamanın kullanmakta olduğu .NET Framework düzeyine bağlıdır. Daha fazla bilgi için [“Yönetilen .NET istemcisi için TLS desteği” sayfa 564](#) başlıklı konuya bakın.

Microsoft.NET SSLStream sınıfı için TLS ' yi kullanıma hazırlamak ve kuyruk yöneticisiyle el sıkışmasını gerçekleştirmek için ayarlamamız gereken parametrelerden biri **SSLProtocol** olur; bu durumda TLS sürüm numarasını belirtmeniz gerekir; bu durumda aşağıdaki değerlerden biri olmalıdır:

- SSL3.0
- TLS1.0
- TLS1.2

Bu parametrenin değeri, tercih edilen CipherSpec ' in ait olduğu Protokol ailesiyle sıkı bir şekilde birleşir. SSLStream sunucusuyla (kuyruk yöneticisi) TLS tokalaşması başladığında, anlaşma için kullanılacak CipherSpecs listesini tanımlamak için **SSLProtocol** içinde belirtilen TLS sürümünü kullanır.

IBM MQ.NET , uygulamaların bu değeri ayarlamak için kullanabileceği özellikleri yapmaz. Bunun yerine, IBM MQ , İletişim Kuralı ailesine ayarlanan CipherSpec ' i dahili olarak eşlemek için bir eşleme tablosu kullanır ve kullanılacak SSLProtocol sürümünü tanımlar. Bu tablo, Microsoft.NET ile IBM MQarasındaki desteklenen CipherSpec ' in her birinin ve ait oldukları Protokol sürümünün eşlemesini gösterir. Daha fazla bilgi için [“Yönetilen .NET istemcisine ilişkinCipherSpec eşlemeleri” sayfa 567](#) başlıklı konuya bakın.

## Yönetilen .NET istemcisine ilişkinCipherSpec desteği

Bir uygulamaya ilişkin CipherSpec ayarları, sunucusuyla el sıkışma sırasında kullanılır.

IBM MQ istemcileri, kuyruk yöneticisiyle el sıkışma sırasında kullanılan bir CipherSpec değerini ayarlamanıza olanak tanır. IBM MQ istemcileri, güvenli bağlantı oluşturmak üzere güvenli bağlantı için geçerli bir CipherSpec ayarlamalıdır, tercihen Windows grup ilkesinde belirtilen CipherSpec . Bu alanı boş bırakmak, yuvalar üzerinde herhangi bir güvenlik olmadan düz metin kanalını gösterir.

IBM MQ.NET yönetilen istemcisi için, TLS ayarları Microsoft.NET SSLStream sınıfı içindir. For SSLStream, a CipherSpec, or a preference list of CipherSpecs, can be set only in the Windows group policy, which is a computer-wide setting. Daha sonra SSLStream, sunucuyla el sıkışma sırasında belirtilen CipherSpec ya da tercih listesini kullanır. In case of other IBM MQ clients, the CipherSpec property can be set in the application on the IBM MQ channel definition and the same setting is used for TLS negotiation. Bu kısıtlama sonucunda, TLS anlaşması, IBM MQ kanal yapılandırmasında belirtilenler dikkate alınmaksızın, desteklenen herhangi bir CipherSpec ' i kararlaştırılabilir. Bu nedenle, bu, kuyruk yöneticisinde AMQ9631 hatasının sonuçlanacaktır olması olasıdır. To avoid this error, set the same CipherSpec as the one that you have set in the application as the TLS configuration in the Windows group policy.

Yeni IBM MQ.NET TLS istemci kodu, yalnızca doğru protokol sürümünün kararlaştırıldığını denetler. TLS iletişim kuralı sürümü, uygulama kümelerinin ve sunucu (kuyruk yöneticisi) ile TLS anlaşması için kullanılan CipherSpec ' den türetilir. Bu nedenle, tasarımın IBM MQ.NET yönetilen istemci uygulamasında CipherSpec ' i ayarlamak için gereklidir. IBM MQ istemcisi tarafından ayarlanan CipherSpec , SSL 3.0, TLS 1.0 ve TLS 1.2 iletişim kurallarından başka bir şey değilse, IBM MQ tarafından yönetilen .NET istemcisi varsayılan olarak SSL 3.0 ya da TLS 1.0 iletişim kurallarındaki şifrelemelerden herhangi biriyle ilişki kurar ve bu, bir hatayı bildirmez.

**Not:** Uygulama tarafından sağlanan CipherSpec değeri, IBM MQ ile bilinen bir CipherSpec değilse, IBM MQ tarafından yönetilen .NET istemcisi bu değeri göz atmaz ve Windows sisteminin grup ilkesine dayalı olarak bağlantıyı kararlaştıramaz.

## CipherSpec ayarlanması

Bir CipherSpec ayarının üç yolu vardır:

### MQEnvironment .NET sınıfı

Aşağıdaki örnek, MQEnvironment sınıfından 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 ayarlanacak olduğunu 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 ilkesinde bir CipherSpec ayarlandığında, SVRCONN kanalındaki ve uygulamada SSLCipherSpec özellik değeri için aynı CipherSpec ayarlanmalıdır. If the Windows group policy is set to the default, that is the group policy is not enabled/edited for CipherSpec setting, applications must set the same default value of the CipherSpec from the Windows group policy TLS configuration in the MQEnvironment class or in the MQQueueManager constructor hashtable properties.

## CCDT kullanımı

IBM MQ.NET yalnızca yerel bir bilgisayarda bulunan İstemci Kanal Tanımlama Çizelgelerini (.TAB dosyaları) destekler. Existing CCDT files that have a CipherSpec value set can be used for IBM MQ.NET connections. Ancak, istemci bağlantı kanalında ayarlanan CipherSpec değeri TLS iletişim kuralı sürümünü belirler ve ayrıca Windows grup ilkesinde ayarlanan CipherSpec ile eşleşmelidir.

### İlgili kavramlar

[“IBM MQ ortamını ayarlama” sayfa 554](#)

Bir kuyruk yöneticisine bağlanmak için istemci bağlantısını kullanmadan önce, IBM MQ ortamını ayarlamanız gerekir.

### İlgili görevler

[CipherSpecsbelirtme](#)

### İlgili başvurular

[MQEnvironment .NET sınıfı](#)

*Yönetilen .NET istemcisine ilişkinCipherSpec eşlemeleri*

IBM MQ.NET arabirimi, bir kuyruk yöneticisiyle güvenli bağlantı kurmak için yönetilen istemcinin 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, istemci uygulamasının kullandığı anlaşmalı CipherSpec ile CipherSpec ile eşleşmeyi dener. Kuyruk yöneticisi eşleşen bir CipherSpecbulamazsa, iletişim hatası AMQ9631ile başarısız olur.

IBM MQ.NET arabirimi, IBM MQ ' den Microsoft.NET CipherSpec eşleme çizelgesine sahiptir. Bu çizelge, istemci kuyruk yöneticisiyle güvenli yuva bağlantısı kurmak için istemcinin kullanmak istediği TLS iletişim kuralı sürümünü belirlemek için kullanılır. Based on the SSLCipherSpec value, the SSLProtocol version can be TLS 1.0, or TLS 1.2, depending on which version of the Microsoft.NET Framework you are using.

Make sure that you provide the correct the SSLCipherSpec value as specifying an incorrect value might result in SSL 3.0 or TLS 1.0 protocols being used.

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 <sup>1</sup>	TLS_RSA_WITH_3DES_EDE_CBC_SHA <sup>1</sup>	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
<b>V 9.1.0.6</b> ECDHE_RSA_AES_128_GCM_SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2
<b>V 9.1.0.6</b> ECDHE_RSA_AES_256_GCM_SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2

Çizelge 76. IBM MQ ve Microsoft.NET eşleme tablosu (devamı var)

IBM MQ CipherSpec	Microsoft.NET CipherSpec	TLS sürümü
ECDHE_RSA_AES_128_CBC_SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P384	TLS 1.2
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384	TLS 1.2
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P521	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P384	TLS 1.2
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P521	TLS 1.2
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384	TLS 1.2
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P521	TLS 1.2

#### Notlar:

1. Bu CipherSpec TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA kullanımdan kaldırılmıştır. However, it can still be used to transfer up to 32 GB of data before the connection is terminated with error AMQ9288. Bu hatayı önlemek için, üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpec komutunu kullanırken gizli anahtar sıfırlamayı etkinleştirmeniz gerekir.

#### Yönetilen .NET istemcisine ilişkin anahtar havuzları

Yönetilen .NET istemcileri tarafından kullanılan anahtar havuzu, Windows anahtar deposudur. Sertifikalar ve özel anahtarlar, bir TLS anlaşması sırasında hem kimlik hem de güven için istemci uygulaması tarafından kullanılabilir için kullanıcı ya da sistem anahtar deposunda kullanılabilir olmalıdır.

### İstemci tarafı

The client side of the TLS configuration in IBM MQ.NET consists of client side key repository, client certificates, and the choices the application program makes.

- İstemci tarafı anahtar havuzu her zaman bir Windows anahtar deposudur. Bu, sertifikaların depolanabileceği bir Kullanıcı ya da Bilgisayar hesabı olabilir.
- Uygulamada, anahtar havuzu için aşağıdaki değerlerden herhangi birini ayarlayabilirsiniz:
  - "\*USER": IBM MQ.NET, istemci sertifikalarını almak için geçerli kullanıcının sertifika deposuna erişir.
  - "\*SYSTEM": IBM MQ.NET accesses the local Computer account to retrieve the certificates.
- İstemcinin sertifikaları, Kullanıcı ya da Bilgisayar hesabının sertifikam depoda depolanmalıdır. Tüm sunucu (CA) sertifikaları, sertifika deposunun kök dizininde saklanmalıdır.



**Not:** Tek bir dosyada birden çok sertifikayı aşağıdaki biçimlerde saklayabilirsiniz:

- Kişisel Bilgiler Exchange-PKCS #12 (.PFX, .P12)
- Şifreleme İletisi Sözdizimi Standardı-PKCS #7 Sertifikaları (.P7B)
- Microsoft Serileştirilmiş Sertifika Deposu (.SST)

*Yönetilen .NET istemcisi için sertifikaları kullanma*

İstemci sertifikaları için, IBM MQ tarafından yönetilen .NET istemcisi Windows anahtar deposuna erişir ve sertifika etiketiyle eşleşen ya da dizgiyle eşleştirilen tüm istemci sertifikalarını yükler.

Kullanmak üzere bir sertifika seçerken, IBM MQ tarafından yönetilen .NET istemcisi her zaman SSLStream TLS anlaşması için ilk eşleşen sertifikayı kullanır.

## Sertifika etiketine göre eşleşen sertifikalar

Sertifika etiketini ayarladıysanız, IBM MQ tarafından yönetilen .NET istemcisi, istemci sertifikasını tanımlamak için belirtilen etiket adıyla Windows sertifika deposunda arama yapar. Eşleşen tüm sertifikaları yükler ve listede ilk sertifikayı kullanır. Sertifika etiketini ayarlamak için iki seçenek vardır:

- Sertifika etiketi, MQEnvironment.CertificateLabel adlı MQEnvironment sınıfına erişerek ayarlanabilir.
- The certificate label can also be set in a hash table properties, supplied as input parameter with MQQueueManager constructor as shown in the following example.

```
Hashtable properties = new Hashtable();
properties.Add("CertificateLabel", "mycert");
```

Ad ("CertificateLabel") ve değer büyük ve küçük harfe duyarlıdır.

## Sertifikalara dizgi temelinde eşleştirme

Sertifika etiketi ayarlanmadıysa, "ibmwebspheremq" dizgisiyle eşleşen sertifika ve yürürlükteki oturum açmış olan kullanıcı (küçük harfli) aranır ve kullanılır.

### İlgili görevler

[İstemcinin kuyruk yöneticisine güvenli bir şekilde bağlanması](#)

### İlgili başvurular

[MQEnvironment .NET sınıfı](#)

*SLAYICI ADI*

SSLPEERNAME özniteliği, eşdüzey kuyruk yöneticisinden alınan sertifikana ilişkin ayırt edici adı (DN) denetlemek için kullanılır.

IBM MQ.NET' ta uygulamalar, aşağıdaki örnekte gösterildiği gibi bir ayırt edici ad kalıbı belirlemek için SSLPEERNAME değerini kullanabilir.

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPPHERE)
```

Diğer IBM MQ istemcilerine gelince, SSLPEERNAME isteğe bağlı bir parametredir.

SSLPEERNAME değeri ayarlanmazsa, IBM MQ.NET tarafından yönetilen istemci herhangi bir Uzak (Sunucu) sertifika geçerlilik denetimi yapmaz ve yönetilen istemci, yalnızca Uzak (/sunucu) sertifikasını kabul eder.

SSLPEERNAME ayarladığınız yol, kullanmakta olduğunuz IBM MQ yığın olanaklarından hangisine bağlıdır.

### IBM MQ classes for .NET

Aşağıdaki gibi üç seçenek vardır.

1. MQEnvironment sınıfındaki MQEnvironment.SSLPeerName ögesini ayarlayın.
2. MQEnvironment.properties.Add(MQC.SSL\_PEER\_NAME\_PROPERTY, *value*)

- Use the queue manager constructor `MQQueueManager (String queueManagerName, Hashtable properties)`. Supply the `SSLPEERNAME` in the `Hashtable properties` as for option 2.

### **XMS .NET**

Bağlantı üreticisinde SSL eş adını ayarlayın:

```
ConnectionFactory.SetStringProperty(XMSC.WMQ_SSL_PEER_NAME, value);
```

### **WCF**

Include `SslPeerName` as a semicolon separated field in the URI.

### **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 istemci, uygulamaların `SSLKeyResetSayımı` 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 bağlantısı](#) başlıklı konuya bakın.

### **İlgili kavramlar**

[SSL ve TLS gizli anahtarlarını sıfırlama](#)

*İptal denetimi*

`SSLStream` sınıfı, sertifika iptal denetimini destekler.

İptal denetimi, sertifika zincirleme motoru tarafından otomatik olarak yapılır. Bu, hem Online Certificate Status Protocol (OCSP) hem de Sertifika İptal listeleri (CRL 'ler) için geçerlidir. `SSLStream` sınıfı, yalnızca sertifikada belirtilen sunucuyu kullanan sertifika iptal özelliğini kullanır; bu, sunucunun kendisi tarafından dikte edilir. HTTP CDP uzantıları ve OCSP HTTP istekleri, HTTP yetkili sunucusu aracılığıyla yetkili sunucuya yönelik olarak istenmektedir.

İptal denetimini ayarladığınız yol, kullanmakta olduğunuz IBM MQ yığın olanaklarından hangisine bağlıdır.

### **IBM MQ.NET**

İptal denetimi, `MQEnvironment.cs` sınıf dosyasındaki

**`MQEnvironment.SSLCertRevocationCheck`** özelliğine erişilerek ayarlanabilir.

### **XMS .NET**

İptal denetimi, aşağıdaki örnekte gösterildiği gibi, bağlantı üreticisi özellik bağlamında ayarlanabilir.

```
ConnectionFactory.SetBooleanProperty(XMSC.WMQ_SSL_CERT_REVOCATION_CHECK, true);
```

### **WCF**

İptal denetimi, aşağıdaki adlandırma kuralını kullanarak URI 'de ayarlanabilir.

```
"SslCertRevocationCheck=true"
```

*Yönetilen IBM MQ .NET için TLS 'nin yapılandırılması*

Yönetilen IBM MQ .NET için TLS 'nin yapılandırılması, imzalayıcı sertifikalarının oluşturulmasından, sonra sunucu tarafı, istemci tarafı ve uygulama programı yapılandırılmasından oluşur.

## Bu görev hakkında

TLS ' yi yapılandırmak için önce uygun imzalayıcı sertifikalarını oluşturmanız gerekir. İmzalayıcı sertifikaları, bir sertifika yetkilisi tarafından sağlanan otomatik olarak imzalanmış ya da sertifikalar olabilir. Kendi kendine imzalanmış sertifikalar geliştirme, test veya üretim öncesi sistem üzerinde kullanılabilir de, bunları bir üretim sisteminde kullanmayın. Bir üretim sisteminde, güvenilir bir dış sertifika yetkilisinden (CA) edindiğiniz sertifikaları kullanın.

## Yordam

### 1. İmzalayıcı sertifikalarını oluşturun.

- a) Kendinden onaylı sertifikalar oluşturmak için, IBM MQ ile birlikte verilen aşağıdaki araçlardan birini kullanın:

Komut satırında **strmqickm** GUI ya da **runmqckm** ya da **runmqakm** komutunu kullanın. Bu araçların kullanılmasına ilişkin ek bilgi için [Dijital sertifikaları yönetmek için runmqckm, runmqakm ve strmqickm olanağının kullanılmasında başlıklı konuya](#) bakın.

- b) Bir sertifika yetkilisinden (CA) kuyruk yöneticisine ve istemcilere ilişkin sertifikaları almak için, [Sertifika yetkilisinden kişisel sertifikaların edinilmesinde başlıklı konu yönergeleriyle ilgili yönergeleri](#) izleyin.

### 2. Sunucu tarafını yapılandırın.

- a) Configure TLS on the queue manager, using GSKit, as described in [İstemcinin kuyruk yöneticisine güvenli bir şekilde bağlanması](#).
- b) SVRCONN kanalı TLS özniteliklerini ayarlayın:
  - **SSLCAUTH** ögesini "REQUIRED/OPTIONAL" olarak ayarlayın.
  - Set **SSLCIPH** to an appropriate CipherSpec.

Daha fazla bilgi için bkz. "Yönetilmeyen .NET istemcisi için TLS ' nin etkinleştirilmesi" sayfa 563.

### 3. İstemci tarafını yapılandırın.

- a) İstemci sertifikalarını Windows sertifika deposuna (Kullanıcı/Bilgisayar hesabı altında) aktarın. IBM MQ .NET accesses client certificates from the Windows certificate store, therefore you must import your certificates into the Windows certificate store to establish a secure socket connection to IBM MQ . Windows anahtar deposuna erişme ve istemci tarafı sertifikalarını içe aktarma hakkında daha fazla bilgi için [Sertifikaların ve özel anahtarların içe aktarılması ya da dışa aktarılması başlıklı konuya](#) bakın.
- b) Supply the CertificateLabel as described in [İstemcinin kuyruk yöneticisine güvenli bir şekilde bağlanması](#).
- c) If needed, edit the Windows Group Policy to set the CipherSpec, then, for the Windows Group Policy updates to take effect, restart the computer.

### 4. Uygulama programını yapılandırın.

- a) MQEnvironment ya da SSLCipherSpec değerini, bağlantıyı güvenli bir bağlantı olarak ifade edecek şekilde ayarlayın.

Kullanılmakta olan protokolü tanımlamak için belirttiğiniz değer kullanılır (TLS). CipherSpec kümesi, desteklenen SSLProtocol sürümünün CipherSpecs öğelerinden biri olmalıdır ve bu, tercihen Windows Group Policy ile aynı olabilir. (Desteklenen SSLProtocol sürümü, kullanılan .NET çerçevesinin üzerine bağlıdır. SSLProtocol sürümü, kullanmakta olduğunuz Microsoft .NET Framework sürümüne bağlı olarak TLS 1.0 ya da TLS 1.2 olabilir.)

**Not:** Uygulama tarafından sağlanan CipherSpec değeri, IBM MQ ile bilinen bir CipherSpec değilse, IBM MQ tarafından yönetilen .NET istemcisi bu değeri göz atmaz ve Windows sisteminin grup ilkesine dayalı olarak bağlantıyı kararlaştıramaz.

- b) SSLKeyRepository özelliğini "\*SYSTEM" ya da "\*USER" olarak ayarlayın.
- c) İsteğe bağlı: SSLPEERNAME değerini sunucu sertifikasının ayırt edici adına (DN) ayarlayın.

- d) Supply the CertificateLabel as described in [İstemcinin kuyruk yöneticisine güvenli bir şekilde bağlanması](#).
- e) Set any further optional parameters that you require such as KeyResetCount, CertificationRevocationCheck, and enable FIPS.

## TLS iletişim kuralının ve TLS anahtar havuzunun nasıl ayarlanmalarına ilişkin örnekler

Temel .NET için, TLS iletişim kuralını ve TLS anahtar havuzunu MQEnvironment sınıfı aracılığıyla aşağıdaki örnekte gösterildiği gibi 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, TLS iletişim kuralını ve TLS anahtar havuzunu MQQueueManager oluşturucusunun bir parçası olarak bir hashtable sağlayarak 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

For more information about getting started with developing IBM MQ .NET managed TLS applications, see [“Basit bir uygulama yazma” sayfa 572](#).

## İlgili başvurular

[MQEnvironment .NET sınıfı](#)

[KeyResetCount \(MQLONG\)](#)

[Federal Information Processing Standards \(FIPS\) for UNIX, Linux, and Windows](#)

### *Basit bir uygulama yazma*

Basit bir IBM MQ yönetilen .NET TLS uygulaması yazmak için ipuçları, bağlantı fabrikalarına ilişkin SSL özelliklerini ayarlama, kuyruk yöneticisi örneği oluşturma, bağlantı, oturum ve hedef yaratma ve sınamaya ilişkin gönderme örnekleri de içinde olmak üzere.

## Başlamadan önce

Öncelikle yönetilen IBM MQ .NET için TLS 'yi [“Yönetilen IBM MQ .NET için TLS 'nin yapılandırılması” sayfa 570](#) 'de açıklandığı şekilde yapılandırmanız gerekir.

.NET tabanındaki uygulama programı yapılandırma için, SSL özelliklerini MQEnvironment sınıfını kullanarak ya da MQQueueManager oluşturucusunun bir parçası olarak bir hashtable belirterek ayarlayın.

XMS .NET' de uygulama programı yapılandırması için, SSL özelliklerini bağlantı üreticilerinin özellik bağlamına ayarladınız.

## Yordam

1. Aşağıdaki örneklerde gösterildiği gibi, bağlantı fabrikalarına ilişkin SSL özelliklerini ayarlayın.

### **IBM MQ.NET Örneği**

```
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 için örnek:.NET

```
cf.SetStringProperty(XMSC.WMQ_SSL_KEY_REPOSITORY, "sslKeyRepository");
cf.SetStringProperty(XMSC.WMQ_SSL_CIPHER_SPEC, cipherSpec);
cf.SetStringProperty(XMSC.WMQ_SSL_PEER_NAME, sslPeerName);
cf.SetIntProperty(XMSC.WMQ_SSL_KEY_RESETCOUNT, keyResetCount);
cf.SetBooleanProperty(XMSC.WMQ_SSL_CERT_REVOCATION_CHECK, true);
```

2. Aşağıdaki örneklerde gösterildiği gibi, kuyruk yöneticisi yönetim ortamını, bağlantıları, oturumu ve hedefi yaratın.

### MQ .NETÖrneği

```
queueManager = new MQQueueManager(queueManagerName, properties);
Console.WriteLine("done");

// accessing queue
Console.WriteLine("Accessing queue " + queueName + "..");
queue = queueManager.AccessQueue(queueName, MQC.MQOO_OUTPUT +
MQC.MQOO_FAIL_IF_QUIESCING);
Console.WriteLine("done");
```

### XMS için örnek:.NET

```
connectionWMQ = cf.CreateConnection();
// Create session
sessionWMQ = connectionWMQ.CreateSession(false, AcknowledgeMode.AutoAcknowledge);

// Create destination
destination = sessionWMQ.CreateQueue(destinationName);

// Create producer
producer = sessionWMQ.CreateProducer(destination);
```

3. Aşağıdaki örneklerde gösterildiği gibi bir ileti gönderin.

### MQ .NETÖrneği

```
// creating a message object
message = new MQMessage();
message.WriteString(messageString);

// putting messages continuously
for (int i = 1; i <= numberOfMsgs; i++)
{
Console.WriteLine("Message " + i + " <" + messageString + ">.. ");
queue.Put(message);
Console.WriteLine("put");
}
```

### XMS için örnek:.NET

```
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 sınıfı ile ilgili izleme olaylarını ve iletileri yakalamak için, uygulamanıza ilişkin uygulama yapılandırma dosyasına sistem tanılması için bir yapılandırma bölümü eklemelisiniz.

## Bu görev hakkında

Uygulama yapılandırma dosyasına sistem tanılması için bir yapılandırma bölümü eklemeyeniz, IBM MQ tarafından yönetilen .NET istemcisi, TLS ve SSLStream sınıflarıyla ilgili herhangi bir olay, izleme ya da hata ayıklama noktası yakalamaz.

**Not:** Starting IBM MQ tracing using `strmqtrc` does not capture all the required TLS tracing.

## Yordam

1. Uygulama projeniz için bir uygulama yapılandırması (App.Config) dosyası oluşturun.
2. Aşağıdaki örnekteki 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 girdisinin Version alanının, kullanılmakta olan .net amqmdnet.dll dosyasının hangi sürümü olması gerekir.

### Yönetilen .NET' ta TLS uygulamasına ilişkin örnek uygulamalar

Sample applications are provided to show the implementation of TLS for managed .NET in IBM MQ classes for .NET, XMS .NET and IBM MQ custom channel for WCF.

Aşağıdaki tabloda örnek uygulamaların yeri gösterilmektedir. `MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Çizelge 77. Yönetilen .NET içinde TLS uygulamasına ilişkin örnek uygulamaların konumu

IBM MQ.NET stack offering	Örneklerin yeri
Taban.NET	<code>MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\base\SimplePut\SimplePut.cs</code> <code>MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\base\SimpleGet\SimpleGet.cs</code>
XMS .NET	<code>MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\xms\simple\wmq\SimpleProducer\SimpleProducer.cs</code> <code>MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\xms\simple\wmq\SimpleConsumer\SimpleConsumer.cs</code>
WCF için IBM MQ özel kanalı	<code>MQ_INSTALLATION_PATH\Tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\MQMessagingOneWayService.cs</code>

## Windows .NET Monitor olanağının kullanılması

.NET Monitor, IBM MQ tetikleyicisi izleyicisine benzer bir uygulamadır.

**Önemli:** See [Features that can be used only with the primary installation on Windows](#) for important information.

İzlenen bir kuyruğun üzerinde bir ileti alındığında somutlaştırılan ve sonra bu iletiyi işleyen .NET bileşenleri yaratabilirsiniz. .NET Monitor, **runmqdnm** komutu tarafından başlatılır ve **endmqdnm** komutu tarafından durdurulur. Bu komutlara ilişkin ayrıntılar için bkz. [runmqdnm](#) ve [endmqdnm](#).

.NET Monitor olanağını kullanmak için, amqmdnm.dll içinde tanımlanmış olan IMQObjectTrigger arabirimini gerçekleştiren bir bileşen yazıyorsunuz.

Bileşenler işlemsel ya da işlemsel olmayan bir bileşen olabilir. Bir işlemsel bileşen System.EnterpriseServices.ServicedComponent ' den edinilmelidir ve RequiresTransaction ya da SupportsTransaction olarak kaydedilmelidir. .NET Monitor zaten bir işlem başlatmış olduğundan, bu değer RequiresNew olarak kaydedilmemelidir.

Bileşen, **runmqdnm** ' den MQQueueManager, MQQueue ve MQMessage nesnelerini alır. Ayrıca, runmqdnm komutu başlatıldığında -u komut satırı seçeneği kullanılarak bir Kullanıcı Değiştirgesi dizisi de alabilir. Bileşeninizin, bir MQMessage nesnesindeki izlenen kuyruğa gelen bir iletinin içeriğini aldığını unutmayın. Kuyruk yöneticisine bağlanmak, kuyruğu açmak ya da iletinin kendisini almak zorunda değildir. The component must then process the message as appropriate and return control to the .NET Monitor.

If your component has been written as a transactional component, it registers to commit or roll back the transaction using the facilities provided by System.EnterpriseServices.ServicedComponent.

Bileşen MQQueueManager ve MQQueue nesnelerinin yanı sıra iletiyi aldıktan sonra, o iletiye ilişkin bağlam bilgilerini tamamlar ve örneğin, IBM MQ' a ayrı ayrı bağlanmak gerekmeden aynı kuyruk yöneticisine başka bir kuyruk açmasını sağlar.

## Windows Örnek kod parçaları

Bu konuda, .NET Monitor 'dan ileti alan ve bu iletiyi yazdıran 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)
//
// runmqdmn -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)
//
// runmqdmn -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 yazılmış .NET uygulamalarını derlemek için kullanılan örnek komutları.

*MQ\_INSTALLATION\_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

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 .NETkomutunu kullanarak 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 sağlayıcısının sağladığı belgelere bakın.

## Bağımsız IBM MQ .NET istemcisinin kullanılması

IBM MQ 8.0.0 Fix Pack 2' den 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

 IBM MQ 9.1.1' de amqmdnetstd.dll kitaplığı, Windows üzerinde .NET Standard desteği için kullanılabilir (bkz. [“kurmaIBM MQ classes for .NET Standard” sayfa 530](#)). amqmdnet.dll kitaplığı sağlanmaya devam eder, ancak bu kitaplık dengelenir; başka bir deyişle, 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.

 IBM MQ 9.1.2' den amqmdnetstd.dll kitaplığı Linuxüzerinde de bulunur.

## Bu görev hakkında

IBM MQ 8.0.0 Fix Pack 2olanağından, IBM MQ .NET uygulamalarınızı tam IBM MQ istemcisinin kurulu olduğu bir makinede oluşturabilir ve daha sonra uygulamanızla birlikte IBM MQ .NET düzeneğini amqmdnet.dllpaketleyebilir ve üretim sistemlerinde konuşlandırabilirsiniz.

Oluşturup konuşlandırdığınız uygulamalar, geleneksel Windows .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 bağlantırlığı kullanılamaz. Diğer iki kipi kullanma girişimi bir uygulama kural dışı durumuyla sonuçlanır.

## Yordam

Uygulamalarda IBM MQ .NET istemci düzeneğine başvurma

- Uygulamanızdaki amqmdnet.dll düzeneğine, önceki yayınlarda olduğu gibi başvuruda bulunun. amqmdnet düzeneğinin uygulamanın bin dizinine kopyalandığından emin olmak için amqmdnet 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.

Uygulama yapılış kütüğü kullanılarak izlemenin başlatılması ve durdurulması

- İzlemeyi başlatmak ve durdurmak için uygulama yapılış kütüğünü ve IBM MQ ' e özgü bir izleme yapılış kütüğünü kullanın.

Tam IBM MQ istemcisi kuruluşu olmadığından, izlemeyi başlatmak ve durdurmak için kullanılan standart araçlar ( **strmqtrc** ve **endmqtrc** ) kullanılmadığından, uygulama yapılandırma dosyasını ve IBM MQ ' a özgü bir izleme yapılandırma dosyasını kullanmanız gerekir.

### Notlar:

- İzleme oluşturmaya ilişkin bu adımlar, .NET yeniden dağıtılabılır yönetilen istemcinin yanı sıra bağımsız .NET istemcisi için de geçerlidir.
- **V9.1.1** Uygulama yapılandırma dosyası .NET Standardinde desteklenmiyor. IBM MQ .NET Standardüzerinde izlemeyi etkinleştirmek için **MQDOTNET\_TRACE\_ON** ortam değişkenini kullanırsınız.

### Uygulama yapılandırma dosyası (app.config ya da web.config)

Uygulamaların, uygulama yapılandırma dosyasının ( app.config ya da web.config dosyası) <appSettings> bölümü altında **MQTRACECONFIGFILEPATH** özelliğini tanımlaması gerekir. (Uygulama yapılandırma dosyasının gerçek adı, uygulamanızın adına bağlıdır.) **MQTRACECONFIGFILEPATH** özelliğinin değeri, aşağıdaki örnekte gösterildiği gibi, IBM MQ özel izleme yapılış kütüğünün ( mqtrace.config) yerine ilişkin yolu belirtir:

```
<appSettings>
  <add key="MQTRACECONFIGFILEPATH" value="C:\MQTRACECONFIG" />
</appSettings>
```

Belirtilen uygulama yapılandırma dosyası yolunda mqtrace.config dosyası bulunamazsa, izleme devre dışı bırakılır. Ancak, uygulamanın yürürlükteki dizine yazma yetkisi varsa, uygulamanın dizininde First Failure Support Technology (FFST) ve hata günlükleri yaratılır.

## IBM MQ özel izleme yapılandırma kütüğü (mqtrace.config)

mqtrace.config dosyası, izlemeyi başlatma ve durdurma özelliklerini, izleme dosyalarının yolunu ve hata günlüklerinin yolunu tanımlayan bir XML dosyasıdır. Aşağıdaki tabloda bu özellikler açıklanmaktadır.

Çizelge 78. mqtrace.config dosyasında tanımlanan özellikler	
Öznitelik	Açıklama
<b>MQTRACELEVEL</b>	0: İzlemeyi durdurur-bu varsayılan değerdir. 1: İzlemeyi daha az ayrıntıyla başlatır. 2: İzlemeyi tam ayrıntılarla başlatır-önerilir.
<b>MQTRACEPATH</b>	İzleme dosyalarının yaratılacağı bir klasörü gösterir. Yol boşsa ya da <b>MQTRACEPATH</b> özniteliği tanımlanmamışsa, uygulamanın yürürlükteki dizini kullanılır.
<b>MQERRORPATH</b>	Hata günlüğü dosyalarının yaratılacağı bir klasörü gösterir. Yol boşsa ya da <b>MQERRORPATH</b> özniteliği tanımlanmamışsa, uygulamanın yürürlükteki dizini kullanılır.

Aşağıdaki örnekte örnek bir mqtrace.config dosyası gösterilmektedir:

```
<?xml version="1.0" encoding="utf-8"?>
<traceSettings>
  <MQTRACELEVEL>2</MQTRACELEVEL>
  <MQTRACEPATH>C:\MQTRACEPATH</MQTRACEPATH>
  <MQERRORPATH>C:\MQERRORLOGPATH</MQERRORPATH>
</traceSettings>
```

Bir uygulama çalışırken, mqtrace.config dosyasındaki **MQTRACELEVEL** özniteliğinin değeri değiştirilerek izleme dinamik olarak başlatılabilir ve durdurulabilir.

Çalışmakta olan uygulamanın, izleme dosyaları oluşturmak için **MQTRACELEVEL** özniteliğiyle belirtilen klasör için yaratma ve yazma izinleri olmalıdır. Microsoft Azure PaaS ortamında çalışan uygulamalar, Microsoft Azure PaaS içinde çalışan bir IBM MQ .NET düzeneğini kullanan web uygulamalarının oluşturma ve yazma izinleri olmayabileceğinden, benzer erişim izinlerini de sağlamalıdır. Uygulama belirtilen klasör için gerekli yaratma ve yazma izinlerine sahip değilse, izleme oluşturma, ilk başarısızlık verilerini yakalama (FDC) ve hata günlükleri başarısız olur.

Uygulama yapılandırma dosyasında bağ tanımları 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"/>
    </dependentAssembly>
  </assemblyBinding>
</runtime>
```

```
<publisherPolicy apply="no" />
</dependentAssembly>
</assemblyBinding>
</runtime>
```

### İlgili kavramlar

IBM MQ bileşenleri ve özellikleri

Yeniden dağıtılabilen istemciler

“WMQDotnetXAMonitor uygulamasının kullanılması” sayfa 546

WMQDotnetXAMonitor uygulamasının el ile çalıştırılması gerekir. Her an başlatılabilir. You can start it when you see the messages on the SYSTEM.DOTNET.XARECOVERY.QUEUE or you can keep it running in the background before you do any transactional work with the applications that are written using IBM MQ .NET classes.

### İlgili başvurular

[.NET uygulama yürütme ortamı-yalnızca Windows](#)

## XMS .NET uygulamalarının geliştirilmesi

IBM Message Service Client for .NET (XMS .NET) provides an application programming interface (API) called XMS that has the same set of interfaces as the Java Message Service (JMS) API. IBM Message Service Client for .NET contains a fully managed implementation of XMS, which can be used by any .NET compliant language.

### Bu görev hakkında

XMS Destek:

- Noktadan Noktaya İleti Sistemi
- Yayınlama/abone olma ileti alışverişi
- Zaman uyumlu ileti teslimi
- Zaman uyumsuz ileti teslimi

Bir XMS uygulaması, aşağıdaki uygulama tipleriyle ileti alışverişi yapabilir:

- Bir XMS uygulaması
- Bir 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 bu sunucularla bağlantı kurabilirler:

### 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, uygulama ile aracı arasında WebSphere MQ Real-Time Transportile iletilir. İletiler WebSphere MQ Multicast Transportkullanılarak uygulamaya teslim edilebilir.

By connecting to an IBM MQ queue manager, an XMS application can use WebSphere MQ Enterprise Transport to communicate with IBM Integration Bus. Diğer bir seçenek olarak, bir XMS uygulaması IBM MQ' e bağlanarak yayınlayabilir ve abone olabilir.

**V 9.1.1** IBM MQ 9.1.1, IBM MQ , Windows ortamlarındaki uygulamalar için .NET Core ' i destekler. Daha fazla bilgi için bkz “[kullanmaIBM MQ classes for XMS .NET Standard” sayfa 587.](#)

**V 9.1.2** IBM MQ 9.1.2, IBM MQ , Linux ortamlarındaki uygulamalar için .NET Core ' i destekler.

IBM MQ 9.1.4' tan, XMS .NET tarafından yönetilen uygulamalar, kümelenmiş kuyruk yöneticilerindeki bağlantıları otomatik olarak dengelemeyi başarır. .NET Framework ve .NET Standard kitaplıkları desteklenmektedir. Daha fazla bilgi için bkz. [Uniform kümeleri](#) ve [Otomatik uygulama dengeleme](#).

## XMStarafından desteklenen ileti alışverişi stilleri

XMS , ileti alışverişindeki noktadan noktaya iletişim ve yayınlama/abone olma biçimlerini destekler.

İleti alışverişi biçimleri de ileti alışverişi etki alanları olarak adlandırılır.

### Noktadan Noktaya İleti Sistemi

Noktadan noktaya ileti alışverişi için ortak bir form kuyruğa alma işlevini kullanır. En basit durumda, bir uygulama, örtük ya da belirtik olarak bir hedef kuyruğu belirleyerek başka bir uygulamaya ileti gönderir. Temeldeki ileti alışverişi ve kuyruğa alma sistemi, iletiyi gönderme uygulamasından alır ve iletiyi hedef kuyruğuna yönlendirir. Alma uygulaması iletiyi kuyruktan alabilir.

Temeldeki ileti sistemi ve kuyruğa alma sistemi IBM Integration Busiçeriyorsa, IBM Integration Bus iletinin bir iletiyi eşleyebilir ve iletinin kopyalarını farklı kuyruklara yönlendirebilir. Sonuç olarak, iletiyi birden çok uygulama alabiliyor. IBM Integration Bus ayrıca bir iletiyi dönüştürebilir ve ona veri ekleyebilirler.

Noktadan noktaya iletişim mesajının önemli bir özelliği, bir uygulamanın ileti gönderirken yerel bir kuyruğa ileti yerleştirmesi olabilir. Temeldeki ileti sistemi ve kuyruğa alma sistemi, iletinin hangi hedef kuyruğunu göndereceğini belirler. Alma uygulaması, iletiyi hedef kuyruktan alır.

### Yayınlama/abone olma ileti alışverişi

Yayınlama/abone olma ileti sisteminde, iki tip uygulama vardır: yayınlayıcı ve abone.

Bir *yayınlayıcı* bilgi kaynağı, yayın iletileri biçiminde bilgi sağlar. Bir yayınlayıcı bir iletiyi yayınladığında, bu ileti, iletinin içindeki bilgilerin konusunu tanımlayan bir konu belirtir.

*abone* , yayınlanan bilgilerin tüketicidir. Abone, abonelikler yaratarak ilgilendiği konuları belirtir.

Yayınlama/abone olma sistemi, yayıncılardan ve aboneliklerden gelen abonelikleri alır. Yayınları abonelere yönlendiriyor. Bir abone, yalnızca abone olduğu konuların yayınlarını alır.

Yayınlama/abone olma ileti sisteminin önemli bir özelliği, yayıncının bir iletiyi yayınlarken bir konuyu tanıması. Bu, aboneleri tanımaz. Abonesi olmayan bir konuyla ilgili bir ileti yayınlanırsa, hiçbir uygulama iletiyi almaz.

Bir uygulama hem yayınlayıcı hem de bir 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

Bir ConnectionFactory nesnesi, bir bağlantı için bir parametre kümesini sarsalıyor. Uygulama, bağlantı yaratmak için ConnectionFactory kullanır. Bir uygulama çalışma zamanında parametreleri sağlayabilir ve bir ConnectionFactory nesnesi oluşturabilirler. Diğer bir seçenek olarak, bağlantı değiştiricileri denetimli nesnelere havuzunda saklanabilir. Bir uygulama, havuzdan bir nesneyi alabilir ve ondan bir ConnectionFactory nesnesi yaratabilir.

#### Bağlantı

Bir Connection nesnesi, bir uygulamadan bir ileti alışverişi sunucusuna etkin bir bağlantıyı sokardır. Uygulama, oturum yaratmak için bir bağlantı kullanır.

## Hedef

Bir uygulama, Destination nesnesini kullanarak ileti gönderir ya da ileti alır. Yayınlama/abone olma etki alanında, bir Destination nesnesi bir konuyu ve noktadan noktaya etki alanında bir Destination nesnesini sarsalır ve bir kuyruğu sarsalır. Uygulama, yürütme sırasında Destination nesnesi yaratmak için gereken parametreleri sağlayabilir. Diğer bir seçenek olarak, yönetilen nesnelere havuzunda saklanan bir nesne tanımlamasından bir Destination nesnesi de yaratabilir.

## Session

Session nesnesi, ileti göndermek ve almak için tek bir iş parçacıklı bağlamdır. Uygulama, Message, MessageProducer ve MessageConsumer nesnelere oluşturmak için bir Session nesnesi kullanır.

## İleti

Bir Message nesnesi, bir uygulamanın MessageProducer nesnesi kullanarak gönderdiği ya da bir MessageConsumer nesnesi kullanılarak aldığı Message nesnesini sarmadır.

## MessageProducer

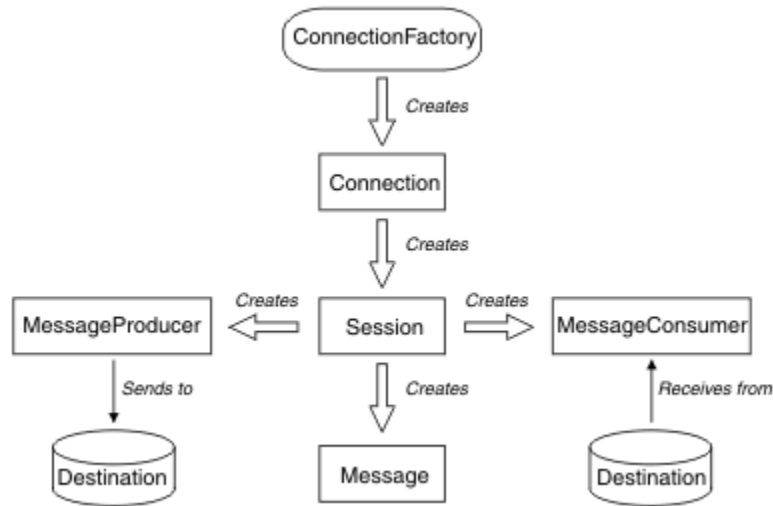
MessageProducer nesnesi, bir uygulamaya ileti göndermek için bir uygulama tarafından kullanılır.

## MessageConsumer

MessageConsumer nesnesi, bir hedef için gönderilen iletileri almak için bir uygulama tarafından kullanılır.

## XMS nesneleri ve ilişkileri

Şekil 58 sayfa 583 shows the main types of XMS object: ConnectionFactory, Connection, Session, MessageProducer, MessageConsumer, Message, and Destination. Uygulama, bağlantı yaratmak için bir bağlantı üreticisi kullanır ve oturum yaratmak için bir bağlantı kullanır. Daha sonra, uygulama, ileti, ileti üreticileri ve ileti tüketicileri oluşturmak için bir oturumu kullanabilir. Uygulama, bir hedefe ileti göndermek için bir ileti üreticisi kullanır ve bir hedefe gönderilen iletileri almak için bir ileti tüketicisi kullanır.



Şekil 58. XMS nesneleri ve ilişkileri

XMS .NET' ta XMS sınıfları, .NET arabirimleri kümesi olarak tanımlanır. XMS .NET uygulamalarını kodlarken yalnızca bildirilen arabirimlere gerek duyarsınız.

XMS nesne modeli, Java Message Service Specification, Sürüm 1.1 içinde açıklanan etki alanı bağımsız arabirimlerine dayalıdır. Etki alanına özgü sınıflar ( Topic, TopicPublisher ve TopicSubscriber gibi) sağlanmaz.

## XMS nesnelere ilişkin özellikleri ve özellikleri

Bir XMS nesnesi, nesneye ilişkin özellikleri olan, farklı şekillerde uygulanan özellikleri ve özelliklere sahip olabilir:

## Öznitelikler

Öznitelik bir değer olmasa bile, her zaman var olan ve depolamayı kaplayan bir nesne karakteristiğidir. Bu bakımdan, öznitelik sabit uzunluklu veri yapısındaki bir alana benzer. Özniteliklerin ayırt edici özelliği, her özniteliğin değerini ayarlamak ve almak için kendi yöntemlerine sahip olması.

## Özellikler

Bir nesnenin özelliği var olur ve yalnızca değeri ayarlandıktan sonra depolanır. Bir özellik silinemez ya da değeri ayarlandıktan sonra depolama birimi kurtarılamaz. Değerini değiştirebilirsiniz. XMS , özellik değerlerini ayarlamak ve almak için bir dizi soysal yöntem sağlar.

## Denetlenen nesnelere

Denetlenen nesnelere kullanarak, merkezi bir havuzdan yönetilecek istemci uygulamaları tarafından kullanılan bağlantı ayarlarını yönetebilirsiniz. Uygulama, merkezi havuzdan nesne tanımlarını alır ve bunları `ConnectionFactory` ve `Destination` nesnelere oluşturmak için kullanır. Yönetilen nesnelere kullanarak, çalıştırma sırasında kullandıkları kaynaklardan birkaç uygulama da yapabilirsiniz.

Örneğin, XMS uygulamaları, bir test ortamındaki bağlantılar ve hedefler kümesine başvuruda bulunan yönetilen nesnelere yazılıp test edilebilir. Uygulamalar konuşlandırıldığında, yönetilen nesnelere, uygulamaların üretim ortamındaki bağlantılara ve hedeflere gönderme yapmak üzere yapılandırılacak şekilde değiştirilebilir.

XMS , denetlenen iki nesneyi destekler:

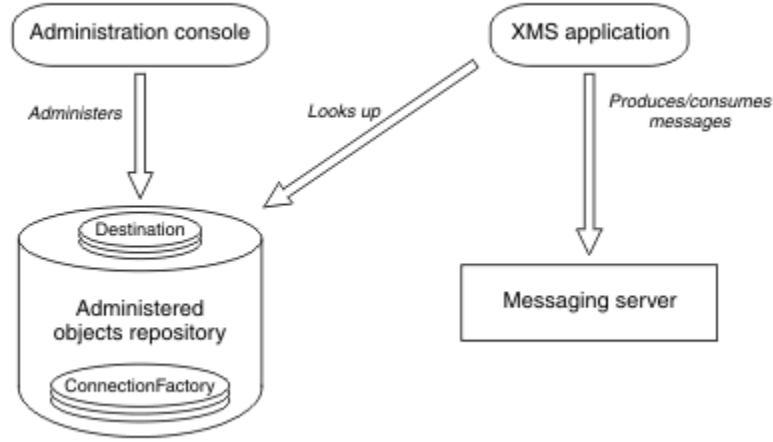
- Sunucu ile ilk bağlantıyı yapmak için uygulamalar tarafından kullanılan bir `ConnectionFactory` nesnesi.
- Gönderilmekte olan iletilere ilişkin hedefi ve alınmakta olan iletilerin kaynağını belirlemek için uygulamalar tarafından kullanılan bir `Destination` nesnesi. Hedef, bir uygulamanın bağlı olduğu sunucudaki bir konu ya da kuyruğdur.

Yönetim aracı **JMSAdmin** , IBM MQ ile birlikte sağlanır. It is used to create and manage administered objects in a central repository of administered objects.

Havuzdaki denetlenen nesnelere, IBM MQ classes for JMS ve XMS uygulamaları tarafından kullanılabilir. XMS applications can use the `ConnectionFactory` and `Destination` objects to connect to an IBM MQ queue manager. Denetimci, havuzda tutulan nesne tanımlamalarını uygulama kodunu etkilemeden değiştirebilir.

Aşağıdaki çizgede, bir XMS uygulamasının tipik olarak yönetilen nesnelere nasıl kullandığı gösterilmektedir. Çizgenin sol tarafında, bir denetim konsolu kullanılarak denetlenen `ConnectionFactory` (`ConnectionFactory`) ve hedef nesne tanımlamaları içeren bir havuz gösterilir. Çizgenin sağ tarafı, havuzdaki nesne tanımlamalarını gösteren bir XMS uygulamasını gösterir ve ardından bir ileti alışverişi sunucusuna bağlanırken bu nesne tanımlamalarını kullanır.





Şekil 59. Typical use of administered objects by an XMS application

## XMS ileti modeli

XMS ileti modeli, IBM MQ classes for JMS ileti modeliyle aynıdır.

Özellikle XMS, IBM MQ classes for JMS' in uyguladığı ileti üstbilgi alanlarını ve ileti özelliklerini uygular:

- JMS üstbilgi alanları. Bu alanlar, JMS önekiyle başlayan adlara sahiptir.
- JMS tanımlı özellikler. Bu alanlarda, adları JMSX önekiyle başlayan özellikler vardır.
- IBM tanımlı özellikler. Bu alanlarda, adları JMS\_IBM\_ önekiyle başlayan özellikler vardır.

As a result, XMS applications can exchange messages with IBM MQ classes for JMS applications. Her iletide üstbilgi alanları ve özellikleri uygulama tarafından ayarlanır ve diğerleri XMS ya da IBM MQ classes for JMS tarafından ayarlanır. Some of the fields set by XMS or IBM MQ classes for JMS are set when the message is sent, and others when it is received. Üstbilgi alanları ve özellikleri, uygun olduğu durumlarda bir ileti alışverişi sunucusu aracılığıyla bir iletiyle geçirilir. Bunlar, iletiyi alan herhangi bir uygulama tarafından kullanılabilir kılınabilirler.

### İlgili kavramlar

[IBM MQ classes for JMS](#)

## İleti alışverişi sunucusu ortamının ayarlanması

The topics in this section describe how to set up the messaging server environment to allow XMS applications to connect to a server.

### Bu görev hakkında

Bir IBM MQ kuyruk yöneticisine bağlanan uygulamalar için, IBM MQ istemcisi (ya da bağ tanımları kipine ilişkin kuyruk yöneticisi) gereklidir.

Şu anda bir aracıya gerçek zamanlı bağlantı kullanan uygulamalar için herhangi bir önkoşul yok.

You must set up the messaging server environment before running any XMS applications, including the sample applications provided with XMS.

Bu bölüm aşağıdaki konuları içerir:

- [“IBM MQ kuyruk yöneticisine bağlanan bir uygulama için kuyruk yöneticisi ve aracısının yapılandırılması” sayfa 591](#)
- [V9.1.1 “kullanmaIBM MQ classes for XMS .NET Standard” sayfa 587](#)
- [“Aracıya gerçek zamanlı bağlantı kullanan bir uygulama için aracı yapılandırılması” sayfa 592](#)

- “WebSphere Application Server ile bağlantı sağlayan bir uygulama için hizmet bütünleştirme veriyolunun yapılandırılması” sayfa 593

## XMS .NET içindeki ileti dinleyicileri

İletiyi zamanuyumsuz olarak almak için bir ileti dinleyici kullanılır. `MessageConsumer.receive()` çağrılarında farklı olarak, ileti dinleyici çağırma iş parçacığını engellemez; bunun yerine, genellikle **onMessage** yöntemi olan, uygulama tarafından belirtilen bir geri çağırma yöntemine ileti 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 hale geldikten sonra, 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.Closed` içinde).

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.

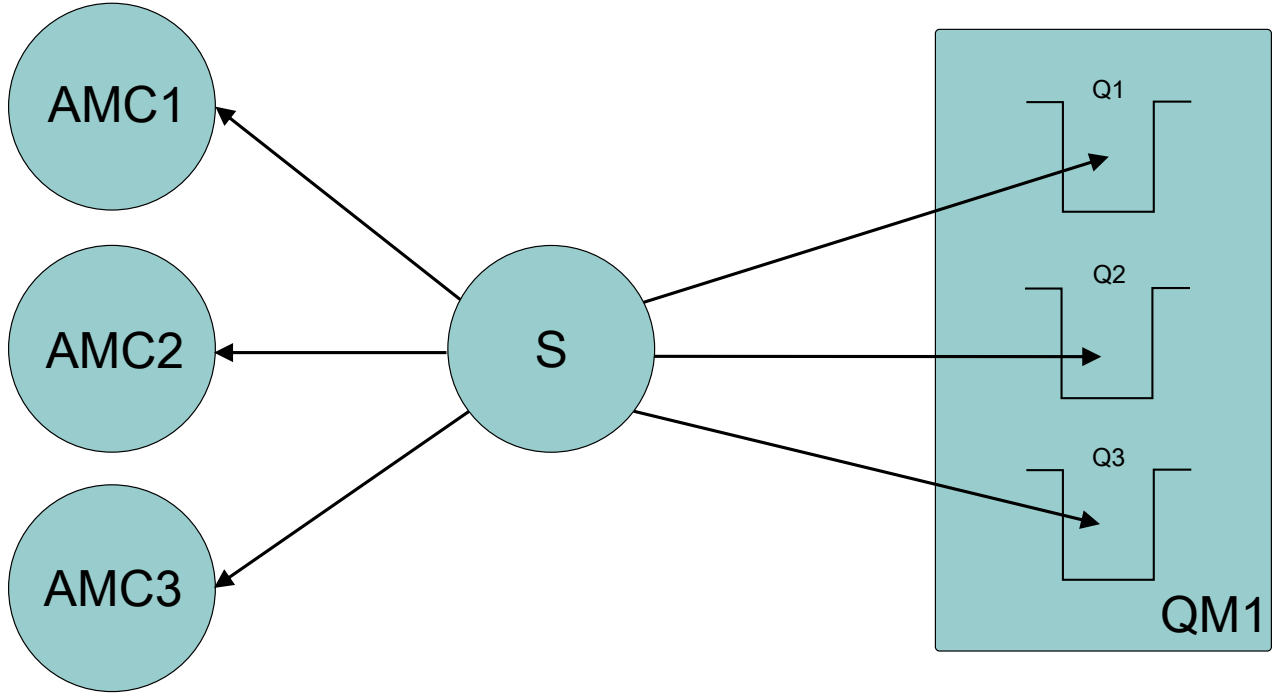
İkinci ileti, yalnızca **onMessage()** yöntemi geri döndükten sonra oturumdaki bir tüketiciye teslim edilir. Bunun nedeni, bir oturumun yalnızca tek bir iş parçacığı kullanarak tüketicilere ileti teslimini yönetmesi olabilir. Bu, aynı anda yalnızca bir iletinin teslim edilebileceği ve tüketicinin herhangi 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.

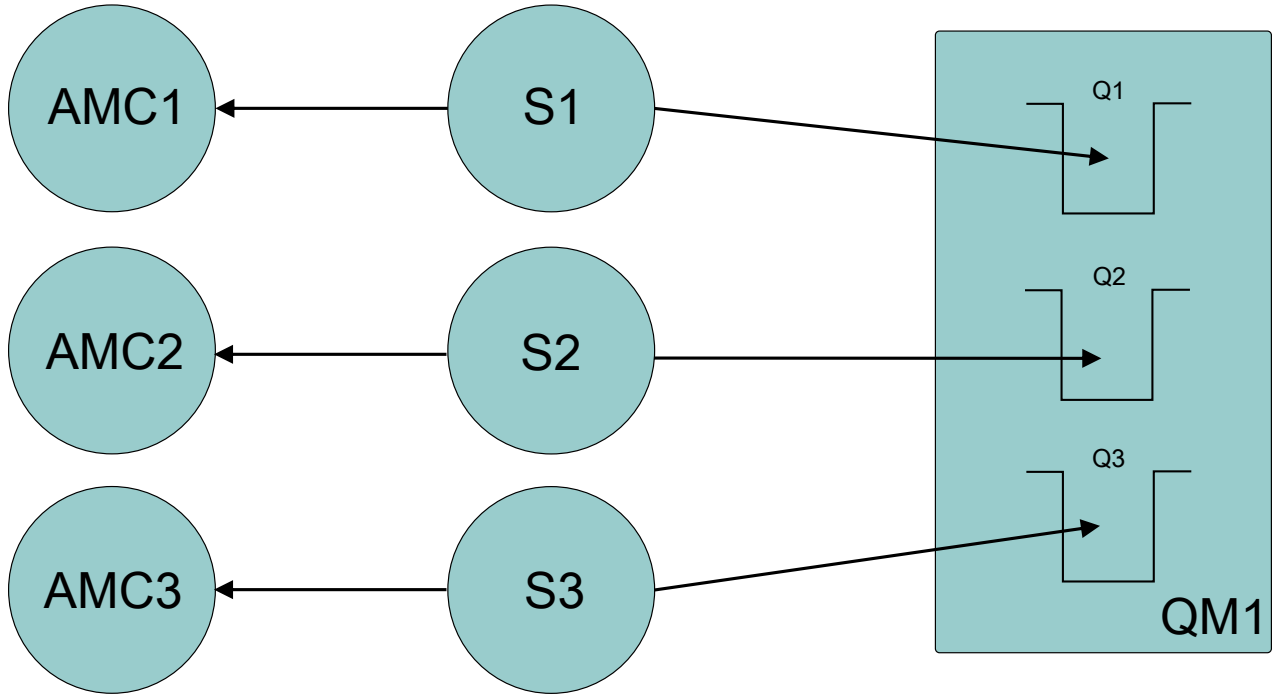
Aşağıdaki örnekler bu özelliği daha net bir şekilde göstermektedir.

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.



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.



Bu, eşzamanlı ileti tesliminize gereksinim duyarsanız, birden çok oturuma ihtiyacınız olduğunu gösterir.

**Linux Windows V 9.1.1 kullanmaIBM MQ classes for XMS .NET Standard**

XMS ürününü Microsoft .NET Standardile nasıl kullanacağınız ve IBM MQ classes for XMS .NET Framework ile IBM MQ classes for XMS .NET Standardarasındaki farklar. IBM MQ classes for XMS .NET Standardiçin Microsoft.NET Core önkoşulu vardır.

## amqmxsstd.dll kitaplık

**Windows** IBM MQ 9.1.1' den IBM MQ classes for XMS .NET Standard kitaplığı amqmxsstd.dll, Windows üzerinde XMS .NET Standard desteği için kullanılabilir.

**V 9.1.2** **Linux** IBM MQ 9.1.2' den amqmxsstd.dll kitaplığı Linux üzerinde de bulunur. Kitaplık, Linux üzerine bir IBM MQ istemcisi kurulduğunda /MQINSTALL\_PATH/lib64 path içine kurulur. XMS .NET örnekleri MQINSTALL\_PATH/samp/dotnet/samples/cs/core/xmsiçinde bulunur.

Daha fazla bilgi için bkz “kurmaIBM MQ classes for .NET Standard” sayfa 530.



**Uyarı:** Tüm IBM .XMS.\* kitaplıkları sağlanmaya devam eder, ancak bu kitaplıklar dengelenir; yani, bu kitaplıklara yeni özellikler eklenmez.

En son özelliklerden herhangi biri için amqmxsstd.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 9.1.4** IBM MQ 9.1.4' den IBM MQ classes for XMS .NET Standard , NuGet havuzundan karşıdan yüklenebilir. NuGet paketi hem amqmxsstd.dll kitaplığını hem de amqmdnetstd.dll kitaplığını içerir. amqmxsstd.dll , amqmdnetstd.dll ' e bağlıdır ve XMS .NET Core uygulaması paketlenirken, hem amqmxsstd.dll hem de amqmdnetstd.dll , XMS .NET Core uygulamasıyla birlikte paketlenmelidir. Daha fazla bilgi için bkz “Downloading IBM MQ classes for XMS .NET Standard from the NuGet repository” sayfa 590.

## dspmqr DELETE ...



IBM MQ 9.1.1olanağından, .NET Core bileşenine ilişkin sürüm ve oluşturma bilgilerini görüntülemek için **dspmqr** komutunu kullanabilirsiniz.

## IBM MQ classes for XMS .NET Framework ve IBM MQ classes for XMS .NET Standard özellikleri

Aşağıdaki çizelgede, IBM MQ 9.1.1 for IBM MQ classes for XMS .NET Framework ve IBM MQ classes for XMS .NET Standardürünlerinde geçerli olan özellikler listelenmektedir.

Çizelge 79. IBM MQ classes for XMS .NET Framework ve IBM MQ classes for XMS .NET Standard özellikleri arasındaki farklar		
Özellik	IBM MQ classes for XMS .NET Framework	IBM MQ classes for XMS .NET Standard
Sınıf Adları (API ' ler)	Tüm sınıflar her ağda aynı kalır.	Tüm sınıflar her ağda aynı kalır.
İşletim Sistemi	Windows	Windows Dockerized konteynerleri <b>V 9.1.2</b> Linux <b>V 9.1.3</b> 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 , .NET Standardiçinde desteklenmez. app.configyerine ortam değişkenlerini kullanmanız gerekir.

Çizelge 79. IBM MQ classes for XMS .NET Framework ve IBM MQ classes for XMS .NET Standard özellikleri arasındaki farklar (devamı var)

Özellik	IBM MQ classes for XMS .NET Framework	IBM MQ classes for XMS .NET Standard
Takip edin	XMS .NET istemcisini izlemek için, izlemeyi etkinleştirmek için kullanılan <b>XMS_TRACE_ON</b> ortam değişkeni gibi var olan ortam değişkenlerini kullanabilirsiniz. Daha fazla bilgi için <u>XMS ortam değişkenlerini kullanarak XMS .NET izlemesini yapılandırma</u> başlıklı konuya bakın.  app.config dosyası, yeniden dağıtılabılır pakete ilişkin izlemeyi etkinleştirmek için kullanılabilir.	XMS .NET istemcisini izlemek için, izlemeyi etkinleştirmek için kullanılan <b>XMS_TRACE_ON</b> ortam değişkeni gibi var olan ortam değişkenlerini kullanabilirsiniz. Daha fazla bilgi için <u>XMS ortam değişkenlerini kullanarak XMS .NET izlemesini yapılandırma</u> başlıklı konuya bakın.
İletim Kipleri	Yönetilen, Yönetilmeyen ve Bağ Tanımları	Yönetilen
TLS	Windows anahtar deposu, sertifikaları saklamak için kullanılır.	 Windows üzerinde, 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 üzerinde, 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 nesnelere	LDAP ve FileSystem ' i destekler	Yalnızca FileSystem ' i destekler

## İlgili görevler

“XMS örnek uygulamalarının kullanılması” sayfa 594

XMS .NET örnek uygulamaları, her bir API ' nin ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluş ve ileti alışverişi sunucusu kurulumunuzu doğrulamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

## Linux Windows V 9.1.4 Downloading IBM MQ classes for XMS .NET Standard from the NuGet repository

From IBM MQ 9.1.4, the IBM MQ classes for XMS .NET Standard are available for downloading from the NuGet repository, so that they can be easily consumed by .NET Developers.

## Bu görev hakkında

NuGet is the package manager for Microsoft development platforms including .NET. NuGet istemci araçları, paketleri üretme ve kullanma yeteneği sağlar. NuGet paketi, derlenmiş kodu (DDL ' ler) içeren .nupkg uzantısına sahip tek bir sıkıştırılmış dosyadır ve bu kodla ilgili diğer dosyalar ve paketin sürüm numarası gibi bilgileri içeren açıklayıcı bir bildirgedir.

IBM MQ 9.1.4olanağından, 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 Galerisinden yükleyebilirsiniz.

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 kuruludur.
- From the command line using either the NuGet Package Manager or the .NET CLI.
- Web tarayıcısı kullanarak.

As for the redistributable package, you enable trace by using the environment variable **XMS\_TRACE\_ON**.

## Yordam

- To download the IBMXMSDotnetClient package by using the Package Manager UI in Microsoft Visual Studio, complete the following steps:
  - a) .NET projesini farenin sağ düğmesiyle tıklatın ve **Nuget Paketleri Yönet**ögesini seçin.
  - b) **Göz At** sekmesini tıklatın ve “IBMXMSDotnetClient” i arayın.
  - c) Paketi seçin ve **Install(Kur)** düğmesini tıklatın.

Kuruluş sırasında, Package Manager, konsol deyimleri biçiminde ilerleme bilgileri sağlar.

- IBMXMSDotnetClient paketini komut satırından yüklemek için aşağıdaki seçeneklerden birini seçin:
  - NuGet Package Manager olanağını kullanarak aşağıdaki komutu girin:

```
Install-Package IBMXMSDotnetClient -Version 9.1.4.0
```

Kuruluş sırasında, Package Manager, konsol deyimleri biçiminde ilerleme bilgileri sağlar. Çıkışı bir günlük dosyasına yeniden yönlendirebilirsiniz.

- .NET CLI ' yı kullanarak aşağıdaki komutu girin:

```
dotnet add package IBMXMSDotnetClient --version 9.1.4
```

- Bir web tarayıcısını kullanarak, IBMXMSDotnetClient paketini <https://www.nuget.org/packages/IBMXMSDotnetClient>olanağından yükleyin.

## İlgili kavramlar

“kurmaIBM MQ classes for .NET Standard” sayfa 530

Örnekler de dahil olmak üzere IBM MQ 9.1.1, IBM MQ classes for .NET Standard'den Windows' ta IBM MQ ile kurulur. **V 9.1.2** IBM MQ 9.1.2, IBM MQ classes for .NET Standard , Linux platformlarında da kullanılabilir. There is a prerequisite of Microsoft.NET Core for IBM MQ classes for .NET Standard.

### İlgili görevler

“IBM MQ classes for .NET Standard dosyasını NuGet havuzundan yükleme” sayfa 533 IBM MQ 9.1.4' den IBM MQ classes for .NET Standard , .NET Developers tarafından kolayca kullanılabilmesi için NuGet havuzundan yüklenebilir.

### İlgili başvurular

[IBM MQ Client for .NET lisans bilgileri](#)

## IBM MQ kuyruk yöneticisine bağlanan bir uygulama için kuyruk yöneticisi ve aracısının yapılandırılması

Bu bölümde, IBM WebSphere MQ 7.0.1ya da daha sonraki bir sürümü kullandığınız varsayılır. Bir IBM MQ kuyruk yöneticisine bağlanan bir uygulamayı çalıştırabilmeniz için, kuyruk yöneticisini yapılandırmanız gerekir. Bir yayınlama/abone olma uygulaması için Queued yayınlama/abone olma arabirimini kullanıyorsanız, bazı ek yapılandırmalara gerek vardır.

### Başlamadan önce

XMS , IBM Integration Bus ya da WebSphere Message Broker 6.1 ile çalışır ya da daha sonra

Bu görevi başlatmadan önce aşağıdaki adımları gerçekleştirin:

- Uygulamanızın çalışmakta olan bir kuyruk yöneticisine erişimi olduğundan emin olun.
- If your application is a publish/subscribe application and uses Queued publish/subscribe interface, make sure that **PSMODE** attribute is set to ETKIN on the queue manager.
- Uygulamanızın, özellikleri kuyruk yöneticisine bağlanmak için 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

You configure the queue manager and broker to run XMS applications in the same way that you configure the queue manager and queued publish/subscribe interface to run IBM MQ JMS applications. Aşağıdaki adımlar, yapmanız gerekenleri özetlemektedir.

### Yordam

1. Kuyruk yöneticisinde, uygulamanızın gereksinim duyduğu kuyrukları yaratın.

Kuyruklar yaratmanıza genel bir bakış için [Kuyrukların tanımlanması](#) başlıklı konuya bakın.

Uygulamanız bir yayınlama/abone olma uygulamasıysa ve IBM MQ classes for JMS sistem kuyruklarına erişmesi gereken Queued yayınlama/abone olma arabirimini kullanıyorsa, kuyrukları yaratmadan önce Adım 4a ' ya kadar bekleyin.

2. Uygulamanızla ilişkili kullanıcı kimliğine, kuyruk yöneticisine bağlanma yetkisi ve kuyruklara erişmek için uygun yetkiyi verin.

Yetkilendirmeye ilgili genel bilgi edinmek için [Securing](#) konusuna bakın. If your application connects to the queue manager in client mode, see also [İstemciler ve sunucular](#).

3. Uygulamanızın istemci kipindeki kuyruk yöneticisine bağlanması durumunda, kuyruk yöneticisinde bir sunucu bağlantı kanalı tanımlandığından ve bir dinleyici başlatıldığından emin olun.

Kuyruk yöneticisine bağlanan her uygulama için bu adımı gerçekleştirmenize gerek yoktur. Bir sunucu bağlantısı kanal tanımlaması ve bir dinleyici, istemci kipinde bağlanan tüm uygulamaları destekleyebilir.

4. Uygulamanız bir yayınlama/abone olma uygulaması ise ve Queued yayınlama/abone olma arabirimini kullanıyorsa, aşağıdaki adımları gerçekleştirin.

- a) Kuyruk yöneticisiyle birlikte, IBM MQ ile birlikte 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 bulacağı ve nasıl çalıştırılacağı hakkında bilgi için bkz. [IBM MQ classes for Java](#).

Bu adımı kuyruk yöneticisi için yalnızca bir kez gerçekleştirin. 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 erişme yetkisi verin.

Kullanıcı kimliği gereksinimlerinin ne olduğu hakkında bilgi için bkz. [Using IBM MQ classes for JMS](#).

- c) For a broker of IBM Integration Bus or WebSphere Message Broker, create and deploy a message flow to service the queue where applications send messages that they publish.

Temel ileti akışı, iletileri yayınlamak için yayınlanan iletileri ve bir Yayın iletisi işleme düğümünü okumak için bir MQInput iletisi işleme düğümlerinden oluşur.

For information about how to create and deploy a message flow, see the IBM Integration Bus or WebSphere Message Broker product documentation available from the [IBM Integration Bus ürün belgeleri kitaplığı web sayfası](#).

Aracıda uygun bir ileti akışı önceden konuşlandırıldıysa, bu adımı gerçekleştirmenize gerek yoktur.

## Sonuçlar

Artık uygulamanıza başlayabilirsiniz.

## Aracıya gerçek zamanlı bağlantı kullanan bir uygulama için aracı yapılandırılması

Bir aracıya gerçek zamanlı bağlantı kullanan bir uygulamayı çalıştırabilmeniz için, o aracıyı yapılandırmanız gerekir.

### Başlamadan önce

Bu görevi başlatmadan önce aşağıdaki adımları gerçekleştirdiniz:

- Uygulamanızın çalışmakta olan bir aracıya erişiminin olduğundan emin olun.
- Uygulamanızın özellikleri, bir aracıyla gerçek zamanlı bağlantı için uygun şekilde ayarlanmış bir bağlantı üreticisi kullandığından emin olun. Bir bağlantı üreticisinin özellikleri hakkında daha fazla bilgi için bkz. [ConnectionFactory özellikleri](#).

### Bu görev hakkında

You configure a broker to run XMS applications in the same way that you configure a broker to run IBM MQ classes for JMS applications. Yapmanız gerekenleri özetlemek için aşağıdaki adımları gerçekleştirin:

### Yordam

1. Bir aracının dinlediği TCP/IP kapısındaki iletileri okumak ve iletileri yayınlamak için ileti akışı oluşturun ve konuşlandırın.

Bunu aşağıdaki yollardan biriyle de yapabilirsiniz:

- **Real-timeOptimizedFlow** ileti işleme düğümü içeren bir ileti akışı yaratın.



- Bir **Real-timeInput** ileti işleme düğümü ve bir Yayın iletisi işleme düğümü içeren bir ileti akışı yaratın.

Gerçek zamanlı bağlantılar için kullanılan bağlantı noktasını dinlemek için **Real-timeOptimizedFlow** ya da **Real-timeInput** düğümünü yapılandırmanızdır. XMS' ta, gerçek zamanlı bağlantılar için varsayılan kapı numarası 1506 'tır.

Aracıda uygun bir ileti akışı önceden konuşlandırıldıysa, bu adımı gerçekleştirmenize gerek yoktur.

2. If you require messages to be delivered to your application using IBM MQ classes for JMS, configure the broker to enable multicast. Güvenilir çok noktaya 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 aracıyı bu bilgileri kullanarak uygulamanızın kimliğini doğrulamamasını istiyorsanız, basit Telnet benzeri parola kimlik doğrulaması için kullanıcı adı sunucusunu ve aracıyı yapılandırın.

## Sonuçlar

Artık uygulamanıza başlayabilirsiniz.

## WebSphere Application Server ile bağlantı sağlayan bir uygulama için hizmet bütünleştirme veriyolunun yapılandırılması

Bir WebSphere Application Server service integration technologies hizmet bütünleştirme veriyoluna bağlanan bir uygulamayı çalıştırabilmeniz için, hizmet bütünleştirmesini, varsayılan ileti alışverişi sağlayıcısını kullanan JMS uygulamalarını çalıştırmak için hizmet tümleştirme veriyolunu yapılandırdığınız şekilde yapılandırmanızdır.

## Başlamadan önce

Bu görevi başlatmadan önce aşağıdaki adımları izlemelisiniz:

- Bir ileti alışverişi veriyolunun yaratıldığından ve sunucunuzun veriyolu üyesi olarak veriyoluna eklendiğinden emin olun.
- Uygulamanızın, çalışmakta olan en az bir ileti alışverişi altyapısı içeren bir hizmet tümleştirme veriyoluna erişimi olduğundan emin olun.
- HTTP işlemi gerekliyse, bir HTTP ileti alışverişi altyapısı gelen iletim kanalı tanımlanmalıdır. Varsayılan olarak, SSL ve TCP kanalları, sunucu kuruluşu sırasında tanımlanır.
- Uygulamanızın, özellikleri bir önyükleme sunucusu kullanılarak hizmet tümleştirme veriyoluna bağlanmak için uygun şekilde ayarlanmış bir bağlantı üreticisi kullandığından emin olun. Gereken en düşük bilgi aşağıdaki gibi olmalıdır:
  - İleti alışverişi sunucusuyla (önyükleme sunucusu aracılığıyla) bir bağlantı kararlaştırılırken kullanılacak yeri ve protokolü tanımlayan sağlayıcı uç noktası. En basit haliyle, 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

Gerekli olan kuyruk ya da konu alanları tanımlanmalıdır. Varsayılan olarak, sunucu kuruluşu sırasında Default.Topic.Space adı verilen bir konu alanı tanımlanır, ancak ek konu alanları gerekiyorsa, bu konu alanlarını kendiniz yaratmanızdır. Bir konu alanında tek tek konuları önceden tanımlamanıza gerek yoktur; sunucu bu konuları gereken şekilde somutlaştırdığından, bu konuları devingen olarak somutlaştırabilirsiniz.

Aşağıdaki adımlar, yapmanız gerekenleri özetlemektedir.

## Yordam

1. Uygulamanızın noktadan noktaya ileti alışverişi için gereksinim duyduğu kuyrukları yaratın.
2. Uygulamanızın yayınlama/abone olma ileti alışverişi için gereksinim duyduğu ek konu alanlarını yaratın.

## Sonuçlar

Artık uygulamanıza başlayabilirsiniz.

## XMS örnek uygulamalarının kullanılması

XMS .NET örnek uygulamaları, her bir API ' nin ortak özelliklerine genel bir bakış sağlar. Bunları, kuruluş ve ileti alışverişi sunucusu kurulumunuzu doğrulamak ve kendi uygulamalarınızı oluşturmanıza yardımcı olmak için kullanabilirsiniz.

### Bu görev hakkında

Kendi uygulamalarınızı yaratmak için yardıma gerek 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 (ConnectionFactory, Connection, Session, Destination, Destination, bir üretici ya da bir tüketici ya da her ikisi) yaratmak ve uygulamanızın nasıl çalışmasını istediğinizi belirtmek için gerekli olan belirli özellikleri yaratmak için gereken temel adımları tanımlayın. Daha fazla bilgi için bkz “XMS uygulamaları yazılıyor” sayfa 597. Örnekler, XMS' un gelecekteki yayın düzeylerinde değişikliğe tabidir.

Aşağıdaki çizelge, XMS ile birlikte verilen örnek uygulamaların kümelerini (her bir API için bir tane) göstermektedir.

Çizelge 80. XMS .NET için örnek uygulamalar	
Örneklerin adı	Tanım
SampleConsumerCS	Bir kuyruktan ileti alan ya da bir konuya abone olan ileti tüketici uygulaması.
SampleProducerCS	Bir kuyruğa ya da konuya ilişkin iletiler üreten bir ileti üreticisi uygulaması.
SampleConfigCS	Dosya tabanlı, yönetilen bir nesne havuzu oluşturmak için kullanabileceğiniz bir yapılandırma uygulaması. Uygulama, belirli bağlantı ayarlarınız için bir bağlantı üreticisi ve hedefi içerir. Bu denetlenen nesne havuzu, örnek tüketici ve üretici uygulamalarının her biriyle birlikte kullanılabilir.

Çeşitli API ' lerde aynı işlevleri destekleyen örnekler, sözdizimsel farklılıklara sahiptir.

- Örnek ileti tüketici ve üretici uygulamalarının her ikisi de aşağıdaki işlevleri destekler:
  - IBM MQ, IBM Integration Bus (bir aracıya gerçek zamanlı bağlantı kullanarak) ve bir WebSphere Application Server service integration busbağlantısı
  - İlk bağlam arabirimini kullanarak nesne havuzu aramalarını denetleyerek
  - Connections to queues (IBM MQ and WebSphere Application Server service integration bus) and topics (IBM MQ, real-time connection to a broker, and WebSphere Application Server service integration bus)
  - Temel, bayt, harita, 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 işlev görebilirler:

## Basit kip

Örnekleri en az kullanıcı girişi ile çalıştırabilirsiniz.

## İleri düzey kip

Örneklerin çalışacağı şekilde daha ayrıntılı bir şekilde özelleştirebilirsiniz.

Tüm örnekler uyumlu ve bu nedenle diller arasında çalışabilir.

**Windows** IBM MQ 9.1.1, IBM MQ , Windows Core for XMS .NET uygulamaları için .NET Core olanağını destekler. Örnekler de içinde olmak üzere IBM MQ classes for .NET Standard, standart IBM MQ kurulumunun bir parçası olarak varsayılan olarak kurulur.

**V 9.1.2** **Linux** From IBM MQ 9.1.2, IBM MQ also supports .NET Core for applications in Linux environments.

XMS .NET için örnek uygulamalar, &MQINSTALL\_PATH&/smp/dotnet/samples/cs/core/xmsiçine kurulur.

Daha fazla bilgi için bkz "[kullanma IBM MQ classes for XMS .NET Standard](#)" sayfa 587.

## .NET örnek uygulamalarının çalıştırılması

.NET örnek uygulamalarını, basit ya da gelişmiş kipte etkileşimli olarak ya da otomatik olarak oluşturulan ya da özelleştirilmiş yanıt dosyaları kullanarak etkileşimli olmayan bir şekilde çalıştırabilirsiniz.

## Başlamadan önce

Sağlanan örnek uygulamaların herhangi birini çalıştırmadan önce, uygulamaların bir sunucuya bağlanabilmesi için ileti alışverişi sunucusu ortamını ilk kez ayarlamanız gerekir. Bkz. "[İleti alışverişi sunucusu ortamının ayarlanması](#)" sayfa 585.

## Yordam

.NET örnek uygulamasını çalıştırmak için aşağıdaki adımları tamamlayın:

**İpucu:** Örnek bir uygulama çalıştırırken tip mi? Bir dahaki sefere ne yapacağınıza dair yardım almak için.

1. Örnek uygulamayı çalıştırmak istediğiniz kipi seçin.

Advanced ya da Simple yazın.

2. Sorularına cevap ver.

Soru sonundaki ayraçlarda gösterilen varsayılan değeri seçmek için Enter tuşuna basın. Farklı bir değeri seçmek için, uygun değeri yazın ve Enter tuşuna basın.

Burada bir örnek soru var:

```
Enter connection type [wpm]:
```

Bu durumda, varsayılan değeri wpm ' dir (bir WebSphere Application Server service integration bus bağlantısıdır).

## Sonuçlar

Örnek uygulamaları çalıştırdığınızda, yanıt dosyaları otomatik olarak geçerli çalışma dizininde oluşturulur. Yanıt dosyası adları `connection_type-sample_type.isp` biçiminde olur; örneğin, `wpm-producer.isp`. Gerekliyse, örnek uygulamayı aynı seçeneklerle yeniden çalıştırmak için, seçenekleri yeniden girmeniz gerekmemesi için, oluşturulan yanıt kütüğünü kullanabilirsiniz.

## İlgili görevler

[.NET örnek uygulamalarının oluşturulması](#)

Örnek bir .NET uygulaması oluşturduğunuzda, seçilen örneğinizin yürütülebilir bir sürümü oluşturulur.

[Kendi uygulamalarınızı oluşturma](#)

Örnek uygulamaları oluştururken kendi uygulamalarınızı oluşturun.

## .NET örnek uygulamalarının oluşturulması

Örnek bir .NET uygulaması oluşturduğunuzda, seçilen örneğinizin yürütülebilir bir sürümü oluşturulur.

### Başlamadan önce

Uygun derleyiciyi kurun. Bu görev, sisteminizde Microsoft Visual Studio 2012 kurulu olduğunu ve bunu kullanmaya alışkın olduğunuzu varsayar.

### Yordam

Bir .NET örnek uygulaması oluşturmak için aşağıdaki adımları tamamlayın:

1. .NET örnekleriyle birlikte sağlanan Samples.sln çözüm dosyasını tıklatın.
2. Right-click the solution Örnekler in the Solution Explorer window and select **Oluşturma Çözümü**.

### Sonuçlar

Seçtiğiniz yapılandırmaya bağlı olarak, örneğin bin/Debug ya da bin/Release örneğinin uygun alt klasöründe yürütülebilir bir program yaratılır. This program has the same name as the folder, with a suffix of CS. Örneğin, ileti üreticisi örnek uygulamasının C# sürümünü oluşturuyorsanız, SampleProducerCS.exe klasörü SampleProducer klasöründe yaratılır.

#### İlgili görevler

[.NET örnek uygulamalarının çalıştırılması](#)

.NET örnek uygulamalarını, basit ya da gelişmiş kipte etkileşimli olarak ya da otomatik olarak oluşturulan ya da özelleştirilmiş yanıt dosyaları kullanarak etkileşimli olmayan bir şekilde çalıştırabilirsiniz.

[Kendi uygulamalarınızı oluşturma](#)

Örnek uygulamaları oluştururken kendi uygulamalarınızı oluşturun.

[“Kendi uygulamalarınızı oluşturma” sayfa 596](#)

Örnek uygulamaları oluştururken kendi uygulamalarınızı oluşturun.

## Kendi uygulamalarınızı oluşturma

Örnek uygulamaları oluştururken kendi uygulamalarınızı oluşturun.

### Başlamadan önce

Uygun derleyiciyi kurun. Bu görev, sisteminizde Microsoft Visual Studio 2012 kurulu olduğunu ve bunu kullanmaya alışkın olduğunuzu varsayar.

### Yordam

- [“.NET örnek uygulamalarının oluşturulması” sayfa 596](#) uygulamasında açıklandığı şekilde .NET uygulamanızı oluşturun.

Kendi uygulamalarınızı nasıl oluşturabilmeye ilişkin ek bilgi için, her bir örnek uygulama için sağlanan makefiles kullanın.

**İpucu:** Bir hata durumunda sorun tanısına yardımcı olmak için, içerilen simgelerle uygulamaları derlemenin yararlı olduğunu bulabilirsiniz.

#### İlgili görevler

[.NET örnek uygulamalarının çalıştırılması](#)

.NET örnek uygulamalarını, basit ya da gelişmiş kipte etkileşimli olarak ya da otomatik olarak oluşturulan ya da özelleştirilmiş yanıt dosyaları kullanarak etkileşimli olmayan bir şekilde çalıştırabilirsiniz.

[.NET örnek uygulamalarının oluşturulması](#)

Örnek bir .NET uygulaması oluşturduğunuzda, seçilen örneğinizin yürütülebilir bir sürümü oluşturulur.

## XMS uygulamaları yazılıyor

The topics in this section provides information to help you when writing XMS applications in general.

### Bu görev hakkında

Bu bölüm, XMS uygulamalarını yazmak için genel kavramlar içerir. XMS .NET uygulamaları yaratılmasına ilişkin bilgiler için ayrıca bkz. [“XMS .NET uygulamaları yazılıyor” sayfa 615](#) .

Bu bölüm aşağıdaki konuları içerir:

- [“Threading modeli” sayfa 598](#)
- [“ConnectionFactories ve Connection nesnelere” sayfa 598](#)
- [“Oturumlar” sayfa 600](#)
- [“Hedefler” sayfa 603](#)
- [“İleti üreticileri” sayfa 606](#)
- [“İleti tüketicileri” sayfa 606](#)
- [“Kuyruk tarayıcıları” sayfa 610](#)
- [“İstekçi” sayfa 610](#)
- [“Nesne silme” sayfa 611](#)
- [“XMS temel tipleri” sayfa 612](#)
- [“Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi” sayfa 612](#)
- [“Yineleyiciler” sayfa 614](#)
- [“Kodlanmış karakter takımı tanıtıcıları” sayfa 615](#)
- [“XMS hatası ve kural dışı durum kodları” sayfa 615](#)
- [“Kendi uygulamalarınızı oluşturma” sayfa 596](#)

Windows

V 9.1.5

### IBM MQ XMS .NET proje şablonunu kullanma

IBM MQ 9.1.5' tan, IBM MQ XMS .NET istemcisi, XMS .NET Core uygulamalarınızı geliştirmenize yardımcı olacak bir proje şablonunu kullanma yeteneği sunar.

### Başlamadan önce

Sisteminizde Microsoft Visual Studio 2017ya da daha sonraki bir sürüm ve .NET Core 2.1 bulunmalıdır.

XMS .NET şablonunu şu şablondan kopyalamanız gerekir:

```
&MQ_INSTALL_ROOT&\tools\dotnet\samples\cs\core\xms\ProjectTemplates\IBMXMS.NETClientApp.zip
```

dizini

```
&USER_HOME_DIRECTORY&\Documents\&Visual_Studio_Version&\Templates\ProjectTemplates
```

dizin, burada:

- &MQ\_INSTALL\_ROOT , kurulumunuzun kök dizinidir.
- &USER\_HOME\_DIRECTORY , ana dizininidir.

Şablonu almak için Microsoft Visual Studio ' u durdurmanız ve yeniden başlatmanız gerekir.

## Bu görev hakkında

XMS .NET proje şablonu, uygulamalarınızı geliştirmenize yardımcı olmak için kullanabileceğiniz bazı ortak kodları içerir. With the in-built code, you can connect to the IBM MQ queue manager, and perform a put or a get operation by simply modifying the properties in the in-built code.

## Yordam

1. Microsoft Visual Studio uygulamasını açın.
2. **Dosya**'yı tıklatın, ardından **Yeni** 'yi ve ardından **Proje**' yi tıklatın.
3. *Create a new project window*(Yeni proje penceresi yarat) alanında IBM XMS .NET Client App (.NET Core) seçeneğini belirleyin ve **Next**düğmesini tıklatın.
4. *Yeni projenizi yapılandır* penceresinde, isterseniz projenizin *Proje adı* 'yı değiştirin ve XMS .NET projesini yaratmak için **Yarat** ' ı tıklatın.  
XMSDotnetApp.cs , proje dosyasıyla birlikte oluşturulan dosyadır. Bu dosya, kuyruk yöneticisine bağlanan kodu içerir ve gönderme ve alma işlemi gerçekleştirir.  
Bağlantı özellikleri varsayılan değerlere ayarlanır:
  - WMQ\_CONNECTION\_NAME\_LIST, *localhost (1414)*olarak ayarlandı
  - XMSC.WMQ\_CHANNEL , *DOTNET.SVRCONN*olarak ayarlandı.Kuyruk *Q1*olarak ayarlıdır ve bu özellikleri uygun şekilde değiştirebilirsiniz.
5. Uygulamayı derleyin ve çalıştırın.

## İlgili kavramlar

[IBM MQ bileşenleri ve özellikleri](#)

[.NET uygulama yürütme ortamı- Windows yalnızca](#)

## Threading modeli

Genel kurallar, çok iş parçacıklı bir uygulamanın XMS nesnelere nasıl kullanabileceğini yönetir.

- Farklı iş parçacıklarında yalnızca, aşağıdaki tiplerin nesnelere koşut zamanlı olarak kullanılabilir:
  - ConnectionFactory
  - Bağlantı
  - ConnectionMetaVerileri
  - Hedef
- Bir Oturum nesnesi tek bir kerede tek bir iş parçacığıda kullanılabilir.

Bu kuralların kural dışı durumları, [IBM Message Service Client for .NET başvurusu](#) içindeki yöntemlerin arabirim tanımlamalarında "İş parçacığı bağlamı" etiketli girişlerle gösterilir.

## ConnectionFactoryies ve Connection nesnelere

ConnectionFactory nesnesi, uygulamanın Connection nesnesi yaratmak için kullandığı bir şablon sağlar. Uygulama, bir Oturum nesnesi yaratmak için Bağlantı nesnesini kullanır.

For .NET, XMS application first uses an XMSFactoryFactory object to get a reference to a ConnectionFactory object that is appropriate to the required type of protocol. Bu ConnectionFactory nesnesi, yalnızca bu protokol tipi için bağlantı üretebilir.

Bir XMS uygulaması birden çok bağlantı yaratabilir ve birden çok iş parçacıklı bir uygulama, birden çok iş parçacığının koşut zamanlı olarak tek bir Connection nesnesini kullanabilir. Bağlantı nesnesi, bir uygulama ile ileti sistemi sunucusu arasındaki bir iletişim bağlantısını sarsalıyor.

Bir bağlantı birkaç amaca hizmet eder:

- Bir uygulama bağlantı yarattığında, uygulamanın doğrulanabileceği bir uygulama olabilir.

- Bir uygulama, benzersiz bir istemci tanıtıcısını bir bağlantıyla ilişkilendirebilir. Yayınlama/abone olma etki alanında dayanıklı abonelikleri desteklemek için istemci tanıtıcısı kullanılır. İstemci tanıtıcısı şu iki şekilde ayarlanabilir:

Bir bağlantı istemcisi tanıtıcısı atanmanın tercih edilen yolu, özellikleri kullanarak istemciye özgü bir ConnectionFactory nesnesinde yapılandırılacak ve bunu yarattığı bağlantıya saydam bir şekilde atayabilmekte.

İstemci tanıtıcısını atanmanın diğer bir yolu, Connection nesnesinde belirlenen sağlayıcıya özgü bir değer kullanmandır. Bu değer, yönetsel olarak yapılandırılmış olan tanıtıcıyı geçersiz kılmaz. Yönetsel olarak belirtilmiş bir tanıtıcı varsa, bunu sağlayıcıya özgü bir değer ile geçersiz kılma girişimi bir kural dışı durumun yayınlanmasına neden olur. Bir uygulama belirtik olarak bir tanıtıcıyı ayarlarsa, bağlantı yaratıldıktan hemen sonra ve bağlantıda başka herhangi bir işlem alınmadan önce bunu yapması gerekir; aksi takdirde, kural dışı durum yayınlanır.

XMS uygulaması tipik olarak bir bağlantı, bir ya da daha çok oturum ve ileti üreticileri ve ileti tüketicileri oluşturur.

Bağlantı yaratılması sistem kaynakları açısından göreceli olarak pahalı olduğundan, iletişim bağlantısı kurulmasını içerdiği için, uygulamanın kimlik doğrulanmasını da gerektirebilir.

## Bağlantı başlatıldı ve durduruldu kipi

Bir bağlantı, başlatılmış ya da durdurulmuş kipte çalışabilir.

Bir uygulama bağlantı yarattığında, bağlantı durdurulmuş kipte olur. Bağlantı durdurulduğunda, uygulama oturumları başlatabilir, iletileri gönderebilir, ancak zamanuyumlu ya da zamanuyumsuz olarak alamazlar.

Bir uygulama, Start Connection yöntemini çağırarak bağlantı başlatabilir. Bağlantı başlatıldığında, uygulama iletileri gönderebilir ve alabilir. Daha sonra, uygulama durdurulabilir ve Bağlantıyı Durdur ve Start Connection yöntemlerini çağırarak bağlantıyı 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 nesnelere siler. Hangi nesnelere silindiği hakkında daha fazla bilgi için bkz. "[Nesne silme](#)" sayfa 611. Aynı zamanda, XMS oturumlar içinde devam etmekte olan tüm işlemleri geri alır.
- İleti, ileti sistemi sunucusuyla iletişim bağlantısını sona erdirir.
- Bu, bellek ve bağlantı tarafından kullanılan diğer iç kaynakları serbest bırakır.

XMS, bağlantı kapatılmadan önce, bir oturum sırasında kabul edememiş olduğu iletilerin alındığını bildirmez. İletilerin alınmasını kabul etme hakkında daha fazla bilgi için bkz. "[İleti alındı bildirimini](#)" sayfa 601.

## Kural dışı durumları işleme

XMS .NET özel durumlarının tümü System.Exception' dan türetilmiştir. Daha fazla bilgi için bkz "[Error handling in .NET](#)" sayfa 619.

## Hizmet tümleştirme veriyoluna bağlantı

Bir XMS uygulaması, doğrudan TCP/IP bağlantısı kullanarak ya da TCP/IP üzerinden HTTP kullanarak bir WebSphere Application Server hizmet tümleştirme veriyoluna bağlanabilir.

HTTP protokolü, doğrudan TCP/IP bağlantısının olmadığı durumlarda kullanılabilir. Ortak bir durum, iki kuruluşun ileti alışverişi yaptığı gibi bir güvenlik duvarı üzerinden iletişim kurmaktadır. Güvenlik duvarı üzerinden iletişim kurmak için HTTP ' nin kullanılması genellikle *HTTP tünellemesi* olarak adlandırılır. Ancak HTTP üstbilgileri aktarılan veri miktarına önemli ölçüde eklendiğinden ve HTTP iletişim kuralı

TCP/IP ' den daha fazla iletişim akışı gerektirdiğinden, HTTP tünelleme doğrudan TCP/IP bağlantısı kullandığından daha yavaş olur.

Bir TCP/IP bağlantısı yaratmak için, bir uygulama `XMSC_WPM_TARGET_TRANSPORT_CHAIN` özelliği `XMSSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC` olarak ayarlanmış bir bağlantı üreticisine sahip olabilir. Bu, özelliğin varsayılan değeridir. Bağlantı başarıyla yaratıldıysa, bağlantının `XMSC_WPM_CONNECTION_PROTOCOL` özelliği `XMSC_WPM_CP_TCP` olarak ayarlanır.

HTTP kullanan bir bağlantı yaratmak için, bir uygulamanın `XMSC_WPM_TARGET_TRANSPORT_CHAIN` özelliği, bir HTTP iletim kanalını kullanmak üzere yapılandırılmış bir gelen iletim zinciri adı olarak ayarlanmış bir bağlantı üreticisi kullanmalıdır. Bağlantı başarıyla yaratıldıysa, bağlantının `XMSSC_WPM_CONNECTION_PROTOCOL` özelliği `XMSC_WPM_CP_HTTP` olarak ayarlanır. İletim zincirlerinin nasıl yapılandırılacağı hakkında bilgi için, WebSphere Application Server ürün belgelerinde [Configuring transport chains](#) başlıklı konuya bakın.

Bir uygulama, bir önyükleme sunucusuna bağlanırken benzer bir iletişim protokolleri seçimine sahiptir. Bir bağlantı üreticisinin `XMSC_WPM_PROVIDER_UÇ` Noktaları özelliği, önyükleme sunucularının bir ya da daha çok uç noktası adreslerinden oluşan bir dizidir. Her uç adresin önyükleme iletim zinciri bileşeni, HTTP kullanan bir bağlantı için, bir önyükleme sunucusuyla ya da `XMSSC_WPM_BOOTSTRAP_HTTP` ile TCP/IP bağlantısı için `XMSC_WPM_BOOTSTRAP_TCP` olabilir.

## Oturumlar

Oturum, ileti göndermek ve almak için tek bir iş parçacıklı bağlamdır.

Bir uygulama, ileti, ileti üreticileri, ileti tüketicileri, kuyruk tarayıcıları ve geçici hedefler yaratmak için bir oturumu kullanabilir. Bir uygulama yerel hareketleri çalıştırmak için de oturum kullanabilir.

Bir uygulama, her oturumun diğer oturumlardan bağımsız olarak ileti ürettiği ve tükettiği birden çok oturum yaratabiliyor. Ayrı oturumlarda (ya da aynı oturumda bile) iki ileti tüketicinin aynı konuya abone olması durumunda, her biri bu konuda yayınlanan herhangi bir iletinin bir kopyasını alır.

Connection nesnesinden farklı olarak, bir Oturum nesnesi farklı iş parçacıklarında koşut zamanlı olarak kullanılamaz. Bir oturum nesnesinin yalnızca Close Session yöntemi, oturum nesnesinin o sırada kullandığı bir iş parçacıktan başka bir iş parçacığıdan çağrılabilir. Oturum Kapat (Close Session) yöntemi bir oturumu sonlandırır ve oturuma ayrılan sistem kaynaklarını serbest bırakır.

Bir uygulamanın iletileri birden çok iş parçacığıda koşut zamanlı olarak işlemesi gerekiyorsa, uygulamanın her bir iş parçacığıda bir oturum yaratması ve söz konusu iş parçacığın içindeki gönderme ya da alma işlemi için bu oturumu kullanması gerekir.

## Hareket edilen oturumlar

XMS uygulamaları yerel işlemleri çalıştırabilir. *Yerel hareket* , yalnızca, uygulamanın bağlı olduğu kuyruk yöneticisinin ya da hizmet tümleştirme veriyolunun kaynaklarında yapılan değişiklikleri içeren bir işlemdir.

Bu konudaki bilgiler, yalnızca bir uygulama bir IBM MQ kuyruk yöneticisine ya da WebSphere Application Server hizmet bütünleştirme veriyoluna bağlıysa anlamlıdır. Bilgi, bir aracıyla gerçek zamanlı bağlantı için uygun değildir.

Bir uygulamanın, yerel hareketleri çalıştırmak için, önce bir Bağlantı nesnesinin Create Session (Oturum Yarat) yöntemini çağırarak, oturumun hareket ettiği bir değıştirge olarak belirtilerek, aktarım sırasında bir oturum yaratması gerekir. Daha sonra, oturum içinde gönderilen ve alınan tüm iletiler bir işlem sırasına göre gruplandırılır. Bir işlem, uygulama başlattığında ya da gönderdiği iletileri geri alırken, işlem başlatıldığından bu yana aldığı iletiler sona erer.

Bir işlem kesinleştirilmek için, bir uygulama Oturum nesnesinin Kesinleştir yöntemini çağırır. Bir işlem kesinleştirildiğinde, işlem içinde gönderilen tüm iletiler diğer uygulamalara teslim edilebilmesi için kullanılabilir duruma gelir ve işlem içinde alınan tüm iletiler, ileti alışverişi sunucusunun bunları yeniden uygulamaya teslim etme girişiminde bulunmayabilmesi için kabul edilir. Noktadan noktaya iletişim alanında, ileti alışverişi sunucusu alınan iletileri kuyruklarından da kaldırır.

Bir işlemi geri yüklemek için, bir uygulama Oturum nesnesinin Geri Alma yöntemini çağırır. Bir hareket geriye işlendiğinde, işlem içinde gönderilen tüm iletiler ileti alışverişi sunucusu tarafından atılır ve



işlem içinde alınan tüm iletiler, teslim için yeniden kullanılabilir duruma gelir. Noktadan noktaya iletişim alanında, alınan iletiler kuyruklarına geri alınır ve diğer uygulamalar tarafından görülebilir duruma gelir.

Uygulama, hareket eden bir oturum yarattığında ya da Commit ya da Rollback yöntemini çağırdığında otomatik olarak yeni bir işlem başlar. Bu nedenle, hareket eden bir oturumun her zaman etkin bir hareketi vardır.

Bir uygulama hareket edilen bir oturumu kapattığında, örtük bir geri alma gerçekleşir. Bir uygulama bağlantıyı kapattığında, tüm bağlantının hareket ettiği oturumlar için örtük bir geri alma gerçekleşir.

Hareket, hareket eden bir oturumda tamamen içerilidir. Bir hareket oturumlara yayılamaz. Diğer bir deyişle, bir uygulamanın iki ya da daha fazla sayıda dönüştürücü oturumuna ileti gönderip alması ve sonra tüm bu işlemleri tek bir işlem olarak kesinleştirmesi ya da geriye işlemleri mümkün değildir.

### **İlgili kavramlar**

#### **İleti alındı bildirimi**

İletilmeyen her oturumda, uygulamanın aldığı iletilerin nasıl kabul edildiğine karar veren bir alındı bildirimi kipi vardır. Üç alındı bildirimi kipi vardır ve alındı bildirimi kipinin seçimi, uygulamanın tasarımını etkiler.

#### **İleti teslimi**

XMS , ileti tesliminin sürekli ve kalıcı olmayan kiplerini ve iletilerin zamanuyumlu ve zamanuyumlu olarak sunulmasını destekler.

### **İleti alındı bildirimi**

İletilmeyen her oturumda, uygulamanın aldığı iletilerin nasıl kabul edildiğine karar veren 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 bütünleştirme veriyoluna bağlıysa anlamlıdır. Bilgi, bir aracıyla gerçek zamanlı bağlantı için uygun değildir.

XMS , JMS ' nin kullandığı iletilerin alınmasının onaylanmak için aynı mekanizmayı kullanır.

Bir oturum hareket edilmezse, uygulamanın aldığı iletilerin kabul edilme şekli, oturumun alındı bildirimi kipiyle belirlenir. Üç alındı bildirimi kipleri aşağıdaki paragraflarda açıklanmaktadır:

#### **XMSCS\_AUTO\_RASK**

Oturum, uygulama tarafından alınan her iletiyi otomatik olarak kabul eder.

İletiler uygulamaya zamanuyumlu olarak teslim ediliyorsa, bir Receive çağrısı her başarıyla tamamlandığında, oturum bir iletinin alındığını kabul eder.

Uygulama başarıyla bir ileti alırsa, ancak hata onayının oluşmasını önlüyorsa, ileti teslim için kullanılabilir duruma gelir. Bu nedenle, uygulamanın yeniden teslim edilen bir iletiyi işleyebilmesi gerekir.

#### **XMSC\_DUPS\_OK\_ALIND**

Bu oturum, uygulama tarafından alınan iletilerin seçim süresinde kabul ettiği kabul eder.

Bu alındı bildirimi kipinin kullanılması, oturumun yapması gereken iş miktarını azaltır, ancak ileti onayını önleyen bir hata, birden çok iletinin teslim edilmesi için kullanılabilir duruma gelmesinden kaynaklanabilir. Bu nedenle, uygulamanın yeniden teslim edilen iletileri işleyebilmesi gerekir.

#### **XMSC\_CLIENT\_ALIND**

Uygulama, İleti sınıfının Acknowy yöntemini çağırarak aldığı iletileri kabul eder.

Uygulama, her iletinin alınmasını tek tek onaylayabilir ya da bir ileti kümesi alabilir ve Acknowy yöntemini yalnızca aldığı son ileti için çağırabilir. Yöntemin çağırdığı son çağrıdan bu yana Acknowy yöntemi çağrıldığında alınan tüm iletiler onaylanır.

Bu onay kiplerinden herhangi biriyle birlikte, bir uygulama Oturum sınıfının Recon yöntemini çağırarak, bir oturumda iletilerin teslim edilmesini durdurabilir ve yeniden başlatabilir. Alındı bildirimi önceden onaylanmamış 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 alanında, özgün iletilerin bazıları başka bir uygulama tarafından tüketilmiş olabilir.

Bir uygulama, iletinin JMSReverulamış üstbilgi alanının içeriğini inceleyerek bir iletinin yeniden teslim edilip edilmediğini belirleyebilir. Bu işlem, Message sınıfının JMSReverlemiş yöntemini çağırarak bunu yapar.

### **İlgili kavramlar**

Hareket edilen oturumlar

XMS uygulamaları yerel işlemleri çalıştırabilir. *Yerel hareket* , yalnızca, uygulamanın bağlı olduğu kuyruk yöneticisinin ya da hizmet tümleştirme veriyolunun kaynaklarında yapılan değişiklikleri içeren bir işlemdir.

İleti teslimi

XMS , ileti tesliminin sürekli ve kalıcı olmayan kiplerini ve iletilerin zamanuyumlu ve zamanuyumlu olarak sunulmasını destekler.

### **İleti teslimi**

XMS , ileti tesliminin sürekli ve kalıcı olmayan kiplerini ve iletilerin zamanuyumlu ve zamanuyumlu olarak sunulmasını destekler.

### **İleti teslim kipi**

XMS , ileti tesliminin iki kipini destekler:

#### **Kalıcı**

Kalıcı iletiler bir kez teslim edilir. Bir ileti alışverişi sunucusu, hata durumunda bile iletilerin geçiş sırasında kaybolmadığından emin olmak için iletileri günlüğe kaydetme gibi özel önlemleri alır.

#### **Kalıcı olmayan**

Kalıcı olmayan iletiler bir kereden fazla teslim edilir. Kalıcı olmayan iletiler, bir hata durumunda geçiş sırasında kaybedebileceğinden, kalıcı iletilerden daha az güvenilirdir.

Teslim kipinin seçimi, güvenilirlik ve başarımlar arasında bir değiş tokardır. Kalıcı olmayan iletiler genellikle kalıcı iletilerden daha hızlı taşınır.

### **Zamanuyumsuz ileti teslimi**

XMS , bir oturuma ilişkin tüm zamanuyumsuz ileti teslimlerini işlemek için bir iş parçacığı kullanır. Başka bir deyişle, bir kerede tek bir ileti dinleyicisi işlevi ya da bir `onMessage()` yöntemi çalıştırılabilir.

Bir oturumda birden çok ileti tüketicisi iletileri zamanuyumsuz olarak alıyorsa ve bir ileti dinleyici işlevi ya da `onMessage()` yöntemi ileti tüketicisi için bir ileti gönderiyorsa, aynı iletiyi bekleyen diğer ileti tüketicileri beklemeye devam etmelidir. Oturuma teslim edilmeyi bekleyen diğer iletiler de beklemeye devam etmelidir.

Bir uygulama iletileri eşzamanlı olarak teslim etmeyi gerektiriyorsa, birden çok oturum yaratın; böylece, XMS zamanuyumsuz ileti teslimi işlemek için birden çok iş parçacığı kullanır. Bu şekilde, birden çok ileti dinleyici işlevi ya da `onMessage()` yöntemi koşut zamanlı olarak çalışabilir.

Tüketiciye bir ileti dinleyicisi atayarak, oturum zamanuyumsuz olarak yapılmaz. Bir oturum, `Connection.Start` yöntemi çağrıldığında zamanuyumsuz hale gelir. `Connection.Start` yöntemi çağrılıncaya kadar tüm zamanuyumlu çağrılara izin verilir. `Message delivery to consumers start when the Connection.Start is called.`

Bir tüketici ya da üretici yaratma gibi zamanuyumlu çağrılar zamanuyumsuz bir oturumda yapılmalıdır; bu çağrılar `Connection.Stop` çağrılmalıdır. İletilerin teslimi başlatmak için `Connection.Start` yöntemi çağrılarak bir oturuma devam edilebilir. Bu durum için tek kural dışı durum, geri bildirme işlevine ileti gönderen iş parçacığıysa, Oturum iletileri teslim iş parçacığıdır. Bu iş parçacığı, iletileri geri çağırma işlevinde oturum (kapatma çağrısı dışında) herhangi bir çağrıyı gerçekleştirmez.

**Not:** Yönetilmeyen kipte, geri çağrılan işlev içindeki MQDISC çağrısı IBM MQ .NET istemcisi tarafından desteklenmez. Bu nedenle, istemci uygulaması Zamanuyumsuz alma kipinde `MessageListener` geri

çağırması içinde oturum Yaratamaz ya da kapatamıyor. Create and dispose the session outside of the MessageListener method.

## Zamanuyumlu ileti teslimi

Uygulama, MessageConsumer nesnelere ilişkin Alma yöntemlerini kullanıyorsa, iletiler bir uygulamaya zamanuyumlu olarak teslim edilir.

Alma yöntemlerinin kullanılması, bir ileti için belirli bir süre bekleyebilir ya da süresiz olarak bekleyebilir. Diğer bir seçenek olarak, bir uygulama bir iletiyi beklemek istemezse, Bekleme yöntemini kullanmadan Al 'ı kullanabilir.

### İlgili kavramlar

Hareket edilen oturumlar

XMS uygulamaları yerel işlemleri çalıştırabilir. *Yerel hareket* , yalnızca, uygulamanın bağlı olduğu kuyruk yöneticisinin ya da hizmet tümleştirme veriyolunun kaynaklarında yapılan değişiklikleri içeren bir işlemdir.

İleti alındı bildirimi

İletilmeyen her oturumda, uygulamanın aldığı iletilerin nasıl kabul edildiğine karar veren bir alındı bildirimi kipi vardır. Üç alındı bildirimi kipi vardır ve alındı bildirimi kipinin seçimi, uygulamanın tasarımını etkiler.

## Hedefler

Bir XMS uygulaması, gönderilmekte olan iletilerin hedefini ve alınmakta olan iletilerin kaynağını belirtmek için bir Hedef nesne kullanır.

Bir XMS uygulaması yürütme sırasında bir hedef nesne yaratabilir ya da yönetilen nesnelere havuzundan önceden tanımlanmış bir hedef alabilir.

Bir ConnectionFactory ile olduğu gibi, XMS uygulamasının bir hedef belirlemesi için en esnek yol, bunu yönetilen bir nesne olarak tanımlamasıdır. Bu yaklaşımın kullanılması, C, C++ ve .NET dillerinde yazılan uygulamaların ve Java'ın hedef tanımlarını paylaşabileceği şekilde ifade eder. Yönetilen Hedef nesnelere özellikleri, herhangi bir kod değiştirilmeden değiştirilebilir.

.NET uygulamaları için, CreateTopic ya da CreateQueue yöntemini kullanarak bir hedef yaratıyorsunuz. Bu iki yöntem, .NET API 'da hem ISSession hem de XMSFactoryFactory nesnelere kullanılabilir. Daha fazla bilgi için bkz. [“.NET içindeki hedefler” sayfa 617](#) ve [../com.ibm.mq.ref.dev.doc/sapidest.dita#sapidest](#).

### Konu URL tipi kaynak tanıtıcıları

Konu tekstili kaynak tanıtıcısı (URI), konunun adını belirtir; bir ya da daha çok özellik de belirtebilir.

Bir konuya ilişkin URI, şu sıra ile başlar: //, onu izleyen konu adı ve (isteğe bağlı) kalan konu özelliklerini ayarlayan ad-değer çiftleri listesi. Bir konu adı boş olamaz.

Burada, .NET kodunun bir parçasında bir örnek vardır:

```
topic = session.CreateTopic("topic://Sport/Football/Results?multicast=7");
```

Bir URI içinde kullanabileceğiniz ad ve geçerli değerler de içinde olmak üzere, bir konunun özellikleri hakkında daha fazla bilgi için [Hedef Özellikleri](#) konusuna bakın.

Abonelikte kullanmak üzere bir konu URI 'si belirtilirken joker karakterler kullanılabilir. Bu genel arama karakterlerinin sözdizimi, bağlantı tipi ve aracı sürümüne bağlıdır; aşağıdaki seçenekler kullanılabilir:

- Karakter düzeyi genel arama karakteri biçimiyle IBM WebSphere MQ 7.0 kuyruk yöneticisi
- IBM WebSphere MQ 7.0 queue manager with Topic level wildcard format
- WebSphere Application Server hizmet bütünleştirme veriyolu

## Karakter düzeyi genel arama karakteri biçimiyle IBM WebSphere MQ 7.0 kuyruk yöneticisi

Karakter düzeyi genel arama karakteriyle IBM WebSphere MQ 7.0 kuyruk yöneticisi, aşağıdaki genel arama karakterlerini kullanır:

- \* 0 ya da daha çok karakter için
- ? 1 karakter için
- % Bir çıkış karakteri için

Çizelge 81 sayfa 604 , bu genel arama karakteri şemasının nasıl kullanılacağını gösteren bazı örnekler verir.

Çizelge 81. IBM WebSphere MQ 7.0 kuyruk yöneticisi için karakter düzeyi genel arama şemasını kullanan örnek URI 'ler		
Bir örnek Kaynak Tanıtıcısı	Eşleşir	Örnekler
"topic://Sport *Results"	"Spor" ile başlayan ve "Sonuçlar" ile biten tüm konular	"topic://SportsResults" ve "topic://Sport/Hockey/National/Div3/Results"
" topic://Sport?Sonuçlar "	"Spor" ile başlayan ve ardından tek bir karakter olan tüm konular, ardından "Sonuçlar"	"topic://SportsResults" ve "topic://SportXResults"
"topic://Sport/ * ball*/Div? / Results*/???"	Konular	"topic://Sport/Football/Div1/Results/2002/Nov" ve "topic://Sport/Netball/National/Div3/Results/02/Jan"

## IBM WebSphere MQ 7.0 queue manager with Topic level wild card format

IBM WebSphere MQ 7.0 queue manager with Topic level wildcard format uses the following wildcard characters:

- # birden çok düzeye eşleştirmek için
- + tek bir düzeye eşleştirmek için

Çizelge 82 sayfa 604 , bu genel arama karakteri şemasının nasıl kullanılacağını gösteren bazı örnekler verir.

Çizelge 82. IBM WebSphere MQ 7.0 kuyruk yöneticisi için konu düzeyi genel arama şemasını kullanan örnek URI 'ler		
Bir örnek Kaynak Tanıtıcısı	Eşleşir	Örnekler
"topic://Sport/ + / Results"	Spor ile Sonuçlar arasında tek bir sıradüzensel düzey adı olan tüm konular	"topic://Sport/Football/Results" ve "topic://Sport/Ju-Jitsu/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/#"	"Sport/Football/" ile başlayan tüm konular	"topic://Sport/Football/Results" ve "topic://Sport/Football/TeamNews/Signings/Managery"

## WebSphere Application Server hizmet bütünleştirme veriyolu

WebSphere Application Server hizmet tümleştirme veriyolu şu genel arama karakterlerini kullanır:

- \* sıradüzendeki bir düzeydeki tüm karakterleri eşleştirmek için
- // 0 ya da daha fazla düzeye eşleştirmek
- // . 0 ya da daha fazla düzeyle eşleştirmek için (Konu ifadesinin sonunda)

Çizelge 83 sayfa 605 , bu genel arama karakteri şemasının nasıl kullanılacağını gösteren bazı örnekler verir.

Çizelge 83. WebSphere Application Server hizmet tümleştirme veriyolu için genel arama şeması kullanan örnek URI 'ler		
Bir örnek kaynak Tanıtıcısı	Eşleşir	Örnekler
"topic://Sport/ * ball/ Results"	Spor ve Sonuçlar arasında "top" ile biten tek bir sıradüzensel 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//."	"Sport/Football/" ile başlayan tüm konular	"topic://Sport/Football/Results" ve "topic://Sport/Football/TeamNews/ Signings/Managery"
"topic://Sport/ * ball// Results//."	Konular	"topic://Sport/Football/Results" ve "topic://Sport/Netball/National/Div3/ Results/2002/November"

### İlgili kavramlar

Kuyruk tek biçimli kaynak tanıtıcıları

Kuyruğun URI 'si, kuyruğun adını belirtir; ayrıca, kuyruğun bir ya da daha fazla özelliğini de belirtebilir.

Geçici hedefler

XMS uygulamaları geçici hedefler oluşturabilir ve kullanabilir.

### Kuyruk tek biçimli kaynak tanıtıcıları

Kuyruğun URI 'si, kuyruğun adını belirtir; ayrıca, kuyruğun bir ya da daha fazla özelliğini de belirtebilir.

Bir kuyruğa ilişkin URI, queue : //sırası ile başlar, ardından kuyruğun adını belirtir; ayrıca, kalan kuyruk özelliklerini ayarlayan ad-değer çiftlerinin bir listesini de içerebilir.

IBM MQ kuyrukları için ( WebSphere Application Server varsayılan ileti alışverişi sağlayıcısı kuyrukları için değil), kuyruğun bulunduğu kuyruk yöneticisi kuyruktan önce, kuyruk yöneticisi adını kuyruk adından ayırıyor ya da kuyruktan ayırıyor olabilir.

If a queue manager is specified, then it must be the one to which XMS is directly connected for the connection using this queue, or it must be accessible from this queue. Uzak kuyruk yöneticileri yalnızca kuyruktan ileti almak için, iletileri kuyruklara koymak için değil, kuyruklardan almak için desteklenir. Tüm ayrıntılar için, IBM MQ kuyruk yöneticisi belgelerine bakın.

Kuyruk yöneticisi belirtilmediyse, ek/ayırıcı isteğe bağlıdır ve varlığın varlığı ya da yokluğu, kuyruğun tanımı üzerinde hiçbir fark göstermez.

Aşağıdaki kuyruk tanımları, QM\_A adlı bir kuyruk yöneticisindeki QB adlı bir IBM MQ kuyruğu için, XMS ' un doğrudan bağlı olduğu bir kuyruk yöneticisine eşdeğerdir:

```
queue://QB
queue:///QB
queue://QM_A/QB
```

### İlgili kavramlar

Konu URL tipi kaynak tanıtıcıları

Konu tekstili kaynak tanıtıcısı (URI), konunun adını belirtir; bir ya da daha çok özellik de belirtebilir.

## Geçici hedefler

XMS uygulamaları geçici hedefler oluşturabilir ve kullanabilir.

## **Geçici hedefler**

XMS uygulamaları geçici hedefler oluşturabilir ve kullanabilir.

Uygulama, genellikle istek iletilerine yanıt almak için geçici bir hedef kullanır. Bir istek iletilerinin yanıtının gönderileceği hedefi belirtmek için, bir uygulama istek iletilerini temsil eden ileti nesnesinin Set JMSReplyTo yöntemini çağırır. Çağrıda belirtilen hedef geçici bir hedef olabilir.

Geçici bir hedef yaratmak için bir oturum kullanılsa da, geçici bir hedefin kapsamı gerçekte, oturumu yaratmak için kullanılan bağlantıdır. Bağlantı oturumlarından herhangi biri geçici hedef için ileti üreticileri ve ileti tüketicileri oluşturabilir. Geçici hedef, açık bir şekilde silininceye ya da bağlantı sona erinceye kadar kalır; hangisi önce olursa.

Bir uygulama geçici bir kuyruk yarattığında, uygulamanın bağlı olduğu ileti sistemi sunucusunda bir kuyruk yaratılır. Uygulama bir kuyruk yöneticisine bağlıysa, adı `XMLC_WMQ_TEMPORARY_MODEL` özelliği tarafından belirtilen model kuyruğundan dinamik bir kuyruk yaratılır ve dinamik kuyruk adını oluşturmak için kullanılan örnek `XMLC_WMQ_TEMP_Q_PREFIX` özelliği tarafından belirtilir. Uygulama bir hizmet bütünleştirme veriyoluna bağlıysa, veriyolunda geçici bir kuyruk yaratılır ve geçici kuyruğun adını oluşturmak için kullanılan örnek `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 örnek `XMSC_WPM_TEMP_TOPIC_PREFIX` özelliği tarafından belirtilir.

## **İlgili kavramlar**

Konu URL tipi kaynak tanıtıcıları

Konu tekstili kaynak tanıtıcısı (URI), konunun adını belirtir; bir ya da daha çok özellik de belirtebilir.

Kuyruk tek biçimli kaynak tanıtıcıları

Kuyruğun URI 'si, kuyruğun adını belirtir; ayrıca, kuyruğun bir ya da daha fazla özelliğini de belirtebilir.

## **İleti üreticileri**

XMSiçinde, bir ileti üreticisi geçerli bir hedefle ya da ilişkili bir hedef olmadan yaratılabilir. Boş değerli bir hedef içeren ileti üreticisi yaratırken, ileti gönderirken geçerli bir hedef belirtilmelidir.

## **İlişkilendirilmiş hedefi olan ileti üreticileri**

Bu senaryoda, ileti üreticisi geçerli bir hedef kullanılarak yaratılır. Gönderme işlemi sırasında hedef belirtilmemesi gerekir.

## **İlişkilendirilmiş hedefi olmayan ileti üreticileri**

XMS .NET' ta, boş değerli bir hedefle bir ileti üreticisi yaratılabilir.

.NET API kullanılırken ilişkili hedefi olmayan bir ileti üreticisi yaratmak için, NULL (boş) ISUse nesnesinin `CreateProducer()` yöntemine (örneğin, `session.CreateProducer(null)`) parametre olarak NULL (boş) geçirilmelidir. Ancak, ileti gönderildiğinde geçerli bir hedef belirtilmelidir.

## **İleti tüketicileri**

İleti tüketicileri, dayanıklı ve dayanıklı olmayan aboneler, zamanuyumlu ve zamanuyumsuz ileti tüketicileri olarak sınıflandırılabilir.

## **Sürekli aboneler**

Kalıcı abone, abone etkin değilken yayınlanan iletiler de içinde olmak üzere, bir konuda yayınlanan tüm iletileri alan bir ileti tüketicisi 'dir.

Bu konudaki bilgiler, yalnızca bir uygulama bir IBM MQ kuyruk yöneticisine ya da WebSphere Application Server hizmet bütünleştirme veriyoluna bağlıysa anlamlıdır. Bilgi, bir aracıyla gerçek zamanlı bağlantı için uygun değildir.

Bir konuya ilişkin dayanıklı bir abone yaratmak için, uygulama, bir Oturum nesnesinin Durable Subscriber yöntemini çağıran, kalıcı aboneliği ve konuyu temsil eden bir Hedef nesneyi tanımlayan bir ad olarak belirtilen bir Oturum nesnesinin Create Durable Subscriber yöntemini çağırır. Uygulama, bir ileti seçici ile ya da bir ileti seçicisiz dayanıklı bir abone yaratabilir ve kalıcı abonenin kendi bağlantısı tarafından yayınlanan iletileri alıp almayacağını belirtebilir.

Kalıcı bir abone yaratmak için kullanılan oturumun ilişkili bir istemci tanıtıcısı olmalıdır. İstemci tanıtıcısı, oturumu yaratmak için kullanılan bağlantıyla ilişkili ile ayıdır; “ConnectionFactoryies ve Connection nesnelere” sayfa 598 içinde açıklandığı şekilde belirtilir.

Kalıcı aboneliği tanımlayan ad, istemci tanıtıcısı içinde benzersiz olmalıdır; dolayısıyla, istemci tanıtıcısı, sürekli aboneliğin tam, benzersiz tanıtıcısının bir parçası olmalıdır. İleti alışverişi sunucusu, kalıcı aboneliğin kaydını tutar ve bu konuda yayınlanan tüm iletilerin, kalıcı abone tarafından kabul edilinceye ya da süre bitimine kadar saklanmalarını sağlar.

İleti alışverişi sunucusu, dayanıklı abone kapandıktan sonra bile sürekli abonelik kaydını tutmaya devam eder. Daha önce yaratılmış bir kalıcı aboneliği yeniden kullanmak için, bir uygulama aynı abonelik adını belirten bir kalıcı abone oluşturmalı ve sürekli abonelik ile ilişkili olan aynı istemci tanıtıcısına sahip bir oturum oluşturmalıdır. Belirli bir kalıcı abonelik için zaman zaman yalnızca bir oturum dayanıklı bir aboneye sahip olabilir.

Kalıcı bir aboneliğin kapsamı, aboneliğin kaydını koruyan ileti alışverişi sunucusudur. Farklı ileti sistemi sunucularına bağlı iki uygulama, aynı abonelik adını ve istemci tanıtıcısını kullanarak her bir kalıcı abone yaratır; tamamen bağımsız iki kalıcı abonelik yaratılır.

Bir kalıcı aboneliği silmek için, uygulama, kalıcı aboneliği tanımlayan ad olarak belirterek, bir Oturum nesnesinin Unsubscription yöntemini çağırır. Oturumla ilişkili istemci tanıtıcısı, sürekli abonelik ile ilişkili olan istemci tanıtıcısı olmalıdır. İleti alışverişi sunucusu, sürdürmekte olduğu dayanıklı aboneliğin kaydını siler ve kalıcı aboneye başka ileti göndermez.

Var olan bir aboneliği değiştirmek için, bir uygulama aynı abonelik adını ve istemci tanıtıcısını kullanarak, ancak farklı bir konu ya da ileti seçici (ya da her ikisi) belirtilerek kalıcı bir abone yaratabilir. Sürekli aboneliği değiştirmek, aboneliğin silinmesine ve yeni bir aboneliğin oluşturulabilmesiyle eşdeğerdir.

For an application that connects to IBM WebSphere MQ 7.0 or later queue manager, XMS manages the subscriber queues. Bu nedenle, uygulamanın bir abone kuyruğu belirtmesi gerekli değildir. XMS , belirtilirse, abone kuyruğunu yoksayacak.

Kalıcı bir abonelik için abone kuyruğunu değiştiremediğinizi unutmayın. Abone kuyruğunu değiştirmenin tek yolu, aboneliği silmeniz ve yeni bir abonelik yaratmasıdır.

Bir hizmet bütünleştirme veriyoluna bağlanan bir uygulama için, her dayanıklı abonenin belirlenmiş bir kalıcı aboneliği ana dizini olmalıdır. Aynı bağlantıyı kullanan tüm kalıcı aboneler için dayanıklı aboneliği ana dizini belirtmek için, bağlantıyı oluşturmak için kullanılan ConnectionFactory nesnesinin XMLC\_WPM\_DUR\_SUB\_HOME özelliğini ayarlayın. Tek bir konuya ilişkin dayanıklı aboneliği ana dizini belirtmek için, konuyu temsil eden Hedef nesnenin XMLC\_WPM\_DUR\_XX\_ENCODE\_CASE\_ONE sub\_home özelliğini ayarlayın. Bir uygulamanın bağlantıyı kullanan dayanıklı bir abone oluşturabilmesi için, bağlantı için sürekli abonelik ana dizini belirtilmelidir. Bir hedef için belirtilen herhangi bir değer, bağlantı için belirtilen değeri geçersiz kılar.

### ***Dayanıklı olmayan aboneler***

Dayanıklı olmayan bir abone, yalnızca abone etkin durumdayken yayınlanan iletileri alan bir ileti tüketicisi olur. Abone etkinlik dışı durumdayken teslim edilen iletiler kaybedilir.

Bu konudaki bilgiler, yalnızca bir IBM WebSphere MQ 6.0 kuyruk yöneticisi üzerinden yayınlama/abone olma ileti alışverişi kullanıyorsanız, bu bilgiler anlamlıdır.

Bağlantı kapatılmadan önce ya da kapanırken tüketici nesnelere silinmezse, iletiler artık etkin olmayan aboneler için aracı kuyruklarında kalabilir.

Bu durumda, JMS için IBM WebSphere MQ classes for JMS Sınıfları ile birlikte sağlanan Temizleme yardımcı programını kullanarak kuyruklar bu iletilerden temizlenebilir. Bu yardımcı programın

kullanılmasına ilişkin ayrıntılar *IBM WebSphere MQ Using Java* içinde sağlanır. Ayrıca, kuyruğun üzerinde çok sayıda ileti kaldıysa, abone kuyruğunun kuyruk derinliğini de artırmamız gerekebilir.

### **Zamanuyumlu ve zamanuyumsuz ileti tüketicileri**

Zamanuyumlu ileti tüketicisi bir kuyruktan gelen iletileri zamanuyumlu bir şekilde alır ve zamanuyumsuz ileti tüketicisi bir kuyruktan zamanuyumsuz bir iletiyi alır.

### **Zamanuyumlu ileti kullanıcıları**

Zamanuyumlu ileti tüketicisi bir kerede bir ileti alır. `Receive(wait interval)` yöntemi kullanıldığında, çağrı bir ileti için ya da ileti tüketicisi kapatılıncaya kadar milisaniye cinsinden belirtilen bir süreyi bekler.

`ReceiveNoWait()` yöntemi kullanılırsa, zamanuyumlu ileti tüketicisi herhangi bir gecikme olmadan ileti alır; sonraki ileti kullanılabiliriyorsa, hemen alınır; tersi durumda, boş değerli bir ileti nesnesine ilişkin gösterge döndürülür.

### **Zamanuyumsuz ileti tüketicileri**

Kuyruğun üzerinde yeni bir ileti bulunduğunda, uygulama tarafından kaydedilen ileti dinleyicisi çağrılır.

### **XMS içindeki zehirli iletiler**

Zehirli ileti, alan MDB uygulaması tarafından işlenemeyen bir iletidir. Zehirli bir iletiyle karşılaşırsa, 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 `BackoutQueueName` '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 oluşturursa, 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. Ters 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 ve önceki sürümlerde Long Term Support, ve IBM MQ 9.1.4 için Continuous Delivery, ve önceki sürümlerde, yerel kuyruk ya da bir diğer ad kuyruğuna ilişkin hedef kuyruk bir küme kuyruğuyorsa, XMS .NET kuyruğu MQ00\_INPUT\_AS\_Q\_DEF, MQ00\_INQUIRE ve MQ00\_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 , MQ00\_INQUIRE ve MQ00\_FAIL\_IF QUIESCINGaçma seçenekleriyle diğer tüm yerel kuyruk tiplerini açar. XMS .NET ' in özniteliklerin değerlerini sorgulaması için, alan uygulamayı çalıştıran kullanıcının yerel kuyrukta sorgu erişimi olmalıdır.

- **V 9.1.0.5** **V 9.1.5** IBM MQ 9.1.5 ve IBM MQ 9.1.0 Fix Pack 5 sürünlerinden XMS .NET 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.

#### *ASF ' de zehirli iletilerin işlenmesi*

Application Server Facilitis (ASF) olanağını kullandığınızda, MessageConsumer yerine ConnectionConsumer, zehirli iletileri işler. ConnectionConsumer , kuyruğun BackoutThreshold ve BackoutQueueQName özelliklerine göre iletilmekte olan iletileri sağlar.

Bir uygulama ConnectionConsumers' ı kullandığında, bir iletinin yedekleneceği koşullar, uygulama sunucusunun sağladığı oturuma bağlıdır:

- Oturum, AUTO\_RELEASE ya da DUP\_OK\_RELEASE ile birlikte işlem dışı olduğunda, bir ileti yalnızca sistem hatasından sonra ya da uygulama beklenmedik bir şekilde sona erdirildikten sonra yedeklenir.
- Oturum CLIENT\_RELEASE ile işlem dışı olduğunda, onaylanmamış iletiler uygulama sunucusu tarafından `Session.Recover()` çağrısıyla yedeklenebilir.

Typically, the client implementation of MessageListener or the application server calls `Message.Acknowledge()`. `Message.Acknowledge()` , oturum üzerinde şu ana kadar sunulan tüm iletileri onaylar.

- When the session is transacted, unacknowledged messages can be backed out by the application server calling `Session.Rollback()`.

## **Kuyruk tarayıcıları**

Uygulama, kuyruğu kaldırmadan kuyruklardaki iletilere göz atmak için bir kuyruk tarayıcısı kullanır.

Bir kuyruk tarayıcısı yaratmak için, uygulama, bir ISUse nesnesinin Kuyruk Tarayıcısı Yarat yöntemini çağırır ve bu nesnenin göz atılmasına ilişkin kuyruğu tanımlayan bir Hedef nesne olarak belirtilir. Uygulama, bir ileti seçici ile ya da ileti seçicisiz bir kuyruk tarayıcısı yaratabilir.

Kuyruk tarayıcısı yarattıktan sonra, uygulama, kuyruklardaki iletilerin listesini almak için IQueueBrowser nesnesinin GetEnumerator yöntemini çağırabilir. Yöntem, İleti nesnelerinin listesini sarmalayan bir sıralı değer listeleme döndürür. Listedeki ileti nesnelerinin sırası, iletilerin kuyruktan alınacağı sıralama düzeniyle aynıdır. Daha sonra, uygulama sırayla her iletiye göz atmak için sıralı değer numaracısı kullanabilir.

İletiler kuyruğa yerleştirildiği ve kuyruktan kaldırıldığı için, sıralı değer listeleme dinamik olarak güncellenir. Uygulamanın kuyruktan sonraki iletiye göz atmak için IEnumerator.MoveNext() çağrısının her çağrısında, ileti kuyruğun yürürlükteki içeriğini yansıtır.

Bir uygulama, belirli bir kuyruk tarayıcısı için bir kereden fazla GetEnumerator yöntemini çağırabilir. Her arama, yeni bir sıralı değer listeleme döndürür. Bu nedenle, uygulama bir kuyruktaki iletilere göz atmak ve kuyruğun içinde birden çok konum korumak için birden çok sıralı değer listeleme kullanabilir.

Bir uygulama, kuyruktan kaldırmak üzere uygun bir iletiyi aramak için bir kuyruk tarayıcısını kullanabilir ve daha sonra iletiyi kaldırmak için ileti seçicisiyle bir ileti tüketicisi kullanabilir. İleti seçici, iletiyi JMSMessageID üstbilgi alanının değerine göre seçebilir. Bu ve diğer JMS iletisi üstbilgi alanları hakkında bilgi için bkz. [“XMS iletisinde üstbilgi alanları” sayfa 631.](#)

## **İstekçi**

Uygulama, bir istek iletisini göndermek için istekte bulunan bir istekte bulunan ve sonra yanıtı beklemek ve yanıtlamak için kullanılır.

Birçok ileti sistemi uygulaması, bir istek iletisi gönderen ve ardından yanıt bekleyen algoritmalara dayalıdır. XMS , bu uygulama stilinin geliştirilmesine yardımcı olmak için İstekte Bulunana (Requestor) adı verilen bir sınıf sağlar.

Bir istek sahibi yaratmak için, bir uygulama Requestor sınıfının Requestor oluşturucusunu çağırır, bir Oturum nesnesi ve istek iletilerinin nereye gönderileceğini tanımlayan bir Destination nesnesi olarak belirtilir. Oturum, XMSC\_CLIENT\_RAND ' ın onaylanmaması ya da bir alındı bildirimini kipinin alınmamasıdır. Oluşturucu otomatik olarak, yanıt iletilerinin gönderileceği geçici bir kuyruk ya da konu yaratır.

İstek, istekte bulunduktan sonra, istek iletilisini göndermek için Requestor nesnesinin Request yöntemi çağırabilir ve istek iletilisini alan uygulamadan bir yanıt almak ve almak için istekte bulunmayı sağlar. Arama, yanıt alınıncaya kadar ya da oturum sona erinceye kadar bekler (hangisi önce olursa). Her istek iletilisi için istekte bulunanın tek bir yanıtı olması gerekir.

Uygulama, istekte bulunan ' yi kapattığında, geçici kuyruk ya da konu silinir. Ancak ilişkili oturum kapanmıyor.

## Nesne silme

Bir uygulama yarattığı bir XMS nesnesini sildiğinde, XMS nesneye ayrılmış iç kaynakları serbest bırakır.

Bir uygulama XMS nesnesi yarattığında, XMS bellek ve diğer iç kaynakları nesneye ayırır. XMS retains these internal resources until the application explicitly deletes the object by calling the object's close or delete method, at which point XMS releases the internal resources. Bir uygulama önceden silinmiş bir nesneyi silmeye çalışırsa, arama yoksayılar.

Bir uygulama bir Bağlantı ya da Oturum nesnesini sildiğinde, XMS belirli ilişkili nesnelere otomatik olarak siler ve iç kaynaklarını serbest bırakır. Bunlar, Connection ya da Session nesnesi tarafından yaratılan ve nesneden bağımsız bir işlevi olmayan nesnelere. Bu nesnelere Çizelge 84 sayfa 611 içinde gösterilir.

**Not:** bir uygulama bağımlı oturumlarla bir bağlantıyı kapattıysa, bu oturumlara bağımlı tüm nesnelere de silinir. Yalnızca bir Bağlantı ya da Oturum nesnesinin bağımlı nesnelere olabilir.

Çizelge 84. Otomatik olarak silinen nesnelere		
Silinen nesne	Yöntem	Otomatik olarak silinen bağımlı nesnelere
Bağlantı	Bağlantıyı Kapat	ConnectionMetaVeri ve Oturum nesnelere
Session	Oturumu Kapat	MessageConsumer, MessageProducer, QueueBrowseve Requestor nesnelere

## Managed IBM MQ XA transactions through XMS

Yönetilen IBM MQ XA hareketleri XMS aracılığıyla kullanılabilir.

XA hareketlerini XMS aracılığıyla kullanmak için, hareket eden bir oturumun yaratılması gerekir. XA hareketi kullanımda olduğunda, hareket denetimi DTC (Distributed Transaction Coordinator; Dağıtım Hareket Eşgüdümleyicisi) genel hareketlerinden geçer ve XMS oturumlarında bu işlem yoktur. When using XA transactions, Session.commit or Session.rollback cannot be issued on the XMS session. Bunun yerine, işlemlere Transscope.Commit ya da Transscope.Rollback DTC yöntemlerini kullanarak kesinleştirin ya da geri alma yöntemini kullanın. XA hareketi için bir oturum kullanılırsa, oturum kullanılarak yaratılan üretici ya da tüketici XA hareketinin bir parçası olmalıdır. Bunlar XA hareket kapsamı dışındaki herhangi bir işlem için kullanılamaz. Bunlar XA hareketinin dışında Producer.send ya da Consumer.receive gibi işlemler için kullanılamaz.

Aşağıdaki durumlarda bir IllegalStateException kural dışı durum nesnesi verilir:

- XA transacted seansı, Session.commit ya da Session.rollback için kullanılır.
- XA transacted oturumunda kullanılan üretici ya da tüketici nesnelere XA hareketi kapsamının dışında kullanılır.

XA hareketleri zamanuysuz tüketicilerde desteklenmez.

### Not:

1. XA hareketinin kesinleştirilmesinden önce Producer, Consumer, Session ya da Connection nesnesinde bir kapatma işlemi yayınlanabilir. Bu durumda, işlemdeki iletiler geriye işlenir. Benzer

şekilde, XA hareketinin kesinleştirilmesinden önce bağlantı kesilirse, işlemdeki tüm iletiler geriye işlenir. Producer nesnesi için geri alma, iletilerin kuyruğa konmadığından emin olur. Consumer nesnesi için geri döndürme, iletilerin kuyruğun üzerinde kaldığı anlamına gelir.

2. If a Producer object puts a message with TimeToLive in the TransactionScope and a commit is issued after the time is elapsed, the message can expire before the commit is issued. Bu durumda, ileti Consumer nesnelere için kullanılabilir kılınmaz.
3. Session nesnelere iş parçacıkları arasında desteklenmez. İş parçacıkları arasında paylaşılan Session nesnelere yapılan işlemlerin kullanılması desteklenmez.

## XMS temel tipleri

XMS , sekiz Java temel tipi (byte, short, int, long, float, double, double, char ve boolean) eşdeğerleri sağlar. This allows the interchange of messages between XMS and JMS without data becoming lost or corrupted.

Çizelge 85 sayfa 612 lists the Java equivalent data type, size, and minimum and maximum value of each XMS primitive type.

XMS veri tipi	UyumluJava veri tipi	Büyüklik	Minimum değer	Maksimum değer
System.Boolean	boolean	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	short	16 bit	$-2^{15}$ (-32768)	$2^{15}-1$ (32767)
System.Int32	int	32 bit	$-2^{31}$ (-2147483648)	$2^{31}-1$ (2147483647)
System.Int64	long	64 bit	$-2^{63}$ (-9223372036854775808)	$2^{63}-1$ (9223372036854775807)
System.Single	kayan nokta	32 bit	-3.402823E+38 (7 basamaklı basamak sayısı)	3.402823E+38 (7 basamaklı basamak sayısı)
System.Double	double	64 bit	-1.79769313486231E+308 (15 basamaklık basamak sayısı)	1.79769313486231E+308 (15 basamaklı basamak sayısı)

## Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi

Bir uygulama bir özelliğin değerini aldığı anda, değer XMS tarafından başka bir veri tipine dönüştürülebilir. Hangi dönüştürmelerin desteklendiğini ve XMS ' in dönüştürmeleri nasıl gerçekleştirdiğini birçok kural yönetir.

Bir nesnenin özelliğinin bir adı ve değeri vardır; değerin ilişkili bir veri tipi vardır; bu değer, bir özelliğin değerinin de *özellik tipi* olarak da anıldığı bir veri tipine sahiptir.

Bir uygulama, nesnelere özelliklerini almak ve ayarlamak için PropertyContext sınıfının yöntemlerini kullanır. Bir özelliğin değerini almak için, bir uygulama, özellik tipine uygun yöntemi çağırır. Örneğin, bir tamsayı özelliğinin değerini almak için, bir uygulama tipik olarak GetIntÖzellik yöntemi çağırır.

Ancak, bir uygulama bir özelliğin değerini aldığı anda, değer XMS tarafından başka bir veri tipine dönüştürülebilir. Örneğin, bir tamsayı özelliğinin değerini almak için, bir uygulama, özelliğin değerini dizgi olarak döndüren GetStringÖzellik yöntemi çağırabilir. XMS tarafından desteklenen dönüştürmeler Çizelge 86 sayfa 613 içinde gösterilir.

Çizelge 86. 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

Desteklenen dönüştürmeleri aşağıdaki genel kurallara göre düzenler:

- Sayısal özellik değerleri bir veri tipinden diğerine dönüştürülebilmesi için, dönüştürme sırasında hiçbir veri kaybolmaz. Örneğin, System.Int32 veri tipine sahip bir özelliğin değeri, System.Int64 veri tipi ile bir değere dönüştürülebilir, ancak System.Int16 veri tipi ile bir değere dönüştürülemez.
- Herhangi bir veri tipindeki bir özellik değeri bir dizgiye dönüştürülebilir.
- Bir dizgi özelliği değeri, dizginin dönüştürme için doğru şekilde biçimlendirilebilmesi koşuluyla, diğer herhangi bir veri tipine dönüştürülebilir. If an application attempts to convert a string property value that is not formatted correctly, XMS may return errors.
- 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 değerini dizgiye dönüştürürken, true değeri "true" dizgisine dönüştürülür ve false değeri "false" dizgisine dönüştürülür.
- Bir Boole özelliği değerini System.SByte'da içinde olmak üzere sayısal bir veri tipine dönüştürürken, true değeri 1'e dönüştürülür ve false değeri 0 olarak dönüştürülür.
- Bir dizgi özelliği değerini Boole değerine dönüştürürken, "doğru" dizgisi (büyük-küçük harfe duyarlı değil) ya da "1" true değerine dönüştürülür ve "yanlış" dizgisi (büyük-küçük harfe duyarlı değildir) ya da "0" false değerine dönüştürülür. Diğer tüm dizgiler dönüştürülemez.
- Bir dizgi özelliği değerini System.Int32, System.Int64, System.SByte'ya da System.Int16 veri tipli bir değere dönüştürürken, dizginin şu biçimde olması gerekir:

[boşluk] [sign]haneler

Dizgi bileşenleri aşağıdaki gibi tanımlanır:

#### **boşluklar**

İsteğe bağlı olarak baştaki boş karakterler.

#### **İşaret**

İsteğe bağlı artı işareti (+) ya da eksi işareti (-) karakteri.

#### **Rakamlar**

Ardışık rakam karakterleri dizisi (0-9). En az bir sayı karakteri var olmalıdır.

Sayı karakterleri sırasından sonra dizgi, rakam karakteri olmayan diğer karakterleri içerebilir, ancak bu karakterlerin ilk kez ulaşıldığı anda dönüştürme durakları durur. Dizilimin ondalık bir tamsayıyı temsil ettiği varsayılır.

XMS may return an error if the string is not formatted correctly.

- Bir dizgi özelliği değerini System.Double ya da System.Float veri tipli bir değere dönüştürürken, dizginin şu biçimde olması gerekir:

[boşluk] [işaret] [basamaklar] [nokta[d\_basamaklar]] [e\_char[e\_işareti]e\_basamakları]

Dizgi bileşenleri aşağıdaki gibi tanımlanır:

#### **boşluklar**

(İsteğe bağlı) Baştaki boş karakterler.

#### **İşaret**

(İsteğe bağlı) Artı işareti (+) ya da eksi işareti (-) karakteri.

#### **Rakamlar**

Ardışık rakam karakterleri dizisi (0-9). *basamaklar* ya da *d\_basamaklar* içinde en az bir sayı karakteri mevcut olmalıdır.

#### **nokta**

(İsteğe bağlı) Ondalık ayırıcı (.).

#### **d\_haneler**

Ardışık rakam karakterleri dizisi (0-9). *basamaklar* ya da *d\_basamaklar* içinde en az bir sayı karakteri mevcut olmalıdır.

#### **e\_char**

E ya da Eolan bir üstel karakter.

#### **e\_sign**

(İsteğe bağlı) Üst öge için artı işareti (+) ya da eksi işareti (-) karakteri.

#### **e\_basamakları**

Üs için bitişik sayı karakterleri dizisi (0-9). Dizgi bir üstel karakter içeriyorsa en az bir sayı karakteri var olmalıdır.

Sayı karakterleri sırasından sonra ya da bir üssü temsil eden isteğe bağlı karakterler sonra, dizgi sayı karakteri olmayan diğer karakterleri içerebilir, ancak bu karakterlere ilk ulaşıldığında dönüştürme durakları durur. Dizginin, 10 'un gücü olan bir üstel bir ondalık kayan noktalı sayıyı temsil ettiği varsayılır.

XMS may return an error if the string is not formatted correctly.

- Bir sayısal özellik değerini System.SByte veri tipindeki bir özellik değeri de içinde olmak üzere bir dizgiye dönüştürürken, değer, o değere ilişkin ASCII karakterini içeren dizgi değil, değer dizgi gösterimine ondalık sayı olarak dönüştürülür. For example, the integer 65 is converted to the string "65", not the string "A".
- Bir bayt dizisi özellik değerini bir dizgiye dönüştürürken, her bayt bayt 'ı temsil eden 2 onaltılı karaktere dönüştürülür. Örneğin, bayt dizisi {0xF1, 0x12, 0x00, 0xFF} , "F11200FF" dizgisine dönüştürülür.

Bir özellik tipinden diğer veri tiplerine dönüştürmeler hem Özellik hem de PropertyContext sınıflarının yöntemleri tarafından desteklenmektedir.

## **Yineleyiciler**

Bir yineleyici, bir nesne listesini ve listedeki geçerli konumu koruyan bir imleci sarmalayan bir imleçle kaplıdır. The concept of an Iterator, as available in IBM Message Service Client for C/C++, is implemented by using IEnumerator interface in IBM Message Service Client for .NET.

Bir yineleyici yaratıldığında, imlecin konumu ilk nesneden önce olur. Uygulama, sırayla her bir nesneyi almak için bir yineleyici kullanır.

The Iterator class of IBM Message Service Client for C/C++ is equivalent to the Enumerator class in Java. IBM Message Service Client 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 harita iletisinin gövdesindeki ad değeri çiftlerini almak için

- Kuyruklardaki iletilere göz atmak için
- Bir bağlantı tarafından desteklenen JMS tanımlı ileti özelliklerinin adlarını almak için

## Kodlanmış karakter takımı tanıtıcıları

XMS .NET' ta, tüm dizgiler yerli .NET dizgisi kullanılarak geçirilir. Bu, sabit bir kodlamaya sahip olduğu için, bunu yorumlamak için daha fazla bilgi gerekmez. Bu nedenle, XMS .NET uygulamaları için XMLC\_CLIENT\_CCSDID özelliği gerekli değildir.

## XMS hatası ve kural dışı durum kodları

XMS , hataları göstermek için bir dizi hata kodu kullanır. Bu hata kodları, yayından yayına farklılık gösterebileceği için, bu belgelerde açıkça listelenmez. Yalnızca XMS kural dışı durum kodları (XMS\_X\_ biçiminde ...), XMS' un yayın düzeylerinde aynı kaldıkları için belgelenir.

## Automatic IBM MQ Client reconnection through XMS

Configure your XMS client to reconnect automatically following a network, queue manager, or server failure while using IBM WebSphere MQ 7.1 client and later, as the message provider.

Otomatik olarak yeniden bağlanmak üzere istemci bağlantısı yapılandırmak için, MQConnectionFactory sınıfının WMQ\_CONNECTION\_NAME\_LIST ve WMQ\_CLIENT\_RECONNECT\_OPTIONS özelliklerini kullanın. Otomatik istemci yeniden bağlantısı, bağlantı hatasından sonra ya da kuyruk yöneticisini durdurduktan sonra bir seçenek olarak bir istemciyi yeniden kurar. Bazı istemci uygulamalarının tasarımı, bunları otomatik yeniden bağlantı için uygun olmayan bir duruma getirir.

Bağlantı kurulduktan sonra otomatik olarak yeniden bağlanabilir istemci bağlantıları yeniden bağlanabilir duruma gelir.

**Not:** The properties İstemci Yeniden Bağlan Seçenekleri, İstemci Yeniden Bağlan Zamanaşımı, and Bağlantı Ad Listesi can also be set via Client Channel Definitions Table (CCDT) or by enabling the client reconnection via the mqclient.ini file.

**Not:** Yeniden bağlantı özellikleri ConnectionFactory nesnesi ve CCDT ' de olduğu gibi ayarlandıysa, öncelik kuralı aşağıdaki gibidir. ConnectionFactory nesnesinde bağlantı adı listesi özelliğinin varsayılan değeri ayarlandıysa, CCDT öncelik kazanır. Bağlantı adı listesi varsayılan değerine ayarlanmazsa, ConnectionFactory nesnesinde belirlenen özellik değerleri öncelikli olarak uygulanır. Bağlantı adı listesinin varsayılan değeri localhost (1414) değeridir.

## XMS .NET uygulamaları yazılıyor

Bu bölümdeki konular, XMS .NET uygulamalarını yazarken size yardımcı olacak bilgiler sağlar.

### Bu görev hakkında

Bu bölümde, XMS .NET uygulamalarının yazılması için özel bilgiler sağlanmaktadır. XMS uygulamalarını yazma hakkında genel bilgi için bkz. [“XMS uygulamaları yazılıyor” sayfa 597](#).

Bu bölüm aşağıdaki konuları içerir:

- [“.NET için veri tipleri” sayfa 616](#)
- [“.NET içindeki yönetilen ve yönetilmeyen işlemler” sayfa 617](#)
- [“.NET içindeki hedefler” sayfa 617](#)
- [“.NET içindeki özellikler” sayfa 618](#)
- [“.NET içinde var olmayan özellikler işleme” sayfa 618](#)
- [“Error handling in .NET” sayfa 619](#)
- [“Using message and exception listeners in .NET” sayfa 619](#)

## .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 türlerini destekler. XMS .NET veri tipleri, XMS C/C++ için veri tiplerinden farklıdır. Bu konuyu, karşılık gelen veri tiplerini tanımlamak için kullanabilirsiniz.

Aşağıdaki çizelge, ilgili XMS .NET ve XMS C/C++ veri tiplerini göstermektedir ve bunları kısaca açıklar.

<i>Çizelge 87. XMS .NET ve XMS C/C++ için veri tipleri</i>		
<b>XMS .NET türü</b>	<b>XMS C/C++ tipi</b>	<b>Tanım</b>
System.SByte	xmsSBYTE xmsINT8	İmzalı 8 bitlik değer
System.Byte	xmsBYTE xmsUINT8	İmsiz 8 bitlik değer
System.Int16	xmsINT16 xmsSHORT	İmzalı 16 bit değer
System.UInt16	xmsUINT16 xmsUSHORT	İmzalanmamış 16 bit değer
System.Int32	xmsINT32 xmsINT	İmzalı 32 bitlik değer
System.UInt32	xmsUINT32 xmsUINT	İmzasız 32 bitlik değer
System.Int64	xmsLONG xmsINT64	İmzalanmış 64 bitlik değer
System.UInt64	xmsULONG xmsUINT64	İmzasız 64 bit değer
System.Char	xmsCHAR16	İmsiz 16 bit karakter (.NET için Unicode)
System.Single	xmsFLOAT	IEEE 32 bit kayar noktalı sayı
System.Double	xmsDOUBLE	IEEE 64 bit kayar noktalı sayı
System.Boolean	xmsBOOL	True/False değeri
Burada geçerli değil	xmsCHAR	İmzalı ya da İmzalanmamış 8 bitlik değer (imzalanmış ya da imzalanmamış, platforma bağlıdır)
System.Decimal	Burada geçerli değil	96-bit signed integer times $10^0$ through $10^{28}$
System.Object	Burada geçerli değil	Tüm tiplerin tabanı
System.String	Burada geçerli değil	Dizgi Tipi



## .NETiçindeki yönetilen ve yönetilmeyen işlemler

Yönetilen kod, özel olarak .NET ortak dil çalıştırma zamanı ortamı içinde yürütülür ve bu çalıştırma zamanı tarafından sağlanan hizmetlere tamamen bağımlıdır. Uygulamanın herhangi bir bölümü .NET ortak dil çalıştırma zamanı ortamı dışında hizmet çalıştırdığında ya da çağırırsa, uygulama, yönetilmeyen bir uygulama olarak sınıflandırılır.

Şu anda yönetilen .NET ortamında belirli gelişmiş işlevsellik desteklenemez.

Uygulamanız, tam olarak yönetilen ortamda desteklenmeyen bazı işlevler gerektiriyorsa, uygulamanızı uygulamanıza önemli bir değişiklik gerektirmeden, yönetilmeyen ortamı kullanacak şekilde değiştirebilirsiniz. Ancak, bu seçim yapıldığında XMS yığınının yönetilmeyen koddan kullanılmasını sağlamanız gerekir.

## IBM MQ kuyruk yöneticisine yönelik bağlantılar

WMQ\_CM\_Client ile yönetilen bağlantılar, TCP dışı iletişimi ve kanal sıkıştırma özelliğini desteklemeyecek. Ancak, bu bağlantılar yönetilmeyen bir bağlantı kullanılarak desteklenebilir (WMQ\_CM\_CLIENT\_UNManaged). Ek bilgi için [Developing .NET applications](#) başlıklı konuya bakın.

Yönetilmeyen bir ortamdaki yönetilen bir nesneden bir bağlantı üreticisi yaratıyorsanız, bağlantı kipinin değerini XMLC\_WMQ\_CM\_CLIENT\_UNCEOTED olarak el ile değiştirmeniz gerekir.

## WebSphere Application Server hizmet bütünleştirme veriyolu ileti alışverişi altyapısına yönelik bağlantılar

SSL protokolünün (HTTPS dahil) kullanılmasını gerektiren bir WebSphere Application Server hizmet bütünleştirme veriyolu ileti alışverişi altyapısına yönelik bağlantılar şu anda yönetilen kod olarak desteklenmiyor.

## .NETiçindeki hedefler

.NET' ta hedefler, protokol tipine göre yaratılır ve yalnızca oluşturulduğu iletişim kuralı tipinde kullanılabilir.

Hedefler, biri konular ve biri kuyruklar için olmak üzere iki işlev sağlar:

- `IDestination CreateTopic(String topic);`
- `IDestination CreateQueue(String queue);`

Bu işlevler, API ' de aşağıdaki iki nesne üzerinde bulunur:

- `ISession`
- `XMSFactoryFactory`

Her iki durumda da bu yöntemler, aşağıdaki biçimde parametre içerebilen bir URI stili dizesini kabul edebilir:

```
"topic://some/topic/name?priority=5"
```

Diğer bir seçenek olarak, bu yöntemler yalnızca bir hedef adı, yani bir adı olmayan bir adı (// ya da kuyruk: // öneki ve değiştirgeleri olmadan kabul edebilir) kabul edebilir.

Bu, aşağıdaki URI stili dizgisinin anlamına gelir:

```
CreateTopic("topic://some/topic/name");
```

aşağıdaki hedef adı ile aynı sonucu üretir:

```
CreateTopic("some/topic/name");
```

WebSphere Application Server hizmet tümleştirme veriyolu JMS için, konular, hem *topicname* hem de *topicspace* ' i içeren, ancak parametreleri içermeyecek bir kısa forma da belirtilebilir:

```
CreateTopic("topicspace:topicname");
```

## .NET içindeki özellikler

Bir .NET uygulaması, nesnelerin özelliklerini almak ve ayarlamak için PropertyContext arabirimindeki yöntemleri kullanır.

PropertyContext arabirimi, özellikleri alan ve ayarlayan yöntemleri kapsüller. Bu yöntemler, aşağıdaki sınıflar tarafından doğrudan ya da dolaylı olarak edinilir:

- [BytesMessage](#)
- [Bağlantı](#)
- [ConnectionFactory](#)
- [ConnectionMetaVerileri](#)
- [Hedef](#)
- [MapMessage](#)
- [İleti](#)
- [MessageConsumer](#)
- [MessageProducer](#)
- [ObjectMessage](#)
- [QueueBrowser](#)
- [Session](#)
- [StreamMessage](#)
- [TextMessage](#)

Bir uygulama bir özelliğin değerini ayarlarsa, yeni değer özelliğin sahip olduğu önceki değerini yerini alır.

XMS özellikleriyle ilgili daha fazla bilgi için bkz. [XMS nesnelerinin özellikleri](#).

For ease of use, XMS property names and values in XMS are predefined as public constants in a struct called XMSC. Bu sabitlerin adları XMSC biçiminde olur. *sabit*; örneğin, XMSC.USERID (bir özellik adı değişmezi) ve XMSC.DELIVERY\_AS\_APP (bir değer değişmezi).

Additionally, you can access IBM MQ constants by using the IBM.XMS.MQC struct. IBM.XMS ad alanı zaten içe aktarıldığında, bu özelliklerin değerlerine MQC biçiminde (*sabit*) erişebilirsiniz. Örneğin, MQC.MQRO\_COA\_WITH\_FULL\_DATA.

Ayrıca, .NET için hem XMS .NET hem de IBM MQ sınıflarını kullanan ve hem de IBM.XMS ve IBM.WMQ ad alanları, her yinelemenin benzersiz olduğundan emin olmak için MQC struct ad alanını tam olarak nitelemeniz gerekir.

Gelişmiş işlevsellik, yönetilen .NET ortamında şu anda desteklenmektedir. Ayrıntılar için [“.NET içindeki yönetilen ve yönetilmeyen işlemler” sayfa 617](#) dosyasına bakın.

## .NET içinde var olmayan özellikler işleme

The handling of non-existent properties in the XMS .NET is broadly consistent with the JMS specification, and also with the C and C++ implementations of XMS.

JMS ' de var olmayan bir özelliğe erişimde, bir yöntem var olmayan (null) değeri gereken tipe dönüştürmeyi denediğinde Java sistemi kural dışı durumu ile sonuçlanabilir. Bir özellik yoksa, aşağıdaki kural dışı durumlar ortaya çıkar:

- `getStringÖzellik` ve `getObjectÖzellik` boş değer döndürür

- `getBooleanProperty` returns false because `Boolean.valueOf(null)` returns false
- `getIntProperty` etc. throw `java.lang.NumberFormatException` because `Integer.valueOf(null)` throws the exception

XMS .NET ' ta bir özellik yoksa aşağıdaki istisnalar ortaya çıkar:

- `GetStringÖzelliği` ve `GetObjectözelliği` (ve `GetBytesözelliği`) boş değer döndürür ( Javaile aynıdır)
- `GetBooleanözelliği`, `System.NullReferenceException` yayını atar.
- `GetIntÖzelliği` vb. atar: `System.NullReferenceException`

Bu uygulama Java' den farklıdır; ancak, JMS belirtimiyle ve XMS C ve C++ arabirimleriyle genel olarak tutarlı olur. Java uygulaması gibi XMS .NET , çağırın çağrıya `System.Convert` çağrısından herhangi bir kural dışı durum yayıyor. Ancak Java aksine, XMS sistem dönüştürme yordamlarıyla boş değer geçirerek, .NET çerçevesinin yerel davranışını kullanmak yerine, `NullReferenceKural Dışı Durumlarını` belirttik olarak atar. Uygulamanız "abc" gibi bir dizgiye bir özellik ayarlarsa ve `GetIntözelliği` çağırıyorsa, `Convert.ToInt32("abc")` tarafından verilen `System.FormatException` çağırısına, Javaile tutarlı bir şekilde geçirilir. `MessageFormatKural dışı durumu`, yalnızca `setProperty` ve `getProperty` için kullanılan tipler uyumsuzsa oluşur. Bu davranış aynı zamanda Javaile de uyumludur.

## Error handling in .NET

XMS .NET özel durumlarının tümü `System.Exception`' dan türetilmiştir. XMS yöntemi çağrıları, `MessageFormatKural Dışı Durumu`, genel `XMSExceptions` ya da `NullReferenceKural Dışı Durumu` gibi sistem özel durumları gibi belirli XMS özel durumları yayınlatabilir.

Uygulama gereksinimlerine uygun olarak, belirli bir yakalama öbeklerinde ya da genel `System.Exception` yakalama öbeklerinde bu hatalardan birini yakalamak için uygulamalar yazın.

## Using message and exception listeners in .NET

Bir .NET uygulaması, iletileri zamanuyumsuz olarak almak için bir ileti dinleyicisi kullanır ve bir bağlantı ile ilgili sorunun zamanuyumsuz olarak bildirilmesi için bir kural dışı durum dinleyicisini kullanır.

## Bu görev hakkında

Hem iletinin hem de kural dışı durum dinleyicilerinin işlevi, .NET ile C + + için aynıdır. Ancak, bazı küçük uygulama farklılıkları vardır.

## Yordam

- İletileri zamanuyumsuz olarak almak için bir ileti dinleyici ayarlamak için aşağıdaki adımları izleyin:
  - a) İleti dinleyici temsilcisinin imzasıyla eşleşen bir yöntem tanımlayın.  
Tanımladığınız yöntem bir statik ya da somut örnek yöntemi olabilir ve erişilebilir herhangi bir sınıfta tanımlanabilir. Temsilci imzası şu şekildedir:

```
public delegate void MessageListener(IMessage msg);
```

ve yöntemi aşağıdaki gibi tanımlayabilirsiniz:

```
void SomeMethodName(IMessage msg);
```

- b) Aşağıdaki örneğe benzer bir şeyi kullanarak bir temsilci olarak bu yöntemi somutlaştırır:

```
MessageListener OnMsgMethod = new MessageListener(SomeMethodName)
```

- c) Temsilciyi, tüketicinin `MessageListener` özelliğine göre ayarlayarak bir ya da daha fazla tüketiciyle kaydedin:

```
consumer.MessageListener = OnMsgMethod;
```

MessageListener ayarını boş değere ayarlayarak, yetkilendirmeyi kaldırabilirsiniz:

```
consumer.MessageListener = null;
```

- Bir kural dışı durum dinleyicisi oluşturmak için aşağıdaki adımları tamamlayın. Kural dışı durum dinleyicisi, ileti dinleyicisiyle aynı şekilde çalışır, ancak farklı bir temsilci tanımlaması vardır ve bağlantı daha sonra ileti tüketicisi olarak atanır. Bu, C++ ile aynıdır.

a) Yöntemi tanımlayın.

Temsilci imzası şu şekildedir:

```
public delegate void ExceptionListener(Exception ex);
```

ve bu nedenle tanımlanan yöntem şöyle olabilir:

```
void SomeMethodName(Exception ex);
```

b) Aşağıdaki örneğe benzer bir şeyi kullanarak bir temsilci olarak bu yöntemi somutlaştırır:

```
ExceptionListener OnExMethod = new ExceptionListener(SomeMethodName)
```

c) Register the delegate with the connection by setting its ExceptionListener property:

```
connection.ExceptionListener = OnExMethod ;
```

You can remove the delegate by resetting the ExceptionListener to:

```
null: connection.ExceptionListener = null;
```

## Working with XMS .NET administered objects

Bu bölümdeki konular denetlenen nesnelere ilgili bilgi sağlar. XMS uygulamaları, merkezi yönetilen bir nesne havuzundan nesne tanımlarını alabilir ve bu nesnelere, bağlantı fabrikaları ve hedefler oluşturmak için kullanabilir.

### Bu görev hakkında

This section provides information to help with creating and managing administered objects, describing the types of administered object repository that XMS supports. Bu bölümde ayrıca, bir XMS uygulamasının, denetlenen nesnelere havuzuna nasıl bağlantı yaptığı ve gerekli denetlenen nesnelere almak için nasıl bir bağlantı yaptığı da açıklanmaktadır.

Bu bölüm aşağıdaki konuları içerir:

- [“XMS .NET supported types of administered object repository” sayfa 621](#)
- [“Denetlenen nesnelere için XMS .NET özellik eşlemesi” sayfa 621](#)
- [“Denetlenen ConnectionFactory nesnelere için XMS .NET gerekli özellikleri” sayfa 623](#)
- [“XMS .NET required properties for administered Destination objects” sayfa 624](#)
- [“XMS .NET creating administered objects” sayfa 624](#)
- [“XMS .NET creating InitialContext objects” sayfa 625](#)
- [“XMS .NET InitialContext özellikleri” sayfa 625](#)
- [“XMS başlangıç bağlamları için URI biçimi” sayfa 626](#)

- “XMS .NET için JNDI arama web hizmeti” sayfa 627
- “Denetlenen nesnelere XMS .NET alınması” sayfa 627

## XMS .NET supported types of administered object repository

Dosya Sistemi ve LDAP denetimli nesnelere, IBM MQ ve WebSphere Application Server'ye bağlanmak için kullanılabilir; ancak, COS Naming yalnızca WebSphere Application Server' e bağlanmak için kullanılabilir.

Dosya Sistemi nesne dizinleri, diziselleştirilmiş Java Naming Directory Interface (JNDI) nesnelere sahip olur. LDAP nesne dizinleri, JNDI nesnelere içeren dizinlerdir. File System and LDAP object directories can be administered by using either the JMSAdmin tool, which is provided with IBM WebSphere MQ 6.0, or the IBM MQ Explorer, which is provided with IBM WebSphere MQ 7.0 and later. Both the File system and the LDAP object directories can be used to administer client connections by centralizing IBM WebSphere MQ connection factories and destinations. Ağ yöneticisi, aynı merkezi havuza gönderme yapan birden çok uygulamayı konuşlandırabilir ve bu, merkezi havuzda yapılan bağlantı ayarlarındaki değişiklikleri yansıtmak için otomatik olarak güncellenir.

Bir COS adlandırma dizini, WebSphere Application Server service integration bus bağlantı fabrikalarını ve hedeflerini içerir ve WebSphere Application Server yönetim konsolu kullanılarak denetlenebilirler. For an XMS application to retrieve objects from the COS naming directory, a JNDI lookup web service must be deployed. Bu web hizmeti tüm WebSphere Application Server service integration technologies üzerinde kullanılamaz. Ayrıntılar için ürün belgelerine bakın.

**Not:** Nesne dizinindeki değişikliklerin yürürlüğe girmesi için uygulama bağlantılarını yeniden başlatın.

## Denetlenen nesnelere için XMS .NET özellik eşlemesi

XMS .NET uygulamalarını IBM MQ JMS ve WebSphere Application Server bağlantı üreticisi ve hedef nesne tanımlamalarını kullanacak şekilde etkinleştirmek için, bu tanımlamalardan alınan özelliklerin, XMS bağlantı üreticileri ve hedefleri üzerinde ayarlanabilen ilgili XMS özellikleriyle eşlenmelidir.

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 88 sayfa 621 , bağlantı fabrikaları ve varış noktalarının en sık kullanılan özellikleri arasındaki eşlemeleri gösterir. Bu çizelgede gösterilen özellikler yalnızca küçük bir örnek kümedir ve gösterilen tüm özelliklerin tüm bağlantı tipleri ve sunucularla ilişkili olmamasını sağlar.

<i>Çizelge 88. Bağlantı üreticisi ve hedef özellikleri için ad eşleme örnekleri</i>		
IBM MQ JMS özellik adı	XMS özellik adı	WebSphere Application Server service integration bus özellik adı
KALICI (	<u>XMLC_DELIVERY_MODE</u>	
SÜRE BITİMİ (EXP)	<u>XMLC_TIME_TO_LIVE</u>	
PRIORITY (PRI)	<u>XMSC_PRIORITY</u>	
	<u>XMLC_WPM_HOST_NAME</u>	serverName
	<u>XMLC_WPM_BUS_ADı</u>	busName
	<u>XMSC_WPM_TOPIC_SPACE</u>	topicName

**Not:** Çizelge 89 sayfa 622 içinde gösterilen özellikler, JMS için ve XMS .NET ile ilgili geçerlidir.

Çizelge 89. XMS .NET özellikler

Özellik	Nesne tipi				
	CF	QCF	TCF	Kuyruk	Konu
UYGULAMA ADI	Y	Y	Y	N/A	N/A
ASYNCEXCEPTION	Y	Y	Y	N/A	N/A
CCDTURL	Y	Y	Y	N/A	N/A
Kanal	Y	Y	Y	N/A	N/A
CONNECTIONNAMELIST	Y	Y	Y	N/A	N/A
CLIENTRECONNECTIONS	Y	Y	Y	N/A	N/A
CLIENTRECONNECTIONTIMEOUT	Y	Y	Y	N/A	N/A
CLIENTID	N/A	Y	N/A	N/A	N/A
COMMPHDR "1" sayfa 623	Y	N/A	Y	N/A	N/A
COMPUTE "1" sayfa 623	Y	Y	Y	N/A	N/A
CONNOPT "1" sayfa 623	Y	Y	Y	N/A	N/A
CONNTAG "1" sayfa 623	Y	Y	Y	N/A	N/A
Açıklama "1" sayfa 623	N/A	Y	N/A	Y	Y
EXPIRY "1" sayfa 623	N/A	N/A	N/A	Y	Y
FAILIFQUIESCE	Y	Y	Y	Y	Y
Anasistem adı	N/A	Y	N/A	N/A	N/A
YEREL ADRES	N/A	Y	N/A	N/A	N/A
Kalıcılık	N/A	N/A	N/A	Y	Y
PORT	N/A	Y	N/A	N/A	N/A
ÖNCELİK "1" sayfa 623	N/A	N/A	N/A	Y	Y
PROVIDERVERSION "1" sayfa 623	N/A	Y	N/A	N/A	N/A
QASYER	Y	Y	Y	Y	N/A
KUYRUK "1" sayfa 623	N/A	N/A	N/A	Y	N/A
MUTABAKAT ZAMANLANDI	Y	Y	Y	N/A	N/A

Çizelge 89. XMS .NET özellikler (devamı var)

Özellik	Nesne tipi				
	CF	QCF	TCF	Kuyruk	Konu
Konu "1" sayfa 623	N/A	N/A	N/A	N/A	Y
TAŞIMA "1" sayfa 623	N/A	Y	N/A	N/A	N/A

**Not:**

1. Bu özelliklerde uygulama düzeyi özellikleri yoktur, ancak isteğe bağlı olarak yönetilen özellikler kullanılarak ayarlanabilir.

## Denetlenen ConnectionFactory nesneleri için XMS .NET gerekli özellikleri

Bir uygulama bir bağlantı üreticisi yarattığında, bir ileti alışverişi sunucusuna yönelik bağlantı yaratmak için bir dizi özellik tanımlanmalıdır.

Aşağıdaki çizelgelerde listelenen özellikler, bir uygulamanın ileti sistemi sunucusuna bağlantı yaratmak için ayarlanmak üzere gereken en küçük özelliklerdir. Bir bağlantının oluşturulma şeklini özelleştirmek istiyorsanız, uygulamanız gereken şekilde ConnectionFactory nesnesinin ek özelliklerini ayarlayabilir. Daha fazla bilgi için bkz. [ConnectionFactory özellikleri](#). Kullanılabilir özelliklerin eksiksiz bir listesi yer alır.

## IBM MQ kuyruk yöneticisi bağlantı

Çizelge 90. Bir IBM MQ kuyruk yöneticisine bağlanmak için yönetilen ConnectionFactory nesnelere ilişkin özellik ayarları

Gerekli XMS özelliği	Eşdeğer IBM MQ JMS özelliği gerekli
<u>XMLC_CONNECTION_TYPE</u>	XMS , bu özelliği bağlantı üreticisi sınıf adı ve TRANSPORT (TRAN) özelliğinden çıkar.
<u>XMLC_WMQ_HOST_NAME</u>	ANASISTEM ADI (ANASISTEM)
<u>XMLC_WMQ_PORT</u>	PORT
<u>XMLC_WMQ_QUEUE_YÖNETİCİSİ</u>	Kuyruk yöneticisinin adı

## Aracıya gerçek zamanlı bağlantı

Çizelge 91. Bir aracıya gerçek zamanlı bağlantılar için denetlenen ConnectionFactory nesnelere ilişkin özellik ayarları

zorunlu XMS	Eşdeğer IBM MQ JMS özelliği gerekli
<u>XMLC_CONNECTION_TYPE</u>	XMS , bu özelliği bağlantı üreticisi sınıf adı ve TRANSPORT (TRAN) özelliğinden çıkar.
<u>XMLC_RTT_HOST_NAME</u>	ANASISTEM ADI (ANASISTEM)
<u>XMLC_RTT_PORT</u>	PORT

## WebSphere Application Server service integration busbağlantısı

<i>Çizelge 92. Bir WebSphere Application Server service integration busbağlantısı için yönetilen ConnectionFactory nesnelere ilişkin özellik ayarları</i>	
XMS özellik	Tanım
<u>XMLC_CONNECTION_TYPE</u>	Bir uygulamanın bağlandığı ileti sistemi sunucusunun tipi.. Bu, bağlantı üretici sınıfı adından saptanır.
<u>XMLC_WPM_BUS_ADı</u>	Bir bağlantı üreticisi için, uygulamanın bağlandığı hizmet tümleştirme veriyolunun adı ya da hedef için, hedefin bulunduğu hizmet tümleştirme veriyolunun adı.

## XMS .NET required properties for administered Destination objects

Hedef yaratan bir uygulama, yönetilen bir hedef nesnede uygulamanın bazı özellikleri ayarlamalıdır.

<i>Çizelge 93. Yönetilen Hedef nesnelere ilişkin özellik ayarları</i>		
Bağlantı tipi	Özellik	Tanım
IBM MQ Kuyruk Yöneticisi	KUYRUK (QU)	Bağlanmak istediğiniz kuyruk
	KONU (TOP)	Uygulamanın hedef olarak kullandığı konu
Aracıya gerçek zamanlı bağlantı	KONU (TOP)	Uygulamanın hedef olarak kullandığı konu
WebSphere Application Server service integration bus	topicName	Uygulamanız bir konuya bağlanıyorsa
	queueName	Uygulamanızın bir kuyruğa bağlanıyorsa

## XMS .NET creating administered objects

XMS uygulamalarının bir ileti alışverişi sunucusuyla bağlantı yapmak için gerekli olan ConnectionFactory ve Destination nesne tanımlamaları, uygun denetim araçları kullanılarak yaratılmalıdır.

### Başlamadan önce

XMS ' in desteklediği yönetilen nesne havuzunun farklı tipleriyle ilgili daha fazla ayrıntı için bkz. [“XMS .NET supported types of administered object repository” sayfa 621.](#)

### Bu görev hakkında

To create the administered objects for IBM MQ use the IBM MQ Explorer or IBM MQ JMS administration (JMSAdmin) tool.

To create the administered objects for IBM MQ or IBM Integration Bus, use the IBM MQ JMS administration (JMSAdmin) tool.

WebSphere Application Server service integration busiçin yönetilen nesnel oluşturmak üzere WebSphere Application Server yönetim konsolunu kullanın.

**V9.1.2** Bu özelliğe ilişkin denetim araçlarında, kısa süre için **APPLICATIONNAME** ya da **APPNAME** olarak bilinir.

**Not:** JMSAdmin 'i TRANSPORT (MANASIZ) olarak ayarlamak için kullanamazsınız. Bu nedenle, yönetimsel olarak seçilmiş bir uygulama adı kullanarak yönetilmeyen XMS istemcisi almak için aşağıdaki komutu girmeniz gerekir:

```
cf.SetIntProperty(XMSC.WMQ_CONNECTION_MODE, XMSC.WMQ_CM_CLIENT_UNMANAGED);
```

Aşağıdaki adımlar, denetlenen nesnel oluşturmak için neler yapacağınızı özetlemektedir.



## Yordam

1. Bir bağlantı üreticisi yaratın ve seçtiğiniz sunucuya uygulamanızı kullanarak bir bağlantı yaratmak için gereken özellikleri tanımlayın.

XMS ' un bir bağlantı yapmak için gerekli olması gereken en düşük özellikler "[Denetlenen ConnectionFactory nesnelere için XMS .NET gerekli özellikleri](#)" sayfa 623 içinde tanımlanır.

2. Uygulamanızın bağlanacağı ileti sunucusu sunucusunda gerekli hedefi yaratın:

- IBM MQ kuyruk yöneticisine yönelik bağlantı için bir kuyruk ya da konu oluşturun.
- Bir aracıya gerçek zamanlı bağlantı için bir konu oluşturun.
- WebSphere Application Server service integration bus ile bağlantı kurmak için bir kuyruk ya da konu yaratın.

XMS ' un bir bağlantı yapmak için gerekli olması gereken en düşük özellikler "[XMS .NET required properties for administered Destination objects](#)" sayfa 624 içinde tanımlanır.

## XMS .NET creating InitialContext objects

Bir uygulama, denetlenen nesnelere havuzuna, gerekli denetlenen nesnelere almak için bir bağlantı kurmak üzere kullanılacak bir başlangıç bağlamı oluşturmalıdır.

### Bu görev hakkında

Bir InitialContext nesnesi, havuzla bir bağlantıyı sarsalıyor. XMS API, aşağıdaki görevleri gerçekleştirmek için yöntemler sağlar:

- Bir InitialContext nesnesi oluşturma
- Denetlenen nesne havuzunda denetlenen bir nesneye bakın.

## Yordam

- Bir InitialContext nesnesi yaratılmasıyla ilgili daha ayrıntılı bilgi için bkz. [InitialContext for .NET and Properties of InitialContext](#).

## XMS .NET InitialContext özellikleri

InitialContext oluşturucusunun değiştiricileri, tek tip kaynak göstergesi (URI) olarak verilen, denetlenen nesnelere havuzunun yerini içerir. Bir uygulamanın havuzla bağlantı kurmasını sağlamak için, URI içinde yer alan bilgilerden daha fazla bilgi sağlamak gerekebilir.

JNDI 'da ve XMS' un .NET uygulamasında, ek bilgi bir ortam Hashtable içinde oluşturucuya sağlanır.

Denetlenen nesne havuzunun konumu, [XMLC\\_IC\\_URL](#) özelliğinde tanımlanır. Bu özellik genellikle Create çağrısında geçirilir, ancak arama işleminden önce farklı bir adlandırma dizinine bağlanmak için değiştirilebilir. FileSystem ya da LDAP bağlamları için bu özellik, dizinin adresini tanımlar. COS adlandırması için bu, JNDI dizinine bağlanmak için bu özellikleri kullanan web hizmetinin adresidir.

Aşağıdaki özellikler, JNDI dizinine bağlanmak için kullanacakları web hizmetine değiştirilmemiş olarak geçirilir.

- [XMLC\\_IC\\_PROVIDER\\_URL](#)
- [XMSC\\_IC\\_SECURITY\\_CREDENTIAL](#)
- [XMSC\\_IC\\_SECURITY\\_AUTHENTICATION](#)
- [XMSC\\_IC\\_SECURITY\\_PRINCIPAL](#)
- [XMSC\\_IC\\_SECURITY\\_PROTOCOL](#)

## XMS başlangıç bağlamları için URI biçimi

Denetlenen nesnelerin havuzunun konumu, tek tip bir kaynak göstergesi (URI) olarak sağlanır. URI biçimi bağlam tipine bağlıdır.

### FileSystem bağlamı

FileSystem bağlamına ilişkin URL, dosya sistemi tabanlı dizinin yerini gösterir. URL 'nin yapısı RFC 1738 tarafından tanımlandığı şekilde, *Uniform Resource Locators (URL)*(Uniform Resource Locators (URL)): URL adresi `file://` önekiye sahiptir ve bu önekin ardından sözdizimi, XMS ' in çalıştığı sistemde açılabilen geçerli bir dosya tanımdır.

Bu sözdizimi altyapıya özgü olabilir ve '/' ayırıcılar ya da '\' ayırıcıları kullanabilir. '\' kullanırsanız, her ayırıcının ek bir '\' kullanılarak çıkış karakteri olması gerekir. Bu, .NET çerçevesinin ayırıcıyı aşağıdaki gibi bir kaçış karakteri olarak yorumlamaya çalışmamasını 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

For the LDAP context, the basic structure of the URL is as defined by RFC 2255, *LDAP URL Biçimi*, with the case-insensitive prefix `ldap://`

Aşağıdaki örnekte, kesin sözdizim gösterilmektedir:

```
LDAP://[Hostname][:Port]["/" [DistinguishedName]]
```

Bu sözdizimi, RFC içinde tanımlandığı şekilde, ancak öznitelikler, kapsam, süzgeçler ya da uzantılar için destek verilmeksizin tanımlanır.

Bu sözdizimine ilişkin örnekler:

```
ldap://madison:389/cn=JMSSData,dc=IBM,dc=UK
ldap://madison/cn=JMSSData,dc=IBM,dc=UK
LDAP:///cn=JMSSData,dc=IBM,dc=UK
```

### WSS bağlamı

For the WSS context, the URL is in the form of a web services endpoint, with the prefix `http://`.

Alternatively, you can use the prefix `cosnaming://` or `wsvc://`,

Bu iki örnek, http üzerinden erişilen URL ile WSS bağlamını kullandığınız anlamına gelir. Bu, ilk bağlam tipinin doğrudan doğruya URL adresinden türetilebilmesini sağlar.

Bu sözdizimine örnek olarak şunlar verilebilir:

```
http://madison.ibm.com:9080/xmsjndi/services/JndiLookup
cosnaming://madison/jndilookup
```

## XMS .NET için JNDI arama web hizmeti

To access a COS naming directory from XMS, a JNDI Lookup web service must be deployed on a WebSphere Application Server service integration bus server. Bu web hizmeti, COS adlandırma hizmetindeki Java bilgilerini XMS uygulamalarının okuyabileceği bir forma çevirir.

Web hizmeti, kuruluş dizini içinde bulunan SIBXJndiLookupEAR.ear kurumsal arşiv dosyası içinde sağlanır. For the current release of IBM Message Service Client for .NET, SIBXJndiLookupEAR.ear can be found in the *install\_dir*\java\lib directory. Bu, yönetim konsolu ya da wsadmin komut dosyası oluşturma aracı kullanılarak bir WebSphere Application Server service integration bus sunucusu içinde kurulabilir. Web hizmeti uygulamalarının konuşlandırılmasına ilişkin ek bilgi edinmek için ürün belgelerine bakın.

Web hizmetini XMS uygulamaları içinde tanımlamak için, InitialContext nesnesinin XMLC\_IC\_URL özelliğini web hizmeti uç noktası URL 'sine ayarlamanız yeterlidir. Örneğin, web hizmeti MyServer adlı bir sunucu anasisteminde konuşlandırıldıysa, bir web hizmeti uç noktası URL 'si örneği:

```
wsvc://MyHost:9080/SIBXJndiLookup/services/JndiLookup
```

XMLC\_IC\_URL özelliğinin ayarlanması, tanımlı uç noktadaki web hizmetini çağırmak için InitialContext arama çağrılarında izin verir; böylece, gereken denetimli nesne COS adlandırma hizmetinden aranır.

.NET uygulamaları web hizmetini kullanabilir. Sunucu tarafı devreye alımı, XMS C, /C++ ve, XMS .NET için aynıdır. XMS .NET, web hizmetini doğrudan Microsoft .NET Framework aracılığıyla çağırır.

## Denetlenen nesnelerin XMS .NET alınması

XMS retrieves an administered object from the repository using the address provided when the InitialContext object is created, or in the InitialContext properties.

Alınacak nesneler aşağıdaki tipte adlara sahip olabilir:

- Hedef nesneyi tanımlayan basit bir ad; örneğin, SalesOrders adlı bir kuyruk hedefi.
- SubContexts '/' ile ayrılmış, '/' ile ayrılmış ve nesne adıyla sona ermesi gereken bileşik ad. Birleşik ad örneği: "Warehouse/PickLists/DispatchQueue2"; burada Warehouse ve Picklists, adlandırma dizininde SubContexts ve DispatchQueue2 bir Hedef nesnesinin adıdır.

## Uygulamaların daha yeni bir XMS sürümünü kullanmalarını önleme

Varsayılan olarak, daha yeni bir XMS sürümü kurulduğunda, önceki sürümü kullanan uygulamalar otomatik olarak neweryeniden derleme yapmadan daha yeni sürüme geçer. Ancak, uygulama yapılandırma dosyasında bir öznitelik ayarlayarak uygulamaların daha yeni bir sürümünü kullanmasını önleyebilirsiniz.

### Bu görev hakkında

Birden çok sürüm birlikte bulunma özelliği, daha yeni bir XMS sürümünün kurulmasının önceki XMS sürümünün üzerine yazmamasını sağlar. Bunun yerine, benzer XMS .NET düzeneklerinin birden çok eşgörünümü, Global Assembly Cache (GAC) içinde birlikte bulunur, ancak farklı sürüm numaralarına sahip olur. Dahili olarak GAC, uygulama çağrılarını en son XMS sürümüne yönlendirmek için bir ilke dosyası kullanır. Uygulamaların yeniden derlenmesi gerekmeksizin çalışır ve daha yeni XMS .NET sürümünde kullanılabilir olan yeni özellikleri kullanabilirler.

### Yordam

- Bir uygulamanın daha eski XMS .NET sürümünü kullanması gerekiyorsa, uygulama yapılandırma dosyasında publisherpolicy özniteliğini no olarak ayarlayın.

**Not:** Uygulama yapılandırma dosyası, dosyanın ilişkilendirdiği yürütülebilir programın adından, .config sonekiyle birlikte bir ad içeren bir dosyadır. For example, the application configuration file for text.exe would have the name text.exe.config.

Ancak, herhangi bir zamanda, bir sistemin tüm uygulamaları aynı XMS .NET sürümünü kullanır.

## XMS uygulamaları için iletişimi güvenlik altına alma

Bu bölümde, XMS uygulamalarının bir WebSphere Application Server service integration bus ileti alışverişi altyapısına ya da IBM MQ kuyruk yöneticisine SSL aracılığıyla bağlanmasını sağlamak için güvenli iletişim kurulmasına ilişkin bilgiler sağlanır.

### Bu görev hakkında

Bölüm aşağıdaki konuları içerir:

- [“IBM MQ kuyruk yöneticisine güvenli bağlantı” sayfa 628](#)
- [“Bir IBM MQ kuyruk yöneticisiyle XMS bağlantıları içinCipherSuite ve CipherSpec ad eşlemeleri” sayfa 629](#)
- [“WebSphere Application Server service integration bus ileti alışverişi altyapısına güvenli bağlantılar” sayfa 629](#)
- [“WebSphere Application Server service integration busile bağlantı içinCipherSuite ve CipherSpec ad eşlemeleri” sayfa 630](#)

### IBM MQ kuyruk yöneticisine güvenli bağlantı

Bir XMS .NET uygulamasını bir IBM MQ kuyruk yöneticisinde güvenli bağlantı sağlamak üzere etkinleştirmek için, ilgili özelliklerin ConnectionFactory nesnesinde tanımlı olması gerekir.

The protocol used in the encryption negotiation can be either Secure Sockets Layer (SSL) or Transport Layer Security (TLS), depending on which CipherSuite you specify in the ConnectionFactory object.

IBM WebSphere MQ 7.0.0 Fix Pack 1 ve daha sonraki istemci kitaplıklarını kullanıyorsanız ve bir IBM WebSphere MQ 7.0 kuyruk yöneticisine bağlansanız, XMS uygulamasında aynı kuyruk yöneticisine birden çok bağlantı yaratabilirsiniz. Ancak, farklı kuyruk yöneticisine bağlantı yapılmasına izin verilmez. Girişimde bulunursanız, MQRC\_SSL\_ALREADY\_INITIALIZED hatasını elde edin.

IBM WebSphere MQ 6.0 ve sonraki istemci kitaplıklarını kullanıyorsanız, önce herhangi bir önceki SSL bağlantısını kapadığınızda bir SSL bağlantısı yaratabilirsiniz. Aynı süreçten aynı ya da farklı kuyruk yöneticilerine birden çok eşzamanlı SSL bağlantısının kullanılmasına izin verilmez. Birden fazla istek girişiminde bulunursanız, SSL bağlantısı için istenen bazı parametrelerin yoksayıldığı anlamına gelen MQRC\_SSL\_ALREADY\_INITIALIZEDuyarısını elde edebilirsiniz.

ConnectionFactory properties for connections via SSL to an IBM MQ queue manager, with a brief description, are shown in the following table:

Özellğin adı	Tanım
<a href="#">XMLC_WM_Q_SSL_CERT_STORES</a>	Kuyruk yöneticisine SSL bağlantısında kullanılacak sertifika iptal listelerini (CRL) bulunduran sunucuların yerleri.
<a href="#">XMLC_WM_Q_SSL_CIPHER_SPEC</a>	Bir kuyruk yöneticisine güvenli bir bağlantıda kullanılacak CipherSpec ' in adı.
<a href="#">XMSC_WM_Q_SSL_CIPHER_SUIT</a>	Bir kuyruk yöneticisine TLS bağlantısında kullanılacak CipherSuite ' in adı. Güvenli bağlantının kararlaştırılmasında kullanılan protokol, belirtilen CipherSuite' e bağlıdır.
<a href="#">XMLC_WM_Q_SSL_CRYPTO_HW</a>	İstemci sistemine bağlı şifreleme donanımına ilişkin yapılandırma ayrıntıları.

Çizelge 94. SSL aracılığıyla bir IBM MQ kuyruk yöneticisine bağlanmak için kullanılan ConnectionFactory özellikleri (devamı var)

Özellik adı	Tanım
<u>XMLC_WMQ_SSL_FIPS_REQUIRED</u>	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.
<u>XMLC_WMQ_SSL_KEY_REPOSITORY</u>	Anahtarların ve sertifikaların saklandığı anahtar veritabanı dosyasının konumu.
<u>XMLC_WMQ_SSL_KEY_RESETCOUNT</u>	KeyResetSayısı, gizli anahtar yeniden anlaşılmadan önce SSL etkileşimi içinde gönderilen ve alınan toplam şifrelenmemiş bayt sayısını temsil eder.
<u>XMLC_WMQ_SSL_PEER_NAME</u>	Kuyruk yöneticisine SSL bağlantısında kullanılacak eşdüzey ad.

### **Bir IBM MQ kuyruk yöneticisiyle XMS bağlantıları için CipherSuite ve CipherSpec ad eşlemeleri**

InitialContext , SSLCIPHERSUITE JMSAdmin Connection Factory özelliği SSLCIPHERSUITE ile XMS near-eşdeğeri XMLC\_WMQ\_SSL\_CIPHER\_SPEC çevreleri arasında çevrim yapar. XMLC\_WMQ\_SSL\_CIPHER\_SUITE için bir değer belirtirseniz, ancak XMLC\_WMQ\_SSL\_CIPHER\_SPEC için değeri atlayın. Benzer bir çeviri gereklidir.

Çizelge 95 sayfa 629 , kullanılabilir CipherSpecs ve JSSE CipherSuite eşdeğerlerini listeler.

Çizelge 95. 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:** TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA kullanımdan kaldırıldı. However, it can still be used to transfer up to 32 GB of data before the connection is terminated with error AMQ9288. Bu hatayı önlemek için, üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpec komutunu kullanırken gizli anahtar sınırlamayı etkinleştirmeniz gerekir.

### **WebSphere Application Server service integration bus ileti alışverişi altyapısına güvenli bağlantılar**

To enable an XMS .NET application to make secure connections to a WebSphere Application Server service integration bus messaging engine, the relevant properties must be defined in the ConnectionFactory object.

XMS , WebSphere Application Server service integration bus ile bağlantı için SSL ve HTTPS desteği sağlar. SSL ve HTTPS, kimlik doğrulama ve gizlilik için güvenli bağlantılar sağlar.

WebSphere güvenliği gibi, XMS güvenliği, güvenli bir bağlantı kararlaştırılırken kullanılan algoritmaları belirtmek üzere CipherSuites kullanımını içeren JSSE güvenlik standartları ve adlandırma kurallarıyla ilgili olarak yapılandırılır. The protocol used in the encryption negotiation can be either SSL or TLS, depending on which CipherSuite you specify in the ConnectionFactory object.

Çizelge 96 sayfa 630 , ConnectionFactory nesnesinde tanımlanması gereken özellikleri listeler.

Çizelge 96. WebSphere Application Server service integration bus ileti alışverişi altyapısına güvenli bağlantılar için ConnectionFactory özellikleri

Özellik adı	Tanım
<u>XMLC_WPM_SSL_CIPHER_SUITE</u>	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.
<u>XMLC_WPM_SSL_KEYRING_LABEL</u>	Sunucuyla kimlik doğrulaması yaparken kullanılacak sertifika.

Aşağıda, WebSphere Application Server service integration bus ileti alışverişi altyapısına yönelik güvenli bağlantılar için bir ConnectionFactory özellikleri örneği yer alıyor:

```
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);
```

Where chain\_name, BootstrapTunneledSecureMessaging ya da BootstrapSecureMessaging değerine ayarlanmalıdır ve kapı\_numarası, 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 gösterilmektedir:

```
/* 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 busile bağlantı içinCipherSuite ve CipherSpec ad eşlemeleri**

GSKit, CipherSuitesyerine CipherSpecs kullandığından, XMLC\_WPM\_SSL\_CIPHER\_SUITE özelliğinde belirtilen JSSE stili CipherSuite adları GSKit stili CipherSpec adlarıyla eşlenmelidir.

Çizelge 97 sayfa 630 , tanınan her CipherSuiteiçin eşdeğer CipherSpec ' i listeler.

Çizelge 97. Kullantılabilir CipherSuites ve eşdeğeri 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:** TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA kullanımdan kaldırıldı. However, it can still be used to transfer up to 32 GB of data before the connection is terminated with error AMQ9288. Bu hatayı önlemek için, üçlü DES kullanmaktan kaçınmanız ya da bu CipherSpeckomutunu kullanırken gizli anahtar sıfırlamayı etkinleştirmeniz gerekir.

## XMS ileti

Bu bölümde, XMS iletilerinin yapısı ve içeriği ele alınmıştır ve uygulamaların XMS iletilerinin nasıl işleyeceğini açıklar.

Bu bölüm aşağıdaki konuları içerir:

- [“XMS iletisinin parçaları” sayfa 631](#)
- [“XMS iletisinde üstbilgi alanları” sayfa 631](#)
- [“XMS iletisinin özellikleri” sayfa 632](#)
- [“XMS iletisinin gövdesi” sayfa 635](#)
- [“İleti seçicileri” sayfa 638](#)
- [“Mapping XMS messages onto IBM MQ messages” sayfa 639](#)

### XMS iletisinin parçaları

XMS iletisi, bir üstbilgiden, bir özellik kümesinden ve bir gövden oluşur.

#### Üstbilgi

Bir iletinin üstbilgisi alanlar içerir ve tüm iletiler aynı üstbilgi alanları kümesini içerir. XMS ve uygulamalar, iletileri tanımlamak ve yönlendirmek için üstbilgi alanlarının değerlerini kullanır. Üstbilgi alanlarıyla ilgili daha fazla bilgi için bkz. [“XMS iletisinde üstbilgi alanları” sayfa 631](#).

#### Özellik kümesi

İletinin özellikleri, iletiye ilişkin ek bilgileri belirtir. Tüm iletiler üstbilgi alanları kümesine sahip olsa da, her ileti farklı bir özellik kümesine sahip olabilir. Daha fazla bilgi için, bkz. [“XMS iletisinin özellikleri” sayfa 632](#).

#### Gövde

Bir iletinin gövdesinde uygulama verileri bulunur. Daha fazla bilgi için, bkz. [“XMS iletisinin gövdesi” sayfa 635](#).

Bir uygulama, almak istediği iletileri seçebilir. Seçim ölçütlerini belirten ileti seçicileri kullanarak. Ölçütler, belirli üstbilgi alanlarının değerlerini ve bir iletinin özelliklerinin herhangi birinin değerlerini temel alarak olabilir. İleti seçiciyle ilgili daha fazla bilgi için bkz. [“İleti seçicileri” sayfa 638](#).

### XMS iletisinde üstbilgi alanları

To allow an XMS application to exchange messages with a WebSphere JMS application, the header of an XMS message contains the JMS message header fields.

Bu üstbilgi alanlarının adları JMSönekiyle başlar. JMS ileti üstbilgisi alanlarına ilişkin açıklamalar için *Java Message Service Specification* başlıklı konuya bakın.

XMS implements the JMS message header fields as attributes of a Message object. Her üstbilgi alanının, değerini belirlemek ve almak için kendi yöntemleri vardır. Bu yöntemlere ilişkin açıklamalar için *IMessage* başlıklı konuya bakın. Bir üstbilgi alanı her zaman okunabilir ve yazılabilir bir alandır.

Çizelge 98 sayfa 631 , JMS ileti üstbilgisi alanlarını listeler ve iletilen bir ileti için her bir alanın değerinin nasıl ayarlanacağını gösterir. Uygulama bir ileti gönderdiğinde ya da JMSReelled durumunda, bir uygulama ileti aldığı anda, bazı alanlar otomatik olarak XMS tarafından ayarlanır.

Çizelge 98. JMS ileti üstbilgisi alanları. ]	
JMS ileti üstbilgisi alanının adı	İletilen bir ileti için değerin nasıl ayarlanıyor ( yöntem biçiminde [sınıf])
JMSCorrelationID	Set JMSCorrelationID [ İleti]
JMSDeliveryMode	[MessageProducer] Gönder
JMSHedef	[MessageProducer] Gönder

Çizelge 98. JMS ileti üstbilgisi alanları. ] (devamı var)

JMS ileti üstbilgisi alanının adı	İletilen bir ileti için değerin nasıl ayarlanıyor ( yöntem biçiminde [sınıf])
JMSSüresi	[MessageProducer] Gönder
JMSMessageID	[MessageProducer] Gönder
JMSönceliği	[MessageProducer] Gönder
JMSRedird	[MessageConsumer] Al
JMSReplyTo	Set JMSReplyTo [ İleti]
JMSTimestamp	[MessageProducer] Gönder
JMSType	JMSType 'ı ayarla [ İleti]

## XMS iletisinin özellikleri

XMS , üç tür ileti özelliğini destekler: JMS tanımlı özellikler, IBM tanımlı özellikler ve uygulama tarafından tanımlanan özellikler.

An XMS application can exchange messages with a WebSphere JMS application because XMS supports the following predefined properties of a Message object:

- 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:

- A JMS name, for a JMS-defined property, or a WebSphere JMS name, for an IBM-defined property.  
Bu ad, özelliğin JMS ya da WebSphere JMSiçinde bulunduğu addır ve aynı zamanda bu özelliğe sahip bir iletiyle iletilen addır. Bir XMS uygulaması, bir ileti seçici ifadesindeki özelliği tanımlamak için bu adı kullanır.
- Bir ileti seçici ifadesi dışında tüm durumlarda özelliği tanımlamak için bir XMS adı. Her XMS adı, IBM.XMS.XMSC sınıfında adlandırılmış bir sabit olarak tanımlanır. Adlandırılmış değişmez değeri, ilgili JMS ya da WebSphere JMS adıdır.

Önceden tanımlanmış özelliklere ek olarak, bir XMS uygulaması kendi ileti özellikleri kümesini yaratabilir ve kullanabilir. Bu özellikler *Uygulama tanımlı özellikler*.olarak adlandırılır.

Bir uygulama ileti yarattıktan sonra, iletinin özellikleri okunabilir ve yazılabilir olur. Uygulama iletiyi gönderdikten sonra özellikler okunabilir ve yazılabilir olarak kalır. Bir uygulama bir ileti aldığıında, iletinin özellikleri salt okunur olur. Bir ileti, bir iletinin özellikleri salt okunur olduğunda İleti sınıfının Clear Properties yöntemini çağırıyorsa, özellikler okunabilir ve yazılabilir olur. Yöntem özellikleri de temizler.

Alınan ileti, ileti özelliklerini temizledikten sonra iletildiğinde, ileti özellikleri temizlenen standart bir WMQ XMS for .NET BytesMessage iletilmesinin davranışına uygun bir şekilde işlev görecektir.

Ancak, aşağıdaki özellikler kaybedildiğinden, bu önerilmez:

- JMS\_IBM\_Encoding özellik değeri, ileti verilerinin anlamlı bir şekilde kodu çözülemez olduğunu ima ediyor.
- JMS\_IBM\_Format özellik değeri; (MQMD ya da yeni MQRFH2) ileti üstbilgisi ile var olan üstbilgiler arasında üstbilgi zincirinin kırılacağını ima eder.

Bir iletinin tüm özelliklerinin değerlerini saptamak için, bir uygulama İleti sınıfının "get Properties" yöntemini çağırabilir. Yöntem, her özellik nesnesinin bir iletinin bir özelliğini gösterdiği özellik nesnelerinin listesini sarmalayan bir yineleyici yaratır. Böylece uygulama, her bir Özellik nesnesini sırayla almak için



Yineator sınıfının yöntemlerini kullanabilir ve her özelliğin adını, veri tipini ve değerini almak için Özellik sınıfı yöntemlerini kullanabilir.

### **JMS-iletinin tanımlı özellikleri**

Bir iletinin JMS tanımlı özellikleri hem XMS hem de WebSphere JMStarafından desteklenir.

Çizelge 99 sayfa 633 , hem XMS hem de WebSphere JMStarafından desteklenen bir iletinin JMS tanımlı özelliklerini listeler. JMStanımlı özelliklerin bir açıklaması için bkz. *Java Message Service Belirtimi*. JMStanımlı özellikler, bir aracıya gerçek zamanlı bağlantı için geçerli değildir.

Çizelge, her özelliğin veri tipini belirtir ve iletilen bir ileti için özelliğin değerinin nasıl ayarının kullanılacağını belirtir. Uygulama ileti gönderdiğinde ya da JMSXDeliveryCount(JMSXDeliveryCount) durumunda, bazı özellikler XMS tarafından otomatik olarak ayarlanır; bir uygulama ileti alır.

<i>Çizelge 99. JMS-iletinin tanımlı özellikleri</i>			
<b>JMS tanımlı özelliğinXMS adı</b>	<b>JMS ad</b>	<b>Veri tipi</b>	<b>İletilen bir ileti için değerin nasıl ayarlanıyor ( yöntem biçiminde [sınıf])</b>
JMS_APPID	JMSXAppID	System.String	[MessageProducer] Gönder
JMSX_DELIVERY_COUNT	JMSXDeliveryCount	System.Int32	[MessageConsumer] Al
JMSX_GROUPID	JMSXGroupID	System.String	Dizgi Özelliğini Ayarla [PropertyContext]
JMSX_GROUPSEQ	JMSXGroupSeq	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMSX_USERID	JMSXUserID	System.String	[MessageProducer] Gönder

### **IBM-iletinin tanımlı özellikleri**

Bir iletinin birden çok IBMtanımlı özelliği, XMS ve WebSphere JMStarafından desteklenir.

Çizelge 100 sayfa 633 lists the IBM defined properties of a message that are supported by both XMS and WebSphere JMS. 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 ayarının kullanılacağını belirtir. Uygulama bir ileti gönderdiğinde bazı özellikler otomatik olarak XMS tarafından ayarlanır.

<i>Çizelge 100. IBM-iletinin tanımlı özellikleri</i>			
<b>IBM tanımlı özelliğinXMS adı</b>	<b>WebSphere JMS adı</b>	<b>Veri tipi</b>	<b>İletilen bir ileti için değerin nasıl ayarlanıyor ( yöntem biçiminde [sınıf])</b>
JMS_IBM_XX_ENCODE_C ASE_ONE charter_set	JMS_IBM_Character_Set	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_ENCODING	JMS_IBM_kodlama	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_EXCEPTIONMESSAGE	JMS_IBM_ExceptionMessage	System.String	[MessageConsumer] Al
JMS_IBM_EXCEPTIONREASON	JMS_IBM_ExceptionReason	System.Int32	[MessageConsumer] Al

Çizelge 100. IBM-iletinin tanımlı özellikleri (devamı var)

IBM tanımlı özelliğinXMS adı	WebSphere JMS adı	Veri tipi	İletilen bir ileti için değerin nasıl ayarlanıyor (yöntem biçiminde [sınıf])
JMS_IBM_EXCEPOTIMESTAMP	JMS_IBM_ExceptionTime stamp	System.Int64	[MessageConsumer] Al
JMS_IBM_EXCEPTIONPROBLEM Hedef	JMS_IBM_ExceptionProblemHedef	System.String	[MessageConsumer] Al
JMS_IBM_FEEDBACK	JMS_IBM_Geri	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS__IBM_	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	[MessageProducer] Gönder
JMS_IBM_PUTDATE	JMS_IBM_PutDate	System.String	[MessageProducer] Gönder
JMS_IBM_PUTTIME	JMS_IBM_PutTime	System.String	[MessageProducer] Gönder
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_Disard_Msg	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_EXCEPTION	JMS_IBM_Report_Exception	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_EXPIRATION	JMS_IBM_Report_SonKullanma	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
JMS_IBM_REPORT_NAN	JMS_IBM_Report_NAN	System.Int32	Tamsayı Özelliğini Ayarla [PropertyContext]
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]

Çizelge 100. IBM-iletinin tanımlı özellikleri (devamı var)

IBM tanımlı özelliğin XMS adı	WebSphere JMS adı	Veri tipi	İletilen bir ileti için değerin nasıl ayarlanıyor (yöntem biçiminde [sınıf])
JMS_IBM_SYSTEM_MESS AGEID	JMS_IBM_System_Messa geID	System.String	[MessageProducer] Gönder

### İletiyeye ilişkin uygulama tanımlı özellikler

Bir XMS uygulaması kendi ileti özellikleri kümesini yaratabilir ve kullanabilir. Bir uygulama bir ileti gönderdiğinde, bu özellikler de iletiyle iletilir. Alma uygulaması, ileti seçicilerinin kullanılması, daha sonra bu özelliklerin değerlerine dayalı olarak hangi iletilerin almak istediğini seçebilir.

Bir WebSphere JMS uygulamasının bir XMS uygulaması tarafından gönderilen iletileri seçmesine ve işlenmesine izin vermek için, uygulama tanımlı bir özelliğin adı, ileti seçici ifadelerinde tanıtıcıların oluşturulmasına ilişkin kurallara uygun olmalıdır. Daha fazla bilgi için bkz. “JMS’indeki ileti seçicileri” sayfa 128. Uygulama tanımlı bir özelliğin değeri şu veri tiplerinden birine sahip olmalıdır: System.Boolean, System.SByte, System.Int16, System.Int32, System.Int64, System.Float, System.Double ya da System.String.

### XMS iletisinin gövdesi

Bir iletinin gövdesinde uygulama verileri bulunur. Ancak, bir iletinin gövdesi olamaz ve yalnızca üstbilgi alanlarını ve özellikleri sıkıştırır.

XMS , ileti gövdesinin beş tipini destekler:

#### Bayt

Gövde bir bayt akışı içerir. Bu gövde tipine sahip bir iletinin *byte iletisi* adı verilir. IMessage arabirimi, bir bayt iletisinin gövdedeki işleyişi işleme yöntemlerini içerir.

#### Eşlem

Gövde, her değerin ilişkili bir veri tipine sahip olduğu ad-değer çiftleri kümesini içerir. Bu gövde tipine sahip bir iletinin *eşleme iletisi* adı verilir. IMessage arabirimi, bir eşlem iletisinin gövdesinin işlenmesine ilişkin yöntemleri içerir.

#### Nesne

Gövde, diziselleştirilmiş bir Java ya da .NET nesnesi içerir. Bu gövde tipine sahip bir ileti *nesne iletisi* olarak adlandırılır. IMessage arabirimi, bir nesne iletisinin gövdesinin işlenmesine ilişkin yöntemleri içerir.

#### Akış

Gövde, her değerin ilişkili bir veri tipine sahip olduğu bir değerler akışı içerir. Bu gövde tipine sahip bir ileti *akış iletisi* olarak adlandırılır. IMessage arabirimi, bir akış iletisinin gövdesinin işlenmesine ilişkin yöntemleri içerir.

#### Metin

Gövde bir dizgi içeriyor. Bu gövde tipine sahip bir ileti *metin iletisi* olarak adlandırılır. IMessage arabirimi, bir metin iletisinin gövdesinin işlenmesine ilişkin yöntemleri içerir.

IMessage arabirimi tüm ileti nesnelerinin üst ögesidir ve ileti alışverişi işlevlerinde XMS ileti tiplerinden herhangi birini temsil etmek için kullanılabilir.

Bu veri tiplerinin her birinin boyutu ve alt sınırı ve en düşük değerleri hakkında bilgi için bkz. Çizelge 85 sayfa 612.

### Byte iletileri

Bir baytlar iletisinin gövdesi, bayt akışını içerir. Gövde yalnızca gerçek verileri içerir ve bu verileri yorumlamak için gönderme ve alma uygulamalarının sorumluluğunda olur.

Bytes messages are useful if an XMS application needs to exchange messages with applications that are not using the XMS or JMS application programming interface.

Bir uygulama byte iletisi yarattıktan sonra, iletinin gövdesi salt yazılır olur. Uygulama, .NET için IBytesMessage arabiriminin uygun yazma yöntemlerini çağırarak uygulama verilerini gövde içine toplar. Uygulamanın, byte iletisi akışına bir değer yazdığı anda, değer, uygulama tarafından yazılan önceki değerden hemen sonra birleştirilir. XMS , birleştirilen son baytın konumunu anımsamak için bir iç imleç tutar.

Uygulama iletisi gönderdiğinde, iletinin gövdesi salt okunur hale gelir. Bu modda, uygulama iletisi sürekli olarak gönderebilir.

Bir uygulama byte iletisi aldığı anda, iletinin gövdesi salt okunur olur. Uygulama, byte iletisi akışının içeriğini okumak için IBytesMessage arabiriminin uygun okuma yöntemlerini kullanabilir. Uygulama, sıra içindeki byte 'ları okur ve XMS , okunan son baytın konumunu hatırlamak için bir iç imleç tutar.

Bir uygulama, bir byte iletisinin gövdesi yazılabilir olduğunda IBytesMessage arabiriminin ilk durumuna getirme yöntemini çağırırsa, gövde salt okunur olur. Yöntem ayrıca, imleci byte iletisi akışının başlangıcındaki konumlarını da yeniden konumlandırır.

Bir uygulama, bir byte iletisinin gövdesi salt okunur olduğunda .NET için IMessage arabiriminin Clear Body yöntemini çağırduğunda, gövde yazılabilir hale gelir. Yöntem aynı zamanda vücudu da temizler.

## Eşleme iletileri

Bir eşleme iletisinin gövdesi, her değer için ilişkili bir veri tipine sahip olduğu ad-değer çiftleri kümesini içerir.

Her ad-değer çiftinde ad, değeri tanımlayan bir dizgidir ve değer, [Çizelge 101 sayfa 637](#) içinde listelenen XMS veri tiplerinden birine sahip uygulama verileri ögesidir. Ad-değer çiftlerinin sırası tanımlı değil. MapMessage sınıfı, ad-değer çiftlerini ayarlamaya ve almaya ilişkin yöntemleri içerir.

Bir uygulama, adını belirterek bir ad-değer çiftine rasgele erişebiliyor.

Bir .NET uygulaması, eşleme iletisinin gövdesindeki adlara ilişkin sıralı değer listesi almak için MapNames özelliğini kullanabilir.

Bir uygulama bir ad-değer çiftinin değerini aldığı anda, değer XMS tarafından başka bir veri tipine dönüştürülebilir. Örneğin, bir eşleme iletisinin gövdesinden bir tamsayı almak için, bir uygulama MapMessage sınıfının GetString yöntemini çağırabilir; bu yöntem, tamsayıyı dizgi olarak döndürür. Desteklenen dönüştürmeler, XMS bir özellik değerini bir veri tipinden diğerine dönüştürdüğünde desteklenenler gibi aynıdır. Desteklenen dönüştürmeler hakkında daha fazla bilgi için bkz. [“Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi” sayfa 612.](#)

Bir uygulama, bir eşleme iletisi yarattıktan sonra, iletinin gövdesi okunabilir ve yazılabilir olur. Uygulama iletisi gönderdikten sonra gövde okunabilir ve yazılabilir olarak kalır. Bir uygulama bir eşleme iletisi aldığı anda, iletinin gövdesi salt okunurdur. Bir uygulama, bir eşleme iletisinin gövdesi salt okunur olduğunda, iletisi sınıfının Clear Body (Gövde Temizle) yöntemini çağırduğunda, gövde okunabilir ve yazılabilir olur. Yöntem aynı zamanda vücudu da temizler.

## Nesne iletileri

Bir nesne iletisinin gövdesi serileştirilmiş birJava ya da .NET nesnesi içeriyor.

Bir XMS uygulaması bir nesne iletisi alabilir, üstbilgi alanlarını ve özelliklerini değiştirebilir ve daha sonra başka bir hedefe gönderilebilir. Ayrıca, bir uygulama, bir nesne iletisinin gövdesini kopyalayabilir ve başka bir nesne iletisi oluşturmak için bu iletisi kullanabilir. XMS , bir nesne iletisinin gövdesi bayt dizisi olarak değerlendirir.

Bir uygulama bir nesne iletisi yarattıktan sonra, iletinin gövdesi okunabilir ve yazılabilir olur. Uygulama iletisi gönderdikten sonra gövde okunabilir ve yazılabilir olarak kalır. Bir uygulama bir nesne iletisi aldığı anda, iletinin gövdesi salt okunurdur. Bir uygulama, bir nesne iletisinin gövdesi salt okunur olduğunda .NET için IMessage arabiriminin Clear Body yöntemini çağırduğunda, gövde okunabilir ve yazılabilir hale gelir. Yöntem aynı zamanda vücudu da temizler.

## Akım iletileri

Bir akış iletisinin gövdesi, her değerin ilişkili bir veri tipine sahip olduğu bir değerler akışı içerir.

Bir değerin veri tipi, [Çizelge 101 sayfa 637'](#) de listelenen XMS veri tiplerinden biridir.

Uygulama bir akış iletisi yarattıktan sonra, iletinin gövdesine yazılır. Uygulama, .NET için IStreamMessage arabiriminin uygun yazma yöntemlerini çağırarak uygulama verilerini gövde içine toplar. Uygulamanın ileti akışına her değer yazdığında, değer ve veri tipi, uygulama tarafından yazılan önceki değerden hemen sonra birleştirilir. XMS , birleştirilen son değerin konumunu hatırlamak için bir iç imleç tutar.

Uygulama iletiyi gönderdiğinde, iletinin gövdesi salt okunur hale gelir. Bu modda, uygulama iletiyi birkaç kez gönderebilir.

Bir uygulama bir akış iletisi aldığında, iletinin gövdesi salt okunurdur. Uygulama, ileti akışının içeriğini okumak için, .NET için IStreamMessage arabiriminin uygun okuma yöntemlerini kullanabilir. The application reads the values in sequence, and XMS maintains an internal cursor to remember the position of the last value that was read.

Bir uygulama ileti akışından bir değer okuduğunda, değer XMS tarafından başka bir veri tipine dönüştürülebilir. For example, to read an integer from the message stream, an application can call the ReadString method, which returns the integer as a string. Desteklenen dönüştürmeler, XMS bir özellik değerini bir veri tipinden diğerine dönüştürdüğünde desteklenenler gibi aynıdır. Desteklenen dönüştürmeler hakkında daha fazla bilgi için bkz. [“Bir özellik değerinin bir veri tipinden diğerine örtük olarak dönüştürülmesi” sayfa 612.](#)

Bir uygulama ileti akışından bir değer okumayı denerken hata oluşursa, imleç gelişmiş değildir. Uygulama, değeri başka bir veri tipi olarak okuma girişiminde bulunarak hatadan kurtulabilir.

Bir uygulama, bir akış iletisinin gövdesi salt yazılır olduğunda XMS için IStreamMessage arabiriminin ilk durumuna getirme yöntemini çağırıyorsa, gövde salt okunur olur. Yöntem ayrıca, imleci ileti akışının başlangıcındaki konumlara doğru konumlar.

Bir uygulama, bir akış iletisinin gövdesi salt okunur olduğunda XMS için IMessage arabiriminin Clear Body yöntemini çağırıyorsa, gövde salt yazılır olur. Yöntem aynı zamanda vücudu da temizler.

## Metin iletileri

Metin iletisi gövdesi bir dizgi içeriyor.

Bir uygulama bir metin iletisi yarattıktan sonra, iletinin gövdesi okunabilir ve yazılabilir olur. Uygulama iletiyi gönderdikten sonra gövde okunabilir ve yazılabilir olarak kalır. Bir uygulama bir metin iletisi aldığında, iletinin gövdesi salt okunurdur. Bir uygulama, bir metin iletisinin gövdesi salt okunur olduğunda .NET için IMessage arabiriminin Clear Body yöntemini çağırduğunda, gövde okunabilir ve yazılabilir hale gelir. Yöntem aynı zamanda vücudu da temizler.

## Uygulama verilerine ilişkin öğeler için veri tipleri

To ensure that an XMS application can exchange messages with an IBM MQ classes for JMS application, both the applications must be able to interpret the application data in the body of a message in the same way.

Bu nedenle, XMS uygulaması tarafından bir iletinin gövdesinde yazılan her uygulama verisi öğesinin, [Çizelge 101 sayfa 637](#) içinde listelenen veri tiplerinden birine sahip olması gerekir. Her veri tipi için, tabloda uyumlu Java veri türü gösterilir. XMS , uygulama verilerinin öğelerini yalnızca bu veri tipleriyle yazabilmek için yöntemleri sağlar.

<i>Çizelge 101. Java veri tipleriyle uyumlu XMS veri tipleri</i>		
<b>XMS Veri tipi</b>	<b>Gösterir</b>	<b>Uyumlu Java veri tipi</b>
System.Boolean	Boole değeri true ya da false	boolean
System.Char16	Çift baytlık karakter	DAMGA

Çizelge 101. Java veri tipleriyle uyumlu XMS veri tipleri (devamı var)

XMS Veri tipi	Gösterir	Uyumlu Java veri tipi
System.SByte	İmzalı 8 bitlik tamsayı	Byte
System.Int16	İmzalı 16 bitlik tamsayı	short
System.Int32	İmzalı 32 bitlik tamsayı	int
System.Int64	İşaretli 64-bitlik tamsayı	long
System.Float	İmzalı kayan noktalı sayı	kayan nokta
System.Double	İmzalı çift duyarlıklı kayan noktalı sayı	double
System.String	Karakter dizgisi	Dizgi

Bu veri tiplerinin her birinin boyutu, üst sınırı ve alt sınır değerleri hakkında bilgi için bkz. [“XMS temel tipleri”](#) sayfa 612.

## İleti seçicileri

Bir XMS uygulaması, almak istediği iletileri seçmek için ileti seçicileri kullanır.

Bir uygulama bir ileti tüketicisi yarattığında, bir ileti seçici ifadesini tüketicisiyle ilişkilendirebilir. İleti seçici ifadesi seçim ölçütlerini belirtir.

Bir uygulama IBM WebSphere MQ 7.0 kuyruk yöneticisine bağlanırken, ileti seçimi kuyruk yöneticisi tarafında yapılır. XMS herhangi bir seçim yapmaz ve yalnızca kuyruk yöneticisinden aldığı iletiyi sağlayarak daha iyi performans sağlar.

Bir uygulama, her biri kendi ileti seçici ifadesiyle birden çok ileti tüketicisi yaratabilir. Gelen bir ileti, birden fazla ileti tüketicisi seçim ölçütlerini karşılıyorsa, XMS bu tüketicinin her birine ileti gönderir.

Bir ileti seçici ifadesi, bir iletinin aşağıdaki özelliklerine gönderme yapabilir:

- JMS tanımlı özellikler
- IBM tanımlı özellikler
- Uygulama tanımlı özellikler

Ayrıca, aşağıdaki ileti üstbilgisi alanlarına da gönderme yapabilir:

- JMSCorrelationID
- JMSDeliveryMode
- JMSMessageID
- JMSönceliği
- JMSTimestamp
- JMSType

Ancak, bir ileti seçici ifadesi, bir iletinin gövdesindeki verilere gönderme yapamazlar.

Aşağıda, bir ileti seçici ifadesi örneği yer alıyor:

```
JMSPriority > 3 AND manufacturer = 'Jaguar' AND model in ('xj6','xj12')
```

XMS, bu ileti seçici ifadesiyle bir ileti tüketicisine yalnızca, ileti 3değerinden daha büyük bir önceliğe sahipse; uygulama tanımlı bir özellik, üretici, Jaguar; değerine sahip ve başka bir uygulama tanımlı özellik, model ( xj6 ya da xj12 . değeriyle) içeren bir ileti sağlar.

XMS içinde bir ileti seçici ifadesi oluşturmak için kullanılan sözdizimi kuralları, IBM MQ classes for JMS' de olanlar ile aynıdır. Bir ileti seçici ifadesinin nasıl oluşturulacağı hakkında bilgi için, IBM MQ

ürün belgelerine bakın; bir ileti seçici ifadesinde, JMStanımlı özelliklerin adları JMS adları olmalıdır ve IBM-tanımlı özelliklerin adları IBM MQ classes for JMS adlarıyla olmalıdır. Bir ileti seçici ifadesindeki XMS adlarını kullanamazsınız.

## Mapping XMS messages onto IBM MQ messages

Bir XMS iletisinin JMS üstbilgi alanları ve özellikleri, IBM MQ iletisinin üstbilgi yapılarındaki alanlarla eşlenir.

When an XMS application is connected to an IBM MQ queue manager, messages sent to the queue manager are mapped onto IBM MQ messages in the same way that IBM MQ classes for JMS messages are mapped onto IBM MQ messages in similar circumstances.

Bir Hedef nesnenin `XMLC_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 yapılarındaki alanlarla eşlenir. `XMLC_WMQ_TARGET_CLIENT` özelliğinin bu şekilde ayarlanması, iletiyi alan uygulamanın bir `MQRFH2` üstbilgisini işleyebileceğini varsayar. Alma uygulaması başka bir XMS uygulaması, bir IBM MQ classes for JMS uygulaması ya da bir `MQRFH2` üstbilgisini işlemek için tasarlanmış yerel bir IBM MQ uygulaması olabilir.

Hedef nesnenin `XMLC_WMQ_TARGET_CLIENT` özelliği `XMLC_WMQ_TARGET_DEST_MQ` olarak ayarlandıysa, 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 herhangi bir JMS üstbilgi alanı ve özellikleri yoksa. Bu nedenle, iletiyi alan uygulama, `MQRFH2` üstbilgisini işlemek için tasarlanmamış bir yerel IBM MQ olabilir.

IBM MQ messages received from a queue manager are mapped onto XMS messages in the same way that IBM MQ messages are mapped onto IBM MQ classes for JMS messages in similar circumstances.

Gelen bir IBM MQ iletisinde `MQRFH2` üstbilgisi varsa, sonuçtaki XMS iletisinin tipi, `MQRFH2` üstbilgisinin `mcd` klasöründe bulunan **Msd** özelliğinin değeri tarafından belirlenir. **Msd** özelliği `MQRFH2` üstbilgisinde yoksa ya da IBM MQ iletisinin `MQRFH2` üstbilgisi yoksa, sonuçtaki XMS iletisinin, tipi `MQMD` üstbilgisindeki *Format* alanının değeri tarafından belirlenen bir gövde vardır. *Format* alanı `MQFMT_STRING` olarak ayarlandıysa, XMS iletisi bir metin iletidir. Ters durumda, XMS iletisi bir bayt iletidir. IBM MQ iletisinde `MQRFH2` üstbilgisi yoksa, yalnızca `MQMD` üstbilgisindeki alanlardan türetilen JMS üstbilgi alanları ve özellikleri ayarlanır.

IBM MQ classes for JMS iletilerini IBM MQ iletilerine eşleme hakkında daha fazla bilgi için bkz. [“Mapping JMS messages onto IBM MQ messages” sayfa 132.](#)

## Reading and writing the message descriptor from a IBM Message Service Client for .NET application

Bir IBM MQ iletisinin tüm ileti tanımlayıcısı (`MQMD`) alanlarına, `StrucId` ve `Version`; `BackoutCount` dışındaki tüm ileti tanımlayıcısı (`MQMD`) alanlarına erişebilirsiniz, ancak bu alanlara yazılamaz. Bu özellik yalnızca IBM WebSphere MQ 6.0 ya da sonraki bir kuyruk yöneticisine bağlanılırken kullanılabilir ve daha sonra açıklanan hedef özellikler tarafından denetlendiğinde kullanılabilir.

The message attributes provided by the IBM Message Service Client for .NET facilitates XMS applications to set `MQMD` fields and also to drive IBM WebSphere MQ applications.

Yayınlama/abone olma ileti alışverişi kullanılırken bazı kısıtlamalar geçerli olur. For example, `MQMD` fields like `MsgID` and `CorrelId`, if set, are ignored.

Bu konuda açıklanan işlev, bir IBM WebSphere MQ 6.0 kuyruk yöneticisine bağlanıyorsanız yayınlama/abone olma ileti alışverişi için kullanılamaz. Ayrıca, **PROVIDERVERSION** özelliği 6 olarak ayarlandığında da kullanılamaz.

## Accessing IBM MQ Message data from a IBM Message Service Client for .NET application

You can access the complete IBM MQ message data including the `MQRFH2` header (if present) and any other IBM MQ headers (if present) within a IBM Message Service Client for .NET application as the body of a `JMSBytesMessage`.

Bu konuda açıklanan işlev yalnızca bir IBM WebSphere MQ 7.0 ya da daha sonraki bir kuyruk yöneticisine bağlanılırken kullanılabilir ve IBM MQ ileti alışverişi sağlayıcısı normal kiptedir.

Hedef nesne özellikleri, XMS uygulamasının tüm IBM MQ iletisine (varsa, MQRFH2 üstbilgisi dahil) JMSBytesMessagegövdesine nasıl erişeceğini belirler.

## Component Object Model Interface olanağının kullanılması (ActiveX için IBM MQ Otomasyon Sınıfları)

The IBM MQ Automation Classes for ActiveX (MQAX) are ActiveX components that provide classes that you can use in your application to access IBM MQ.

IBM MQ 9.0' tan Microsoft Active X için IBM MQ desteği kullanımdan kaldırılmıştır. IBM MQ classes for .NET are the recommended replacement technology. Daha fazla bilgi için bkz [Developing .NET applications](#).

MQAX, iletişim kuracağı bir IBM MQ ortamı ve karşılık gelen bir IBM MQ uygulaması gerektirir.

It gives your ActiveX application the ability to run transactions and access data on any of your enterprise systems that you can access through IBM MQ.

ActiveX için IBM MQ Otomasyon Sınıfları:

- Give you access to the functions and features of the IBM MQ API, permitting full interconnectivity to other IBM MQ platforms.
- Bir ActiveX bileşeni için beklenen olağan sözleşmeler ile uyumludur.
- .NET, C + +, Java ve LotusScript için de kullanılabilir olan IBM MQ nesne modeline uyar.

MQAX başlangıç örnekleri sağlanır. Bu örnekleri başlangıçta, MQAX kurulumunuzun başarılı olup olmadığını ve temel IBM MQ ortamınızın yerinde olup olmadığını denetlemek için kullanabilirsiniz. Ayrıca, MQAX 'in nasıl kullanılacağı de gösterir.

### COM ve ActiveX komut dosyası oluşturma

Component Object Model (COM), Microsoft tarafından tanımlanan, nesne tabanlı bir programlama modelidir. Yazılım bileşenlerinin, yazıldığı ya da konularının bulunduğu bilgisayar dilinden bağımsız olarak birbirlerinin yerini belirleyebilmelerini ve birbirleriyle iletişim kurmalarını sağlayan bir şekilde nasıl sağlanabileceğini belirtir.

ActiveX is a set of technologies, based on COM, that integrates application development, reusable components, and Internet technologies on the Microsoft Windows platforms. ActiveX bileşenleri, uygulamalar tarafından dinamik olarak erişilebilecek arabirimler sağlar. ActiveX komut dosyası istemcisi, örneğin, ActiveX (ya da COM) bileşenleri tarafından sağlanan arabirimleri kullanan bir program ya da komut dosyası oluşturabilen ya da yürütebilen bir uygulamadır.

### IBM MQ ortam desteği

IBM MQ Automation Classes for ActiveX can only be used with **32 bit** ActiveX scripting clients.

COM bileşeni yalnızca **32 bit** uygulamalar için kullanılabilir. 64 bit COM uygulaması yazmak istiyorsanız, .NET arabirimini kullanabilirsiniz.

MQAX 'i bir IBM MQ sunucusu ortamında çalıştırmak için sisteminizde Windows 2000 ya da üstü kurulu olmalıdır.

To run the MQAX in an IBM MQ MQI client environment you need IBM MQ MQI client on Windows 2000 or later installed on your system:

IBM MQ MQI client , en az bir IBM MQ sunucusuna erişim gerektirir. Sisteminizde hem IBM MQ MQI client hem de IBM MQ sunucusu kurulu olduğunda, MQAX uygulamaları her zaman sunucuya karşı çalışır. MQAI ' ye ActiveX arabirimi yalnızca IBM MQ sunucu ortamlarında kullanılabilir.



# ActiveX için IBM MQ Otomasyon Sınıflarını kullanarak tasarlama ve programlama

## ActiveX dışı uygulamalara erişen MQAX uygulamalarının tasarlanması

IBM MQ Otomasyon Sınıfları, IBM MQ API ' nin işlevlerine erişim sağlar. Bu nedenle, IBM MQ ' un Windows uygulamanızı getirebileceği tüm avantajlardan yararlanabilirsiniz.

Uygulamanızın genel tasarımı, herhangi bir IBM MQ uygulaması için aynıdır, bu nedenle [“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#) bölümünde açıklanan tüm tasarım yönlerini göz önünde bulundurun.

IBM MQ Otomasyon Sınıflarını kullanmak için, uygulamanıza ilişkin Windows programlarını, COM nesnelere yaratılmasını ve kullanılmasını destekleyen bir dil kullanarak kodlatın. Örneğin, Visual Basic, Java ve diğer ActiveX komut dosyası istemcileri. Daha sonra, gereksinim duyduğunuz IBM MQ nesnelere uygulama dilinin yerel sözdizimini kullanarak kodlanabileceğinden, sınıfların uygulamanıza kolayca bütünleştirilebilmesi gerekir.

## ActiveX için IBM MQ Otomasyon Sınıflarını Kullanma

ActiveX için IBM MQ Otomasyon Sınıflarını kullanan bir ActiveX uygulaması tasarlanırken, bilgilerin en önemli ögesi, uzak IBM MQ sisteminden gönderilen ya da alınan iletidir. Bu nedenle, iletiye eklenen öğelerin biçimini bilmelisiniz. Bir MQAX komut dosyası için, hem bu komut dosyası hem de iletiyi alan ya da gönderen IBM MQ uygulaması, ileti yapısını bilmelidir.

MQAX uygulamasıyla bir ileti gönderiyorsanız ve MQAX uçunda veri dönüştürme işlemi gerçekleştirmek istiyorsanız, şunları bilmeniz gerekir:

- Uzak sistem tarafından kullanılan kod sayfası
- Uzak sistem tarafından kullanılan kodlama

kodunuzu taşınabilir tutmak için, kod sayfası ve kodlamayı ayarlamak iyi bir uygulamadır. şu anda hem gönderme hem de alma sistemlerinde aynı anda aynı olsa bile.

Tasarladığınız sistemin uygulamasını nasıl yapılandıracağını göz önünde bulundurarak, MQAX komut dosyalarınızın, IBM MQ kuyruk yöneticisi ya da IBM MQ istemcisi kurulu olduğu makineden aynı makinede çalıştırıldığını unutmayın.

## Programlama ipuçları ve öneriler

Aşağıdaki ipuçları ve ipuçları önemli bir sırada yer almaz. Bunlar, yaptığınız işe uygun bir şekilde, size zaman kazanmanızı sağlar.

## İleti Tanımlayıcı özellikleri

Bir programdaki ileti tanımlayıcı özelliklerini değiştirdiğinizde, alanların onaltılı eşdeğerlerini kullanmanız daha iyi olabilir.

Bu bölümdeki bilgiler aşağıdaki özelliklere başvurur:

- AccountingToken
- CorrelationId
- GroupId
- MessageId

Where an IBM MQ application is the originator of a message and IBM MQ generates these properties, it is better to use the AccountingTokenHex, CorrelationIdHex, GroupIdHex, and MessageIdHex properties if you want to look at their values, or manipulate them in any way, including passing them back in a message to IBM MQ. Bunun nedeni, IBM MQ tarafından oluşturulan değerlerin, 0 ile 255 arasında herhangi bir değere sahip olan bayt dizgileri olması, bunlar da yazdırılabilir karakterlerin dizgileri olmalarıdır.

MQAX komut dosyanızın bir iletinin kaynağı olduğu durumlarda, AccountingToken, CorrelationId, GroupId ve MessageId özelliklerini ya da Hex eşdeğerlerini kullanabilirsiniz.

## IBM MQ sabitler

IBM MQ constants are provided as members of the enum IBM MQ in library MQAX200.

## IBM MQ dizgi değişmezleri

IBM MQ string constants are not available when using IBM MQ Automation Classes for ActiveX. Aşağıdaki listede gösterilen açık karakter dizisini ve ihtiyacınız olabilecek diğer kişileri kullanmanız gerekir. Komutların boşluk kullanılarak sekiz karaktere kadar doldurulması gerekir:

<i>Çizelge 102. IBM MQ dizgi değişmezleri ve bunlara karşılık gelen karakter dizilimleri.</i>	
<b>Dizgi değişmezi</b>	<b>İlgili karakter dizgisi</b>
MQFMT_NONE	" "
MQFMT_ADMIN	"MQADMIN"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM"
MQFMT_CICS	"MQCICS"
MQFMT_COMMAND_1	"MQCMD1 "
MQFMT_COMMAND_2	"MQCMD2 "
MQFMT_DEAD_LETTER_HEADER	"MQDEAD"
MQFMT_DIST_HEADER	"MQHDIST"
MQFMT_OLAY	"MQOLAY"
MQFMT_IMS	"MQIMS"
MQFMT_IMS_VAR_STRING	"MQIMSVS"
MQFMT_MD_EXTENSION	"MQHMDE"
MQFMT_PCF	"MQPCF"
MQFMT_REF_MSG_HEADER	"MQHREF"
MQFMT_RF_HEADER	"MQHRF"
MQFMT_STRING	"MQSTR"
MQFMT_TETIKLEYICISI	"MQTRIG"
MQFMT_WORK_INFO_HEADER	"MQHIH"
MQFMT_XMIT_Q_HEADER	"MQXMIT"

## Boş dizgi değişmezleri

Dört MQMessage özelliklerinin kullanıma hazırlanması için kullanılan IBM MQ değişmezleri, MQMI\_NONE (24 NULL karakter), MQCI\_NONE (24 NULL karakter), MQGI\_NONE (24 NULL karakter) ve MQACT\_NONE (32 NULL karakter), ActiveX için IBM MQ Otomasyon Sınıfları tarafından desteklenmez. Bunları boş dizgilerle ayarlamak aynı etkiye sahiptir.

Örneğin, bir MQMessage 'in çeşitli tanıtıcılarını şu değerlere ayarlayın: **mymessage.MessageId** = ""  
**mymessage.CorrelationId** = "" **mymessage.AccountingToken** = ""

## IBM MQ' dan ileti alma

IBM MQ' dan bir ileti almanın çeşitli yolları vardır:

- Bir GET işlemi, Visual Basic TIMER işlevini kullanarak, bir GET işlemi yayınlayarak yoklanıyor.
- Issuing a GET with the Wait option; you specify the wait duration by setting the WaitInterval property. Sisteminizi çok iş parçacıklı bir ortamda çalışacak şekilde ayarlamanıza rağmen, o sırada çalışan yazılımların yalnızca tek iş parçacıklı çalışabileceğini göz önünde bulundurun. Bu, süresiz olarak sistem kilitlenmesini önler.

Diğer iş parçacıkları etkilenmeden çalışır. Ancak, diğer iş parçacıklarınız IBM MQ'e erişim gerektiriyorsa, ek MQAX kuyruk yöneticisi ve kuyruk nesnelerini kullanarak IBM MQ ' e ikinci bir bağlantı gerekir.

Bekleme seçeneğiyle GET komutu verilmesi ve WaitInterval ile MQWI\_UNESSININ değerine ayarlanması, işlemin tek iş parçacıklı olması durumunda GET çağrısına kadar sistemin kilitlenmesine neden olur.

## Veri dönüştürmenin kullanılması

Two forms of data conversion are supported by IBM MQ Automation Classes for ActiveX - numeric encoding, and character set conversion.

## Sayısal kodlama

MQMessage Encoding özelliğini ayarlarsanız, farklı sayısal kodlama sistemleri arasında aşağıdaki yöntemler dönüştürülür:

- ReadDecimal2 yöntemi
- ReadDecimal4 yöntemi
- ReadDouble yöntemi
- ReadDouble4 yöntemi
- ReadFloat yöntemi
- ReadInt2 yöntemi
- ReadInt4 yöntemi
- ReadLong yöntemi
- ReadShort yöntemi
- ReadUInt2 yöntemi
- WriteDecimal2 yöntemi
- WriteDecimal4 yöntemi
- WriteDouble yöntemi
- WriteDouble4 yöntemi
- WriteFloat yöntemi
- WriteInt2 yöntemi
- WriteInt4 yöntemi
- WriteLong yöntemi
- WriteShort yöntemi
- WriteUInt2 yöntemi

Kodlama özelliği, sağlanan IBM MQ değişmezleri kullanılarak ayarlanabilir ve yorumlanabilir. [Şekil 60 sayfa 644](#) , şunlara bir örnek gösterir:

```

/* Encodings for Binary Integers */
MQENC_INTEGER_UNDEFINED
MQENC_INTEGER_NORMAL
MQENC_INTEGER_REVERSED

/* Encodings for Decimals */
MQENC_DECIMAL_UNDEFINED
MQENC_DECIMAL_NORMAL
MQENC_DECIMAL_REVERSED

/* Encodings for Floating-Point Numbers */
MQENC_FLOAT_UNDEFINED
MQENC_FLOAT_IEEE_NORMAL
MQENC_FLOAT_IEEE_REVERSED
MQENC_FLOAT_S390

```

### Şekil 60. Kodlama için sağlanan IBM MQ değişmezleri

For example, to send an integer from an Intel system to a System/390 operating system in System/390 encoding:

```

Dim msg As New MQMessage 'Define an IBM MQ message for our use..
Print msg. Encoding      'Currently 546 (or X'222')
                                'Set the encoding property
                                to 785 (or X'311')
msg. Encoding = MQENC_INTEGER_NORMAL OR MQENC_DECIMAL_NORMAL
                                OR MQENC_FLOAT_S390
Print msg. Encoding      'Print it to see the change
Dim local_num As long   'Define a long integer
local_num = 1234        'Set it
msg. WriteLong (local_num) 'Write the number into the message

```

## Karakter kümesi dönüştürmesi

Bir sistemden başka bir sisteme ileti gönderdiğinizde, kod sayfalarının farklı olduğu bir ileti gönderdiğinizde karakter takımı dönüştürme gereklidir. Kod sayfası dönüşümü aşağıdaki gibi kullanılır:

- ReadString yöntemi
- ReadNullTerminatedString yöntemi
- WriteString yöntemi
- WriteNullTerminatedString yöntemi
- MessageData özelliği

MQMessage CharSet özelliğini desteklenen bir karakter kümesi değerine (CCSID) ayarlamalısınız.

ActiveX için IBM MQ Otomasyon Sınıfları, karakter kümesi dönüştürmesi gerçekleştirmek için dönüştürme tablolarını kullanır.

Örneğin, dizgileri otomatik olarak kod sayfası 437 'ye dönüştürmek için:

```

Dim msg As New MQMessage 'Define an IBM MQ message
msg.CharacterSet = 437   'Set code page required
msg.WriteString "A character string" 'Put character string in message

```

WriteString yöntemi, dize verilerini (örnekte A character string) bir Unicode dizesi olarak alır. Daha sonra, 34B001B5.TBL dönüştürme çizelgesini kullanarak bu verileri Unicode 'dan kod sayfası 437 'ye dönüştürür.

Unicode diziliminde, kod sayfası 437 tarafından desteklenmeyen karakterler, kod sayfası 437 'den standart yerine koyma karakteri olarak verilir.

Benzer şekilde, ReadString yöntemini kullandığınızda, gelen iletinin IBM MQ Message Descriptor (MQMD) değeri tarafından kurulmuş bir karakter kümesi vardır ve komut dosyası dilinize geçirilmeden önce bu kod sayfasından Unicode 'a dönüştürme vardır.

## Threading

ActiveX için IBM MQ Otomasyon Sınıfları, nesnelerin iş parçacıkları arasında kullanılabilceği bir serbest threading modelini uygular.

MQAX, MQQueue ve MQQueueManager nesnelerinin kullanılmasına izin vermekle birlikte, IBM MQ şu anda farklı iş parçacıkları arasında çekme noktalarının paylaşılmasına izin vermez.

Bu işlemi başka bir iş parçacığıda kullanma girişimleri hatayla sonuçlanabilir ve IBM MQ , MQRC\_HCONN\_ERROR dönüş kodunu döndürür.

**Not:** İşlem başına yalnızca bir MQSession nesnesi var. Using the MQSession CompletionCode and ReasonCode is not recommended in multithreaded environments. MQSession hata değerlerinin üzerine, ilk iş parçacığıdaki yükseltilmiş ve denetlenmekte olan bir hata arasındaki ikinci bir iş parçacığı tarafından yazılabilir. İş parçacıkları, her bir yöntem çağırısı ya da özellik erişimi süresi için diziselleştirilir. Bu nedenle, Bekleme seçeneği ile Alma seçeneği verilmesi, MQAX nesnelere erişen diğer iş parçacıklarının işlem tamamlanincaya kadar askıya alınmasına neden olur.

## Hata işleme

Bu bilgilerde, MQAX nesnesi özellikleri, hata işleme yöntemi, kural dışı durumların nasıl yükseltildiğini açıklayan kurallar ve bir özellik alınması açıklanır.

Her MQAX nesnesi, hata bilgilerinin tutulacağı özellikleri ve ilk duruma getirme ya da temizleme yöntemini içerir. Özellikler şunlardır:

- CompletionCode
- ReasonCode
- ReasonName

Yöntem şöyledir:

- ClearErrorKodları

## Hata işleme yöntemi

MQAX komut dosyası ya da uygulamanız bir MQAX nesnesinin yöntemini çağırır ya da MQAX nesnesinin bir özelliğine erişir ya da bu nesneye erişir:

1. İlgili nesnelere ReasonCode ve CompletionCode değeri güncellenir.
2. MQSession nesnesindeki ReasonCode ve CompletionCode , aynı bilgilerle güncellenir.

**Not:** İş parçacıklı uygulamalarda MQSession hata kodlarının kullanımına ilişkin kısıtlamalar için [“Threading” sayfa 645 ' e bakın.](#)

CompletionCode , MQSession 'ın ExceptionThreshold özelinden büyük ya da ona eşitse, MQAX kural dışı durum yayınlar (sayı 32000). Bunu işlemek için On Error (ya da eşdeğer) deyimini kullanarak komut dosyanızın içinde kullanın.

3. Bu forma sahip olan ilişkili hata dizgisini almak için Error işlevini kullanın:

```
MQAX: CompletionCode=xxx, ReasonCode=xxx, ReasonName=xxx
```

On Error deyimlerini kullanmaya ilişkin daha fazla bilgi için, ActiveX komut dosyası dilinize ilişkin belgelere bakın.

MQSession nesnesindeki CompletionCode ve ReasonCode kullanılarak, basit hata işleyiciler için uygun bir değer vardır.

ReasonName özelliği, ReasonCode' un geçerli değeri için IBM MQ simgesel adını döndürür.

## Özel durumlar oluşturu

Aşağıdaki kurallar, kural dışı durumların nasıl işlendiğini açıklar:

- Bir özellik ya da yöntem, tamamlanma kodunu, kural dışı durum eşliğinden büyük ya da ona eşit bir değere ayarladığında (genellikle 2 olarak ayarlanır) bir kural dışı durum oluşturulur.
- Tüm yöntem çağruları ve özellik kümeleri tamamlama kodunu ayarlar.

## Özellik alma

CompletionCode ve ReasonCode her zaman güncellenmediği için bu, özel bir olaydır:

- Bir özellik başarılı olursa, nesne ve MQSession nesnesi ReasonCode ve CompletionCode değiştirilmeden kalır.
- If a property get fails with a CompletionCode of warning, the ReasonCode and CompletionCode remain unchanged.
- Bir özellik CompletionCode hata ile başarısız olursa, ReasonCode ve CompletionCode , doğru değerleri yansıtacak şekilde güncellenir ve hata işleme devam edilir olarak işlenir.

The MQSession class has a method *ReasonCodeAd* which might be used to replace an IBM MQ reason code with a symbolic name. Bu, özellikle beklenmedik hataların ortaya çıkabileceği programlar geliştirirken kullanışlıdır. Ancak, ad, kullanıcılara sunum yapmak için ideal değildir.

Her sınıfın, o sınıfa ilişkin geçerli neden kodunun simgesel adını döndüren *ReasonName* özelliği de vardır.

## ActiveX için IBM MQ Otomasyon Sınıfları başvurusu

This section describes the classes of the IBM MQ Automation Classes for ActiveX (MQAX), developed for ActiveX. Sınıflar, IBM MQ kullanarak ActiveX ortamınızda çalışan diğer uygulamalara erişebilen ActiveX uygulamalarını yazmanızı sağlar.

### ActiveX için IBM MQ Otomasyon Sınıfları Arabirimi

ActiveX için IBM MQ Otomasyon Sınıfları, sınıfları kullanmak için gereken önceden tanımlı sayısal ActiveX değişmezlerini (MQMT\_REQUEST gibi) sağlar.

ActiveX otomasyon sınıfları aşağıdaki gibi oluşur:

- [“MQSession Sınıfı” sayfa 648](#)
- [“MQQueueManager sınıfı” sayfa 651](#)
- [“MQQueue sınıfı” sayfa 662](#)
- [“MQMessage sınıfı” sayfa 677](#)
- [“MQPutMessageSeçenekleri sınıfı” sayfa 699](#)
- [“MQGetMessageSeçenekleri sınıfı” sayfa 702](#)
- [“MQDistributionList sınıfı” sayfa 704](#)
- [“MQDistributionListÖğe sınıfı” sayfa 708](#)

Ayrıca, ActiveX için IBM MQ Otomasyon Sınıfları, sınıfları kullanmak için gereken önceden tanımlı sayısal ActiveX değişmezleri (MQMT\_REQUEST gibi) sağlar. Bunlar, MQAX200 kitaplığındaki sıralı değer listesi MQ ' da sağlanır. Sabitler, IBM MQ C üstbilgi dosyalarında (cmqc \*.h), ActiveX Neden kodları için bazı ek IBM MQ Otomasyon Sınıfları ile tanımlanan bir alt kümedir.

### ActiveX sınıfları için IBM MQ Otomasyon Sınıfları Hakkında

Bu bilgileri, [Developing applications reference](#) (Uygulamalar başvurusu) altındaki başvuru konularının yanında okuyun.

See [Features that can be used only with the primary installation on Windows](#) for important information.

MQSession sınıfı, MQAX nesnelere herhangi birinde gerçekleştirilen son işlemin durumunu içeren bir kök nesne sağlar. Ek bilgi için [“Hata işleme” sayfa 645](#) başlıklı konuya bakın.

MQQueueManager ve MQQueue sınıfları, temeldeki IBM MQ nesnelere erişim sağlar. Methods or property accesses for these classes in general result in calls being made across the IBM MQ MQI.

MQMessage, MQPutMessageSeçenekleri ve MQGetMessageSeçenekleri sınıfları, MQMD, MQPMO ve MQGMO veri yapılarını sarmala ve kuyruklara ileti göndermenize ve bunlara ileti göndermenize yardımcı olmak için kullanılır.

MQDistributionList sınıfı, çıkışa ilişkin bir kuyruklar derlemine (yerel, uzak ya da diğer ad) sarsalıyor. MQDistributionListÖğe sınıfı, MQOR, MQRR ve MQPMR yapılarını sarsalıyor ve bunları sahip olan bir dağıtım listesiyle ilişkilendirir.

## Parametre geçişi

Yöntem çağırımlarındaki parametreler, parametrenin bir nesne olduğu durumlar dışında, geçirilen bir başvurudur. Bu durumda, geçirilen bir nesne, bu parametrenin bir nesneden olduğu durumlar dışında,

Belirtilen sınıf tanımlamaları, her değiştirge ya da özellik için Veri Tipi listesini içerir. Visual Basic gibi birçok ActiveX istemcisi için, kullanılan değişken gerekli tipte değilse, değer otomatik olarak gerekli tipten ya da bu tür bir dönüştürmeye dönüştürülmüş olur. İstemcinin standart kuralları şöyledir; MQAX bu tür bir dönüştürmeyi sağlar.

Yöntemlerin çoğu değişmez uzunluklu dizgi değiştirgeleri alır ya da değişmez uzunluklu bir karakter dizilimi döndürür. Dönüştürme kuralları şunlardır:

- Kullanıcı, bir giriş parametresi olarak ya da bir dönüş değeri olarak yanlış uzunluğun değişmez uzunluklu bir dizgisi sağladıysa, değer kısaltılır ya da gerektiği şekilde sondaki boşluklarla doldurulur.
- Kullanıcı, giriş parametresi olarak yanlış uzunluğun değişken uzunluklu bir dizilimini sarf ettiyse, değer kesilerek ya da sondaki boşluklarla doldurulursa doldurulur.
- Kullanıcı, dönüş değeri olarak yanlış uzunluğa ait bir değişken uzunluklu bir dizgi varsa, dizgi gereken uzunluğa göre ayarlanır (çünkü bir değer döndürülürse, dizedeki önceki değeri yok eder).
- Giriş değiştirgeleri olarak sağlanan dizgiler gömülü boş değer içerebilir.

Bu sınıflar, MQAX200 kitaplığında bulunabilir.

## Nesne erişim yöntemleri

Bu yöntemler, doğrudan herhangi bir IBM MQ çağırısını ilişkilendirmez. Bu yöntemlerin her biri, daha sonra, bir IBM MQ nesnesine bağlanılarak ya da açılarak, başvuru bilgilerinin tutulmasındaki bir nesne oluşturur:

When a connection is made to a queue manager, it holds the 'connection handle' attribute generated by IBM MQ.

Bir kuyruk açıldığında, IBM MQ tarafından oluşturulan 'object handle' özniteliğini tutar.

Bu IBM MQ öznitelikleri doğrudan MQAX programının kullanımına sunulmaz.

## Hatalar

Parametre geçişindeki sözdizimsel hatalar, ActiveX istemcisi tarafından derleme sırasında ve çalıştırma sırasında saptlanabilir. Hatalar Visual Basic 'te Error (Hata) ile kapana kısılabılır.

IBM MQ ActiveX sınıflarının tümü, iki özel salt okunur özellik içerir: ReasonCode ve CompletionCode. Bu özellikler her zaman okunabilir.

Başka bir özelliğe erişmeye ya da herhangi bir yöntem çağırısını yayınlamaya ilişkin bir hata IBM MQ' den bir hata oluşturabilir.

Bir özellik kümesi ya da yöntem çağırısı başarılı olursa, sahip nesnenin ReasonCode değeri MQRC\_NONE olarak ayarlanır ve CompletionCode MQCC\_OK olarak ayarlanır.

Özellik erişimi ya da yöntem çağırısı başarılı olamazsa, neden ve tamamlanma kodları bu alanlarda ayarlanır.

## MQSession Sınıfı

Bu, ActiveX için IBM MQ Otomasyon Sınıfları'nın kök sınıfıdır.

Her zaman ActiveX istemci işlemi başına yalnızca bir MQSession nesnesi vardır. İkinci bir nesne yaratma girişimi, özgün nesneye ilişkin ikinci bir başvuru yaratır.

### Yaratma

**Yeni** seçeneği, yeni bir MQSession nesnesi yaratır.

### Sözdizimi

**dim** *mqtaknti* **Yeni Olarak MQSession Küme** *mqtaks* = **Yeni MQSession**

### Özellikler

- [“CompletionCode özelliği” sayfa 648.](#)
- [“ExceptionThreshold özelliği” sayfa 649.](#)
- [“ReasonCode özelliği” sayfa 649.](#)
- [“ReasonName özelliği” sayfa 649.](#)

### Yöntem

- [“AccessGetMessageOptions yöntemi” sayfa 649.](#)
- [“AccessMessage yöntemi” sayfa 650.](#)
- [“AccessPutMessageOptions yöntemi” sayfa 650.](#)
- [“AccessQueueManager yöntemi” sayfa 650.](#)
- [“ClearErrorCodes yöntemi” sayfa 650.](#)
- [“ReasonCodeAd yöntemi” sayfa 650.](#)

### CompletionCode özelliği

Salt okunur. Herhangi bir IBM MQ nesnesi için verilen en son yöntem ya da özellik kümesiyle ayarlanan IBM MQ tamamlanma kodunu döndürür.

Bir yöntem ya da özellik kümesi herhangi bir MQAX nesnesine başarıyla çağrıldığında MQCC\_OK değerine sıfırlanır.

Bir hata olayı işleyicisi, hangi nesnenin dahil olduğunu bilmek zorunda kalmadan hatayı tanılamak için bu özelliği inceleyebilir.

MQSession nesnesindeki CompletionCode ve ReasonCode kullanılarak, basit hata işleyiciler için çok uygun bir durum vardır.

**Not:** İş parçacıklı uygulamalarda MQSession hata kodlarının kullanımına ilişkin kısıtlamalar için [“Threading” sayfa 645](#) ' e bakın.

#### Tanımlandığı yer:

MQSession sınıfı

#### Veri Tipi:

Uzun

#### Değerler:

- MQCC\_OK
- MQCC\_UYARI
- MQCC\_FAILED



**Sözdizimi:**

Almak için: *completioncode & = MQSession.CompletionCode*

**ExceptionThreshold özelliği**

Okuma-yazma. MQAX 'in bir kural dışı durum yayınlayacağı IBM MQ hatası düzeyini tanımlar. Varsayılan olarak MQCC\_FAILED değerine ayarlanır. A value greater than MQCC\_FAILED effectively prevents exception processing, leaving the programmer to perform checks on the CompletionCode and ReasonCode.

**Tanımlı:** MQSession sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Herhangi biri, ancak MQCC\_UYARI, MQCC\_FAILED ya da daha büyük bir değer düşünün.

**Sözdizimi:**

Almak için: *ExceptionThreshold& = MQSession. ExceptionThreshold*

Ayarlamak için: *MQSession. ExceptionThreshold = ExceptionThreshold\$*

**ReasonCode özelliği**

Salt okunur. Herhangi bir IBM MQ nesnesi için verilen en son yöntem ya da özellik kümesiyle ayarlanan neden kodunu döndürür.

Bir hata olayı işleyicisi, hangi nesnenin dahil olduğunu bilmek zorunda kalmadan hatayı tanılamak için bu özelliği inceleyebilir.

MQSession nesnesindeki CompletionCode ve ReasonCode kullanılarak, basit hata işleyiciler için çok uygun bir durum vardır.

**Not:** İş parçacıklı uygulamalarda MQSession hata kodlarının kullanımına ilişkin kısıtlamalar için "Threading" sayfa 645 ' e bakın.

**Tanımlı:** MQSession sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- "Reason codes IBM MQ Automation Classes for ActiveX" sayfa 715 altında listelenen ek MQAX değerleri için bkz. [Neden \(MQlong\)](#) ve ek MQAX değerleri.

**Sözdizimi:** almak için: *reasoncode & = MQSession. ReasonCode*

**ReasonName özelliği**

Salt okunur. En son neden kodunun simgesel adını döndürür. Örneğin, "MQRC\_QMGR\_NOT\_AVAILABLE".

**Not:** İş parçacıklı uygulamalarda MQSession hata kodlarının kullanımına ilişkin kısıtlamalar için "Threading" sayfa 645 ' e bakın.

**Tanımlı:** MQSession sınıfı

**Veri Tipi:** Dize

**Değerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasonname \$= MQSession .ReasonName*

**AccessGetMessageOptions yöntemi**

Yeni bir MQGetMessageSeçenekleri nesnesi yaratır.

**Tanımlandığı yer:**

MQSession sınıfı

**Sözdizimi:**

*gmo* = MQSession .AccessGetMessageOptions()

**AccessMessage yöntemi**

Yeni bir MQMessage nesnesi yaratır.

**Tanımlandığı yer:**

MQSession sınıfı

**Sözdizimi:**

*msg* = MQSession .AccessMessage()

**AccessPutMessageOptions yöntemi**

Yeni bir MQPutMessageSeçenekleri nesnesi yaratır.

**Tanımlandığı yer:**

MQSession sınıfı

**Sözdizimi:**

*pmo* = MQSession .AccessPutMessageOptions()

**AccessQueueManager yöntemi**

Yeni bir MQQueueManager nesnesi yaratır ve bunu, IBM MQ MQI client ya da IBM MQ sunucusu aracılığıyla gerçek bir kuyruk yöneticisine bağlar. Bir bağlanma işlemi gerçekleştirildikçe, bu yöntem kuyruk yöneticisi nesnesi için de bir açma işlemi gerçekleştirir.

Sisteminizde hem IBM MQ MQI client hem de IBM MQ sunucusu kurulu olduğunda, MQAX uygulamaları varsayılan olarak sunucuya karşı çalışır. MQAX 'i istemciye karşı çalıştırmak için, istemci bağ tanımları kitaplığı GMQ\_MQ\_LIB ortam değişkeninde belirtilmeli; örneğin, GMQ\_MQ\_LIB=mqic.dll.

Yalnızca istemci kuruluğu için, GMQ\_MQ\_LIB ortam değişkeninin ayarlanması gerekli değildir. Bu değişken ayarlanmadığında, IBM MQ amqzst.dlldosyasını yükleme girişiminde bulunur. Bu DLL yoksa (vakayı yalnızca istemcinin kuruluşunda olduğu gibi), IBM MQ mqic.dlldosyasını yüklemeyi dener.

Başarılı olursa, MQQueueManager' ın ConnectionStatus değerini TRUE olarak ayarlıyor.

Kuyruk yöneticisi, ActiveX yönetim ortamı başına en çok bir MQQueueManager nesnesine bağlanabilir.

Kuyruk yöneticisine yönelik bağlantı başarısız olursa, bir hata olayı oluşturulur ve MQSession nesnesinin ReasonCode ve CompletionCode ayarı ayarlanır.

**Tanımlı:** MQSession sınıfı

**Sözdizimi:** *set qm* = MQSession .AccessQueueManager ( *Name\$* )

**Parametre:** *Name\$* String. Bağlanılacak Kuyruk Yöneticisinin adı.

**ClearErrorCodes yöntemi**

CompletionCode ögesini MQCC\_OK değerine ve ReasonCode ' ı MQRC\_NONE olarak sıfırlar.

**Tanımlı:** MQSession sınıfı

**Sözdizimi:**

```
Call MQSession.ClearErrorCodes()
```

**ReasonCodeAd yöntemi**

Belirtilen sayısal değer ile neden kodunun adını döndürür. Kullanıcılara hata durumlarına ilişkin daha net göstergeler vermek yararlı olur. Ad hala bir şekilde şifreli (örneğin, ReasonCodeAd (2059)

**MQRC\_Q\_MGR\_NOT\_AVAM**); bu nedenle, olası hataların yakalanması ve uygulamaya uygun açıklayıcı metinle değiştirilmesi gerekir.

**Tanımlı:** MQSession sınıfı

**Sözdizimi:** *errmsg \$= MQSession .ReasonCodeAd ( reasonCode& )*

**Parametre:** *reasoncode* & Long. Simgesel adın gerekli olduğu neden kodu.

## MQQueueManager sınıfı

Bu sınıf, kuyruk yöneticisine yönelik bir bağlantıyı temsil eder. Kuyruk yöneticisi yerel olarak çalıştırılabilir (bir IBM MQ sunucusu) ya da IBM MQ istemcisi tarafından sağlanan erişim ile uzaktan çalıştırılabilir. Bir uygulama bu sınıfın bir nesnesini yaratmalı ve bunu bir kuyruk yöneticisine bağlamalıdır. Bu sınıfın bir nesnesi yok edildiğinde, bu sınıfın bir nesnesi otomatik olarak kuyruk yöneticisinden çıkarılır.

## Bulundurma

MQQueue sınıfı nesnelere bu sınıfla ilişkilendirilir.

Yeni, yeni bir MQQueueManager nesnesi yaratır ve tüm özellikleri ilk değerlerine ayarlar. Diğer bir seçenek olarak, MQSession sınıfının AccessQueueManager yöntemini kullanabilirsiniz.

## Yaratma

Yeni, bir **yeni** MQQueueManager nesnesi yaratır ve tüm özellikleri ilk değerlerine ayarlar. Diğer bir seçenek olarak, MQSession sınıfının AccessQueueManager yöntemini kullanabilirsiniz.

## Sözdizimi

**Dim mgr Yeni MQQueueManager kümesi mgr = Yeni MQQueueManager**

## Özellikler

- [“AlternateUserTanıtıcı özelliği” sayfa 653.](#)
- [“AuthorityEvent özelliği” sayfa 653.](#)
- [“BeginOptions özelliği” sayfa 653.](#)
- [“ChannelAutoTanımlaması özelliği” sayfa 653.](#)
- [“ChannelAutoDefinitionEvent özelliği” sayfa 654.](#)
- [“ChannelAutoDefinitionExit özelliği” sayfa 654.](#)
- [“CharacterSet özelliği” sayfa 654.](#)
- [“CloseOptions özelliği” sayfa 654.](#)
- [“CommandInputQueueName özelliği” sayfa 654.](#)
- [“CommandLevel özelliği” sayfa 654.](#)
- [“CompletionCode özelliği” sayfa 655.](#)
- [“ConnectionHandle özelliği” sayfa 655.](#)
- [“ConnectionStatus özelliği” sayfa 655.](#)
- [“ConnectOptions özelliği” sayfa 655.](#)
- [“DeadLetterQueueName özelliği” sayfa 656.](#)
- [“DefaultTransmissionQueueName özelliği” sayfa 656.](#)
- [“Açıklama özelliği” sayfa 656.](#)
- [“DistributionLists özelliği” sayfa 656.](#)
- [“InhibitEvent özelliği” sayfa 656.](#)

- [“IsConnected özelliđi” sayfa 656.](#)
- [“IsOpen özelliđi” sayfa 657.](#)
- [“LocalEvent özelliđi” sayfa 657.](#)
- [“MaximumHandles özelliđi” sayfa 657.](#)
- [“MaximumMessageUzunluk özelliđi” sayfa 657.](#)
- [“MaximumPriority özelliđi” sayfa 657.](#)
- [“MaximumUncommittedİletileri özelliđi” sayfa 658.](#)
- [“Ad özelliđi” sayfa 658.](#)
- [“ObjectHandle özelliđi” sayfa 658.](#)
- [“PerformanceEvent özelliđi” sayfa 658.](#)
- [“Platform özelliđi” sayfa 658.](#)
- [“ReasonCode özelliđi” sayfa 659.](#)
- [“ReasonName özelliđi” sayfa 659.](#)
- [“RemoteEvent özelliđi” sayfa 659.](#)
- [“StartStopOlay özelliđi” sayfa 659.](#)
- [“SyncPointKullanılabilirlik özelliđi” sayfa 659.](#)
- [“TriggerInterval özelliđi” sayfa 660.](#)

## Yöntemler

- [“AccessQueue yöntemi” sayfa 660.](#)
- [“AddDistributionListe yöntemi” sayfa 660.](#)
- [“Geriletme yöntemi” sayfa 661.](#)
- [“Başlangıç yöntemi” sayfa 661.](#)
- [“ClearErrorCodes yöntemi” sayfa 661.](#)
- [“Kesinleştirme yöntemi” sayfa 661.](#)
- [“Bağlanma yöntemi” sayfa 661.](#)
- [“Bağlantı kesme yöntemi” sayfa 662.](#)

## Özellik Erişimi

Aşağıdaki özelliklere herhangi bir zamanda erişilebilir.

- [“AlternateUserTanıtıcı özelliđi” sayfa 653.](#)
- [“CompletionCode özelliđi” sayfa 655.](#)
- [“ConnectionStatus özelliđi” sayfa 655.](#)
- [“ReasonCode özelliđi” sayfa 659.](#)

Kalan özelliklere yalnızca, nesne bir kuyruk yöneticisine bağlıysa ve kullanıcı kimliđi o kuyruk yöneticisine ilişkin sorgulama için yetkilendirildiyse erişilebilir. Diđer bir kullanıcı kimliđi ayarlandıysa ve yürürlükteki kullanıcı kimliđi bu kimliđi kullanmaya yetkiliyse, diđer kullanıcı kimliđi sorgulamak için yetki olarak denetlenir.

Bu koşullar geçerli deđilse, IBM MQ Automation Classes for ActiveX , kuyruk yöneticisine bağlanmayı dener ve otomatik olarak sorgulanmak üzere açar. Bu işlem başarısız olursa, çağrı bir CompletionCode olarak MQCC\_FAILED ve aşağıdaki ReasonCodes' lerden birini ayarlar:

- MQRC\_CONNECTION\_BROKEN
- MQRC\_NOT\_YETKILI
- MQRC\_Q\_MGR\_NAME\_ERROR

- MQRC\_Q\_MGR\_NOT\_VAR

### ***AlternateUserTanıtıcı özelliği***

Okuma-yazma. Kuyruk yöneticisi özniteliklerine erişimi doğrulamak için kullanılacak diğer kullanıcı kimliği.

IsConnected değeri TRUE ise bu özellik ayarlanmamalıdır.

Nesne açık durumdayken bu özellik ayarlanamaz.

**Defined in:** MQQueueManager sınıfı

**Data Type:** 12 karakterlik bir dizgi

**Syntax:** almak için: *altuser \$= MQQueueManager .AlternateUserId* ayarlamak için: *MQQueueManager .AlternateUserId = altuser \$*

### ***AuthorityEvent özelliği***

Salt okunur. MQI AuthorityEvent özniteliği.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *authevent = MQQueueManager .AuthorityEvent*

### ***BeginOptions özelliği***

Okuma-yazma. Bunlar, Başlat (Begin) yöntemi için geçerli olan seçeneklerdir. Başlangıçta MQBO\_NONE.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

- MQBO\_NONE

**Sözdizimi:** almak için: *beginoptions & =MQQueueManager .BeginOptions*

Ayarlamak için: *MQQueueManager .BeginOptions = beginoptions &*

### ***ChannelAutoTanımlaması özelliği***

Salt okunur. Bu, otomatik kanal tanımlamasına izin verilip verilmediğini denetler.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

- MQCHAD\_ENGELLI
- MQCHAD\_ENABLED

**Sözdizimi:** almak için: *channelautodef & = MQQueueManager .ChannelAutoTanımlaması*

### ***ChannelAutoDefinitionEvent özelliği***

Salt okunur. Bu, otomatik kanal tanımlama olaylarının oluşturulup oluşturulmayacağını denetler.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *channelautodefenvent & =MQQueueManager*. **ChannelAutoDefinitionEvent**

### ***ChannelAutoDefinitionExit özelliği***

Salt okunur. Otomatik kanal tanımlaması için kullanılan kullanıcı çıkışının adı.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Dizgi

**Sözdizimi:** almak için: *channelautodefexit\$=MQQueueManager*. **ChannelAutoDefinitionExit**

### ***CharacterSet özelliği***

Salt okunur. MQI CodedCharSetId özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *characterset & =MQQueueManager*. **CharacterSet**

### ***CloseOptions özelliği***

Okuma-yazma. Kuyruk yöneticisi kapatıldığında ne olacağını denetlemek için kullanılan seçenekler. İlk değer MQCO\_NONE olur.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

- MQCO\_NONE

**Sözdizimi:** almak için: *closeopt & =MQQueueManager*. **CloseOptions**

Ayarlamak için: *MQQueueManager*. **CloseOptions** = *closeopt &*

### ***CommandInputQueueName özelliği***

Salt okunur. MQI CommandInputQName özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *commandinputqname \$=MQQueueManager*. **CommandInputQueueName**

### ***CommandLevel özelliği***

Salt okunur. IBM MQ kuyruk yöneticisi somutlamasının sürümünü ve düzeyini döndürür (MQI CommandLevel özniteliğinin)

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *level & = MQQueueManager .CommandLevel*

### ***CompletionCode özelliği***

Salt okunur. Nesneye yönelik olarak verilen son yöntem ya da özellik erişimi tarafından ayarlanan tamamlanma kodunu döndürür.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQCC\_OK
- MQCC\_UYARI
- MQCC\_FAILED

**Sözdizimi:** almak için: *completioncode & = MQQueueManager .CompletionCode*

### ***ConnectionHandle özelliği***

Salt okunur. IBM MQ kuyruk yöneticisi nesnesine ilişkin bağlantı tanıtıcısı.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Uzun

**Sözdizimi:** almak için: *hconn & = MQQueueManager .ConnectionHandle*

### ***ConnectionStatus özelliği***

Salt okunur. Nesnenin, kuyruk yöneticisine bağlı olup olmadığını gösterir.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Boole

**Değerler:**

- TRUE (-1)
- FALSE (0)

**Sözdizimi:** almak için: *status = MQQueueManager .ConnectionStatus*

### ***ConnectOptions özelliği***

Okuma yazma. Bu seçenekler Connect yöntemi için geçerlidir. Başlangıçta MQCNO\_NONE.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

- MQCNO\_STANDARD\_BINDING
- MQCNO\_FASTPATH\_BINDING
- MQCNO\_NONE

**Sözdizimi:** almak için: *connectoptions & =MQQueueManager .ConnectOptions*

Ayarlamak için: *MQQueueManager .ConnectOptions = connectoptions &*

### ***DeadLetterQueueName özelliği***

Salt okunur. MQI DeadLetterQName özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: *dlqname \$= MQQueueManager .DeadLetterQueueName*

### ***DefaultTransmissionQueueName özelliği***

Salt okunur. MQI DefXmitQName özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *defxmiqname \$= MQQueueManager .DefaultTransmissionQueueName*

### ***Açıklama özelliği***

Salt okunur. MQI QMgrDesc özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** 64 karakter dizgisi

**Sözdizimi:** almak için: *description \$= MQQueueManager .Açıklama*

### ***DistributionLists özelliği***

Salt okunur. Bu, dağıtım listelerini desteklemek için kuyruk yöneticisinin yeteneğidir.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Boole

**Değerler:**

- TRUE (-1)
- FALSE (0)

**Sözdizimi:** almak için: *distributionlist= MQQueueManager .DistributionLists*

### ***InhibitEvent özelliği***

Salt okunur. MQI InhibitEvent özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *inhibevent & = MQQueueManager .InhibitEvent*

### ***IsConnected özelliği***

Kuyruk yöneticisinin şu anda bağlı olup olmadığını gösteren bir değer.

Salt okunur.



**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Boole

**Değerler:**

- TRUE (-1)
- FALSE (0)

**Sözdizimi:** almak için: *isconnected* = *MQQueueManager* .**IsConnected**

### ***IsOpen özelliği***

Kuyruk yöneticisinin sorgu için açık olup olmadığını gösteren bir değer.

Salt okunur.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri Tipi:**

Boole

**Değerler:**

- TRUE (-1)
- FALSE (0)

**Sözdizimi:** almak için: *IsOpen* = *MQQueueManager* .**IsOpen**

### ***LocalEvent özelliği***

Salt okunur. MQI LocalEvent özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *localevent* & = *MQQueueManager* .**LocalEvent**

### ***MaximumHandles özelliği***

Salt okunur. MQI MaxHandles özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *maxhandle* & = *MQQueueManager* .**MaximumHandles**

### ***MaximumMessageUzunluk özelliği***

Salt okunur. MQI MaxMsgLength Kuyruk Yöneticisi özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *maxmessageength* & = *MQQueueManager* .**MaximumMessageLength**

### ***MaximumPriority özelliği***

Salt okunur. MQI MaxPriority özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *maxpriority & = MQQueueManager .MaximumPriority*

### **MaximumUncommittedİletileri özelliği**

Salt okunur. MQI MaxUncommittedMsgs özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *maxuncommittion & = MQQueueManager .MaximumUncommittedİletileri*

### **Ad özelliği**

Okuma-yazma. MQI QMgrName özniteliği. MQQueueManager bağlandıktan sonra bu özellik yazılamaz.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: *name \$= MQQueueManager .name*

Ayarlamak için: *MQQueueManager .name = name \$*

**Not:** Visual Basic, "Name" özelliğini görsel arabirimde kullanılmak üzere ayırır. Bu nedenle, Visual Basic içinde kullanılırken "name" ("ad") adlı küçük harf kullanımı da kullanılır.

### **ObjectHandle özelliği**

Salt okunur. IBM MQ kuyruk yöneticisi nesnesine ilişkin nesne tanıtıcısı.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Veri tipi**

Uzun

**Sözdizimi:** almak için: *hobj & = MQQueueManager. ObjectHandle*

### **PerformanceEvent özelliği**

Salt okunur. MQI PerformanceEvent özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *perfevent & = MQQueueManager.PerformanceEvent*

### **Platform özelliği**

Salt okunur. MQI Platform özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQPL\_WINDOWS\_NT
- MQPL\_WINDOWS

**Sözdizimi:** almak için: *platform & = MQQueueManager .Altyapı*

### ***ReasonCode özelliđi***

Salt okunur. Nesneye karřı verilen son yöntem ya da özellik eriřimiyle belirlenen neden kodunu döndürür.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Deđerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasoncode & = MQQueueManager .ReasonCode*

### ***ReasonName özelliđi***

Salt okunur. En son neden kodunun simgesel adını döndürür. Örneđin, "MQRC\_QMGR\_NOT\_AVALABILIR".

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Dize

**Deđerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasonname \$= MQQueueManager .ReasonName*

### ***RemoteEvent özelliđi***

Salt okunur. MQI RemoteEvent özniteliđi.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Deđerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *remoteevent & = MQQueueManager .RemoteEvent*

### ***StartStopOlay özelliđi***

Salt okunur. MQI StartStopolayı özniteliđi.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Deđerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *strstpevent & = MQQueueManager .StartStopOlayı*

### ***SyncPointKullanılabilirlik özelliđi***

Salt okunur. MQI SyncPoint özniteliđi.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Deđerler:**

- MQSP\_AVALABILIR
- MQSP\_NOT\_VAR

**Sözdizimi:** almak için: `syncpointavailability & = MQQueueManager .SyncPointUygunluğu`

### **TriggerInterval özelliği**

Salt okunur. MQI TriggerInterval özniteliği.

**Tanımlı:** MQQueueManager sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: `trigint & = MQQueueManager .TriggerInterval`

### **AccessQueue yöntemi**

Bir MQQueue nesnesi yaratır ve kuyruğun bağlantı başvurusu özelliğini ayarlayarak bu nesneyi bu MQQueueManager nesnesiyle ilişkilendirir. MQQueue nesnesinin Name (Ad), OpenOptions, DynamicQueueName ve AlternateUser tanıttığı özelliklerini sağlanan değerlere ayarlar ve onu açma girişiminde bulunur.

Açma işlemi başarısız olursa, arama başarısız olur. Nesneye karşı bir hata olayı oluşturulur. Nesnenin ReasonCode ve CompletionCode ve MQSession ReasonCode ve CompletionCode değeri belirlenir.

DynamicQueueName, QueueManagerName ve AlternateUser tanıttığı değıştirmeleri isteğe bağlıdır ve varsayılan değeri "" ' dir.

Kuyruk özellikleri okunacaksa, diğeri seçeneklere ek olarak OpenOption MQOO\_SORGULAMA/ BELIRTILMELI.

Açılacak kuyruk yerelse, QueueManagerAdını ayarlamaz ya da "" değerine ayarlamayın. Ters durumda, kuyruğun sahibi olan uzak kuyruk yöneticisinin adını ve uzak kuyruğun yerel tanımlamasını açma girişiminde bulunmaya çalışılır. Uzak kuyruk adı çözümlenmesi ve kuyruk yöneticisi yöneltme ile ilgili daha fazla bilgi için [Diğeri adlar nedir?](#) başlıklı konuya bakın.

Ad özelliği bir model kuyruğu adı olarak ayarlandıysa, DynamicQueueName\$ parametresinde yaratılacak dinamik kuyruk adını belirtin. DynamicQueueName\$ parametresinde sağlanan değeri "" ise, kuyruk nesnesine ayarlanan ve açık aramada kullanılan değeri "AMQ. \*" olur. Dinamik kuyrukların adlandırılmasıyla ilgili ek bilgi için ["Dinamik kuyruklar oluşturma"](#) sayfa 789 ' e bakın.

## **Tanımlama**

**Tanımlanan:** MQQueueManager sınıfı.

## **Sözdizimi**

**Sözdizimi:** set queue = MQQueueManager. **AccessQueue** (Name\$, OpenOptions&, QueueManagerName\$, DynamicQueueName\$, AlternateUserId\$)

## **Parametreler**

**Name\$** Dize. IBM MQ kuyruğunun adı.

**OpenOptions:** Uzun. Kuyruk açıldığında kullanılacak seçenekler. Bkz. [OpenOptions \(MQUZE\)](#).

**QueueManagerName\$** Dizesi. Açılacak kuyruğun sahibi olan kuyruk yöneticisinin adı. "" değeri, kuyruk yöneticisinin yerel olduğunu belirtir.

**DynamicQueueName\$** Dizesi. Name\$ parametresi bir model kuyruğunu belirtiyorsa, kuyruğun açıldığı sırada dinamik kuyruğa atanan ad.

**AlternateUserId\$** Dize. Kuyruğu açarken erişimi doğrulamak için kullanılan diğeri kullanıcı kimliği.

## **AddDistributionListe yöntemi**

Yeni bir MQDistributionList nesnesi yaratır ve onun bağlantı başvurusunu sahip kuyruk yöneticisine ayarlar.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Sözdizimi:** *set distributionlist = MQQueueManager.AddDistributionListesi*

**Geriletme yöntemi**

Kesinleştirilmemiş herhangi bir iletiyi geri alır ve son eşitleme noktasından bu yana bir iş biriminin bir parçası olarak bu işlemi alır.

**Tanımlı:** MQQueueManager sınıfı

**Sözdizimi:**

```
Call MQQueueManager.Backout()
```

**Başlangıç yöntemi**

Kuyruk yöneticisi tarafından eşgüdümlü bir çalışma birimi başlatır. Başlangıç seçenekleri, bu yöntemin davranışını etkiler.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Sözdizimi:**

```
Call MQQueueManager.Begin()
```

**ClearErrorCodes yöntemi**

CompletionCode ögesini, hem MQQueueManager sınıfı, hem de MQSession sınıfı için MQCC\_NONE ve MQRC\_NONE MQRC\_NONE olarak sıfırlar.

**Tanımlandığı yer:**

MQQueueManager sınıfı

**Sözdizimi:**

**Arama** *MQQueueManager.ClearErrorCodes ()*

**Kesinleştirme yöntemi**

Son eşitleme noktasından bu yana bir iş biriminin bir parçası olarak, herhangi bir ileti koyar ve bunu alır.

**Tanımlı:** MQQueueManager sınıfı

**Sözdizimi:**

```
Call MQQueueManager.Commit()
```

**Bağlanma yöntemi**

MQQueueManager nesnesini IBM MQ MQI client ya da sunucu aracılığıyla gerçek bir kuyruk yöneticisine bağlar. Bu yöntem, bağlantının yanı sıra, kuyruk yöneticisi nesnesini de açar; böylece, bu nesne sorgulanabilir.

IsConnected değerini TRUE olarak ayarlar.

Bir kuyruk yöneticisine bağlanmasına izin verilen ActiveX başına en çok bir MQQueueManager nesnesi kullanılmasına izin verilir.

**Tanımlı:** MQQueueManager sınıfı

## Sözdizimi:

```
Call MQQueueManager.Connect()
```

## Bağlantı kesme yöntemi

Kuyruk yöneticisinden MQQueueManager nesnesinin bağlantısını keser.

IsConnected ögesini FALSE olarak ayarlar.

MQQueueManager nesnesiyle ilişkili tüm kuyruk nesneleri kullanılamaz duruma gelir ve yeniden açılmaz.

Kesinleştirilmemiş değişiklikler (ileti koyar ve alır) kesinleştirilir.

**Tanımlı:** MQQueueManager sınıfı

## Sözdizimi:

```
Call MQQueueManager.Disconnect()
```

## MQQueue sınıfı

Bu sınıf, bir IBM MQ kuyruğuna erişimi temsil eder. Bu bağlantı, ilişkili bir MQQueueManager nesnesi tarafından sağlanıyor. Bu sınıfların bir nesnesi yok edildiğinde otomatik olarak kapatılır.

## Bulundurma

MQQueue sınıfı MQQueueManager sınıfı tarafından içerilir.

## Yaratma

New , yeni bir MQQueue nesnesi yaratır ve tüm özellikleri ilk değerlere ayarlar. Diğer bir seçenek olarak, MQQueueManager sınıfının AccessQueue yöntemini kullanın.

## Sözdizimi

```
Dim que As New MQQueue Set que = New MQQueue
```

## Özellikler

- [“AlternateUserTanıtıcı özelliği” sayfa 665.](#)
- [“BackoutRequeueAd özelliği” sayfa 665.](#)
- [“BackoutThreshold” sayfa 665.](#)
- [“BaseQueueAd özelliği” sayfa 665.](#)
- [“CloseOptions özelliği” sayfa 665.](#)
- [“CompletionCode özelliği” sayfa 666.](#)
- [“ConnectionReference özelliği” sayfa 666.](#)
- [“CreationDateSaat özelliği” sayfa 666.](#)
- [“CurrentDepth özelliği” sayfa 666.](#)
- [“DefaultInputOpenOption özelliği” sayfa 666.](#)
- [“DefaultPersistence özelliği” sayfa 667.](#)
- [“DefaultPriority özelliği” sayfa 667.](#)
- [“DefinitionType özelliği” sayfa 667.](#)

- [“DepthHighOlay özelliği” sayfa 667.](#)
- [“DepthHighSınır özelliği” sayfa 667.](#)
- [“DepthLowOlay özelliği” sayfa 667.](#)
- [“DepthLowÖzelliği sınırla” sayfa 668.](#)
- [“DepthMaximumOlay özelliği” sayfa 668.](#)
- [“DepthHighOlay özelliği” sayfa 667.](#)
- [“DepthHighSınır özelliği” sayfa 667.](#)
- [“DepthLowOlay özelliği” sayfa 667.](#)
- [“DepthLowÖzelliği sınırla” sayfa 668.](#)
- [“DepthMaximumOlay özelliği” sayfa 668.](#)
- [“Açıklama özelliği” sayfa 668.](#)
- [“DynamicQueueAd özelliği” sayfa 668.](#)
- [“HardenGetGeri alma özelliği” sayfa 668.](#)
- [“InhibitGet özelliği” sayfa 669.](#)
- [“InhibitPut özelliği” sayfa 669.](#)
- [“InitiationQueueAd özelliği” sayfa 669.](#)
- [“IsOpen özelliği” sayfa 669.](#)
- [“MaximumDepth özelliği” sayfa 670.](#)
- [“MaximumMessageUzunluk özelliği” sayfa 670.](#)
- [“MessageDeliverySıra özelliği” sayfa 670.](#)
- [“ObjectHandle özelliği” sayfa 670.](#)
- [“OpenInputSayı özelliği” sayfa 670.](#)
- [“OpenOptions özelliği” sayfa 671.](#)
- [“OpenOutputCount özelliği” sayfa 671.](#)
- [“OpenStatus özelliği” sayfa 671.](#)
- [“ProcessName özelliği” sayfa 671.](#)
- [“QueueManagerAd özelliği” sayfa 671.](#)
- [“QueueType Özelliği” sayfa 672.](#)
- [“ReasonCode özelliği” sayfa 672.](#)
- [“ReasonName özelliği” sayfa 672.](#)
- [“RemoteQueueManagerName özelliği” sayfa 672.](#)
- [“RemoteQueueAd özelliği” sayfa 672.](#)
- [“ResolvedQueueManagerName özelliği” sayfa 673.](#)
- [“ResolvedQueueAd özelliği” sayfa 673.](#)
- [“RetentionInterval özelliği” sayfa 673.](#)
- [“Kapsam özelliği” sayfa 673.](#)
- [“ServiceInterval özelliği” sayfa 673.](#)
- [“ServiceIntervalOlay özelliği” sayfa 673.](#)
- [“Paylaşılabilirlik özelliği” sayfa 674.](#)
- [“TransmissionQueueAd özelliği” sayfa 674.](#)
- [“TriggerControl özelliği” sayfa 674.](#)
- [“TriggerData özelliği” sayfa 674.](#)
- [“TriggerDepth özelliği” sayfa 674.](#)

- [“TriggerMessageÖnceliđi özelliđi” sayfa 674.](#)
- [“TriggerType özelliđi” sayfa 675.](#)
- [“Kullanım özelliđi” sayfa 675.](#)

## Yöntemler

- [“ClearErrorCodes yöntemi” sayfa 675](#)
- [“Yöntemi kapat” sayfa 675](#)
- [“Get yöntemi” sayfa 675](#)
- [“Open yöntemi” sayfa 676](#)
- [“Put yöntemi” sayfa 677](#)

## Özellik Erişimi

Kuyruk nesnesi bir kuyruk yöneticisine bağlanmadıysa, aşağıdaki özellikleri okuyabilirsiniz:

- [“CompletionCode özelliđi” sayfa 666](#)
- [“OpenStatus özelliđi” sayfa 671](#)
- [“ReasonCode özelliđi” sayfa 672](#)

ve okuma yazma işlemi yapabilmemiz için aşağıdakileri yapabilirsiniz:

- [“AlternateUserTanıtıcı özelliđi” sayfa 665](#)
- [“CloseOptions özelliđi” sayfa 665](#)
- [“ConnectionReference özelliđi” sayfa 666](#)
- [“Ad özelliđi” sayfa 670](#)
- [“OpenOptions özelliđi” sayfa 671](#)

Kuyruk nesnesi kuyruk yöneticisine bağlıysa, tüm özellikleri okuyabilirsiniz.

## Kuyruk Özniteliđi özellikleri

Önceki bölümde yer alan özellikler, temeldeki IBM MQ kuyruğunun tüm öznitelikleridir. Bunlar yalnızca, nesne bir kuyruk yöneticisine bağlı olduğunda ve kullanıcının kullanıcı kimliđi sorgulamak ya da bu kuyruđa karşı ayarlama için yetkilendirilmiş ise erişilebilir. Başka bir kullanıcı kimliđi belirlendiyse ve yürürlükteki kullanıcı kimliđi bu kimliđi kullanmaya yetkiliyse, diđer kullanıcı kimliđi bunun yerine yetkilendirmek üzere denetlenir.

Özellik, verili QueueType için uygun bir özellik olmalıdır. Ek bilgi için [Kuyruklar için öznitelikler](#) başlıklı konuya bakın.

Bu koşullar geçerli deđilse, özellik erişimi bir CompletionCode olarak MQCC\_FAILED ve aşağıdaki ReasonCodes' lardan biri olarak ayarlanacaktır:

- MQRC\_CONNECTION\_BROKEN
- MQRC\_NOT\_YETKILI
- MQRC\_Q\_MGR\_NAME\_ERROR
- MQRC\_Q\_MGR\_NOT\_CONNECTED
- MQRC\_SELECTOR\_NOT\_FOR\_TYPE (CompletionCode , MQCC\_UYARI olur)

## Kuyruđun Açılması

Bir MQQueue nesnesini yaratmanın tek yolu, MQQueueManager AccessQueue yöntemini ya da Yeni öğesini kullanarak olur. Açık bir MQQueue nesnesi, kapatılıncaya ya da silininceye ya da kuyruk yöneticisi nesnesi silininceye ya da kuyruk yöneticisine bağlantı kayboluncaya kadar açık kalır (OpenStatus= TRUE).



MQQueue CloseOptions özelliğinin değeri, MQQueue nesnesi silindiğinde gerçekleşen kapatma işleminin işleyişini denetler.

The MQQueueManager AccessQueue method opens the queue using the OpenOptions parameter. The MQQueue.Open method opens the queue using the OpenOptions property. IBM MQ , açık kuyruk sürecinin bir parçası olarak kullanıcı yetkilendirmesine karşı OpenOptions ' ı doğrular.

### ***AlternateUserTanıtıcı özelliği***

Okuma-yazma. Açıldığı sırada kuyruğa erişimi doğrulamak için kullanılan diğer kullanıcı kimliği.

Bu özellik, nesne açıkken ayarlanamaz (bu durumda, IsOpen TRUE olduğunda).

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 12 karakter dizgisi

**Sözdizimi:** almak için: *altuser \$= MQQueue .AlternateUserTnt*

Ayarlamak için: *MQQueue. AlternateUserTanıtıcısı = altuser \$*

### ***BackoutRequeueAd özelliği***

Salt okunur. MQI BackOutRequeueQName özniteliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *backoutrequeuead \$= MQQueue .BackoutRequeueAd*

### ***BackoutThreshold***

Salt okunur. MQI BackoutThreshold özniteliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Bkz. [BackoutThreshold \(MQUZE\)](#)

**Sözdizimi:** almak için: *backoutthreshold & = MQQueue. BackoutThreshold*

### ***BaseQueueAd özelliği***

Salt okunur. Diğer adın çözümleyicilerin bulunduğu kuyruk adı.

Yalnızca diğer ad kuyrukları için geçerlidir.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *baseqname \$= MQQueue .BaseQueueName*

### ***CloseOptions özelliği***

Okuma yazma. Kuyruk kapatıldığında ne olacağını denetlemek için kullanılan seçenekler.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQCO\_NONE
- MQCO\_DELETE
- MQCO\_DELETE\_PURGE

MQCO\_DELETE ve MQCO\_DELETE\_PURGE yalnızca dinamik kuyruklar için geçerlidir.

**Sözdizimi:** almak için: *closeopt & = MQQueue .CloseOptions*

Ayarlamak için: *MQQueue .CloseOptions = closeopt &*

### **CompletionCode özelliği**

Salt okunur. Nesneye yönelik olarak verilen son yöntem ya da özellik erişimi tarafından ayarlanan tamamlanma kodunu döndürür.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQCC\_OK
- MQCC\_UYARI
- MQCC\_FAILED

**Sözdizimi:** almak için: *completioncode & = MQQueue .CompletionCode*

### **ConnectionReference özelliği**

Okuma-yazma. Bir kuyruk nesnesinin ait olduğu kuyruk yöneticisi nesnesini tanımlar. Bir kuyruk açıkken bağlantı başvurusu yazılamaz.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** MQQueueManager

**Değerler:**

- Etkin bir IBM MQ Queue Manager nesnesine yönelik başvuru

**Sözdizimi:** Ayarlamak İçin: *set MQQueue .ConnectionReference = ConnectionReference*

Almak için: *set ConnectionReference = MQQueue .ConnectionReference*

### **CreationDateSaat özelliği**

Salt okunur. Bu kuyruğun yaratıldığı tarih ve saat.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Tip 7 (tarih/saat) ya da EMPTY tipinde olan değişken

**Sözdizimi:** almak için: *datetime = MQQueue .CreationDateTime*

### **CurrentDepth özelliği**

Salt okunur. Şu anda kuyruklardaki ileti sayısı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *currentdepth & = MQQueue .CurrentDepth*

### **DefaultInputOpenOption özelliği**

Salt okunur. OpenOptions MQOO\_INPUT\_AS\_Q\_DEF değerini belirtiyorsa, kuyruğun açılacağı yolu denetler.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQOO\_INPUT\_EXCLUSIVE
- MQOO\_INPUT\_SHARED

**Sözdizimi:** almak için: *defaultinop & = MQQueue .DefaultInputOpenOption*

### ***DefaultPersistence özelliği***

Salt okunur. Kuyruklardaki iletiler için varsayılan kalıcılık.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *defpersistence & = MQQueue .DefaultPersistence*

### ***DefaultPriority özelliği***

Salt okunur. Kuyruklardaki iletiler için varsayılan öncelik değeri.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *defpriority & = MQQueue .DefaultPriority*

### ***DefinitionType özelliği***

Salt okunur. Kuyruk tanımlaması tipi.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQQDT\_ÖNCEDEDEN tanımlı
- MQQDT\_PERMANENT\_DYNAMIC
- MQQDT\_TEMPORARY\_DYNAMIC

**Sözdizimi:** almak için: *deftype & = MQQueue .DefinitionType*

### ***DepthHighOlay özelliği***

Salt okunur. MQI QDepthHighOlay özniteliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *depthhighevent & = MQQueue .DepthHighOlayı*

### ***DepthHighSınır özelliği***

Salt okunur. MQI QDepthHighSınır özniteliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *depthhighlimit & = MQQueue .DepthHighSınırı*

### ***DepthLowOlay özelliği***

Salt okunur. MQI QDepthLowOlay özniteliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *depthlowevent & = MQQueue*. **DepthLowOlayı**

### ***DepthLowÖzelliği sınırla***

Salt okunur. MQI QDepthLowSınırı özneliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *depthlowlimit & = MQQueue*. **DepthLowSınırı**

### ***DepthMaximumOlay özelliği***

Salt okunur. MQI QDepthMaxOlay özneliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQEVR\_DISABLE
- MQEVRENABLED

**Sözdizimi:** almak için: *depthmaximevent & = MQQueue*. **DepthMaximumOlayı**

### ***Açıklama özelliği***

Salt okunur. Kuyruğun açıklaması.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 64 karakter dizgisi

**Sözdizimi:** almak için: *description \$= MQQueue* .**Açıklama**

### ***DynamicQueueAd özelliği***

Okuma yazma, kuyruk açık olduğunda salt okunur olur.

Bu, bir model kuyruğu açıldığında kullanılan dinamik kuyruk adını denetler. Kullanıcı bir genel arama karakteri ile bir özellik kümesi (yalnızca kuyruk kapatıldığında) ya da MQQueueManager.AccessQueue() için bir parametre olarak ayarlanabilir.

Dinamik kuyruğun gerçek adı, QueueNamesorgulanarak bulunur.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Değerler:**

- Geçerli bir IBM MQ kuyruğu adı.

**Sözdizimi:** Ayarlamak İçin: *MQQueue* .**DynamicQueueName** = *dynamicqueuename* \$

Almak için: *dynamicqueuename* \$ = *MQQueue* .**DynamicQueueAd**

### ***HardenGetGeri alma özelliği***

Salt okunur. Doğru bir geri sayım sayımının sağlanıp korunmayacağı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQQA\_BACKUT\_HARDENED
- MQQA\_BACKUTUP\_NOT HARDENED

**Sözdizimi:** Alınmak için: *hardengetback & = MQQueue .HardenGetBackout*

### ***InhibitGet özelliği***

Okuma-yazma. MQI InhibitGet özneliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQQA\_GET\_INHIBITED
- MQQA\_GET\_ALLOWD

**Sözdizimi:** Alınmak için: *getstatus & = MQQueue .InhibitGet*

Ayarlamak için: *MQQueue .InhibitGet = getstatus &*

### ***InhibitPut özelliği***

Okuma-yazma. MQI InhibitPut özneliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQQA\_PUT\_INHIBITED
- MQQA\_PUT\_ALLOWD

**Sözdizimi:** almak için: *putstatus & = MQQueue .InhibitPut*

Ayarlamak için: *MQQueue .InhibitPut = putstatus &*

### ***InitiationQueueAd özelliği***

Salt okunur. Başlatma kuyruğunun adı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: *initqname \$= MQQueue .InitiationQueueName*

### ***IsOpen özelliği***

Kuyruğun açık olup olmadığını döndürür.

Salt okunur.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Boole

**Değerler:**

- TRUE (-1)
- FALSE (0)

**Sözdizimi:** almak için: *open = MQQueue .IsOpen*

### **MaximumDepth özelliği**

Salt okunur. Kuyruk derinliği üst sınırı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *maxdepth* & = *MQQueue* .**MaximumDepth**

### **MaximumMessageUzunluk özelliği**

Salt okunur. Bu kuyruk için bayt cinsinden izin verilen ileti uzunluğu üst sınırı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *maxlength* & = *MQQueue* .**MaximumMessageLength**

### **MessageDeliverySıra özelliği**

Salt okunur. Mesaj teslim sırası.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQMDS\_PRIORITY
- MQMDS\_FIFO

**Sözdizimi:** almak için: *messdelseq* & = *MQQueue* .**MessageDeliverySequence**

### **Ad özelliği**

Okuma-yazma. MQI Kuyruğu özneliği. Bu özellik, MQQueue açıldıktan sonra yazılamaz.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: *name* \$= *MQQueue* .**name**

Ayarlamak için: *MQQueue* .**name** = *name* \$

**Not:** Visual Basic, "Name" özelliğini görsel arabirimde kullanılmak üzere ayırır. Bu nedenle, Visual Basic kullanırken "name" (alt harf) kullanan Visual Basic kullanımı kullanılmıssa.

### **ObjectHandle özelliği**

Salt okunur. IBM MQ kuyruk nesnesi için nesne tanıtıcısı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *hobj* & = *MQQueue* .**ObjectHandle**

### **OpenInputSayı özelliği**

Salt okunur. Giriş için açılan açma sayısı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için:

```
openincount& = MQQueue.OpenInputCount
```

### ***OpenOptions özelliği***

Okuma-yazma. Kuyruğu açmak için kullanılacak seçenekler.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Bkz. [OpenOptions \(MQUZE\)](#).

**Sözdizimi:** almak için:

```
openopt& = MQQueue.OpenOptions
```

Ayarlamak için: `MQQueue.OpenOptions = openopt &`

### ***OpenOutputCount özelliği***

Salt okunur. Çıkış için açılan açma sayısı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için:

```
openoutcount& = MQQueue.OpenOutputCount
```

### ***OpenStatus özelliği***

Salt okunur. Kuyruğun açılıp açılmayacağı edilmediğini belirtir. İlk değer, AccessQueue yönteminden sonra TRUE ya da New 'den sonra FALSE değerini verir.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Boole

**Değerler:**

- TRUE (-1)
- FALSE (0)

**Sözdizimi:** almak için:

```
status& = MQQueue.OpenStatus
```

### ***ProcessName özelliği***

Salt okunur. MQI ProcessName özniteliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: `procname $ = MQQueue.ProcessName`

### ***QueueManagerAd özelliği***

Okuma-yazma. IBM MQ kuyruk yöneticisi adı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Dize

**Sözdizimi:** almak için: *QueueManagerName\$* = *MQQueue* .**QueueManagerName**

Ayarlamak için: *MQQueue* .**QueueManagerName** = *QueueManagerName\$*

### ***QueueType Özelliği***

Salt okunur. MQI QType özniteliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQQT\_ALIAS
- MQQT\_LOCAL
- MQQT\_MODEL
- MQQT\_REMOTE

**Sözdizimi:** almak için: *queuetype* & = *MQQueue* .**QueueType**

### ***ReasonCode özelliği***

Salt okunur. Nesneye karşı verilen son yöntem ya da özellik erişimiyle belirlenen neden kodunu döndürür.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasoncode* & = *MQQueue* .**ReasonCode**

### ***ReasonName özelliği***

Salt okunur. En son neden kodunun simgesel adını döndürür. Örneğin, "MQRC\_QMGR\_NOT\_AVAILABLE".

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Dize

**Değerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasonname \$* = *MQQueue* .**ReasonName**

### ***RemoteQueueManagerName özelliği***

Salt okunur. Uzak kuyruk yöneticisinin adı. Yalnızca uzak kuyruklar için geçerlidir.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *remqmanname \$* = *MQQueue* .**RemoteQueueManagerName**

### ***RemoteQueueAd özelliği***

Salt okunur. Uzak kuyruk yöneticisinde bulunduğu gibi, kuyruğun adı. Yalnızca uzak kuyruklar için geçerlidir.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize



**Sözdizimi:** Almak İçin: *remqname \$= MQQueue .RemoteQueueName*

### ***ResolvedQueueManagerName özelliği***

Salt okunur. Yerel kuyruk yöneticisiyle bilinen son hedef kuyruk yöneticisinin adı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *resqmanname \$= MQQueue .ResolvedQueueManagerName*

### ***ResolvedQueueAd özelliği***

Salt okunur. Yerel kuyruk yöneticisiyle bilinen son hedef kuyruğunun adı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *resqname \$= MQQueue .ResolvedQueueAdı*

### ***RetentionInterval özelliği***

Salt okunur. Kuyruğun alıkonması gereken süre.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *retinterval & = MQQueue .RetentionInterval*

### ***Kapsam özelliği***

Salt okunur. Bu kuyruğa ilişkin bir girişin bir hücre dizininde de bulunup bulunmadığını denetler.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQSCO\_Q\_MGR
- MQSCO\_CEL

**Sözdizimi:** almak için: *scope & = MQQueue .Scope*

### ***ServiceInterval özelliği***

Salt okunur. MQI QServiceInterval özneliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *serviceinterval & = MQQueue. ServiceInterval*

### ***ServiceIntervalOlay özelliği***

Salt okunur. MQI QServiceIntervalOlay özneliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQQSI\_YükSEK
- MQQSI\_OK
- MQQSI\_NONE

**Sözdizimi:** almak için: *serviceintervalevent & = MQQueue*. **ServiceIntervalOlayı**

### ***Paylaşılabilirlik özelliği***

Salt okunur. Kuyruk paylaşılabilirliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQQA\_SHAREABLE
- MQQA\_NOT\_SHAREABLE

**Sözdizimi:** almak için: *shareubility & = MQQueue*. **Shareability**

### ***TransmissionQueueAd özelliği***

Salt okunur. İletim kuyruğu adı. Yalnızca uzak kuyruklar için geçerlidir.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *transqname \$= MQQueue*. **TransmissionQueueAdı**

### ***TriggerControl özelliği***

Okuma-yazma. Tetik kontrolü.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQTC\_OFF
- MQTC\_ON

**Sözdizimi:** Almak İçin: *trigcontrol & = MQQueue*. **TriggerControl**

Ayarlamak için: *MQQueue*. **TriggerControl** = *trigcontrol &*

### ***TriggerData özelliği***

Okuma-yazma. Verileri tetikler.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** 64 karakter dizgisi

**Sözdizimi:** Almak İçin: *trigdata \$= MQQueue*. **TriggerData**

Ayarlamak için: *MQQueue*. **TriggerData** = *trigdata \$*

### ***TriggerDepth özelliği***

Okuma-yazma. Bir tetikleme iletisi yazılmadan önce kuyruğun üzerinde olması gereken ileti sayısı.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** Almak İçin: *trigdepth & = MQQueue*. **TriggerDepth**

Ayarlamak için: *MQQueue*. **TriggerDepth** = *trigdepth &*

### ***TriggerMessageÖnceliği özelliği***

Okuma-yazma. Tetikleyiciler için eşik iletisi önceliği.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** Almak İçin: *trigmesspriority & = MQQueue .TriggerMessageÖnceliği*

Ayarlamak için: *MQQueue .TriggerMessagePriority = trigmesspriority &*

### ***TriggerType özelliği***

Okuma-yazma. Tetikleyici tipi.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQTT\_NONE
- MQTT\_BIRINCI
- MQTT\_EVERY
- MQTT\_DERINLIK

**Sözdizimi:** Almak İçin: *trigtype & = MQQueue .TriggerType*

Ayarlamak için: *MQQueue .TriggerType = Trigtype &*

### ***Kullanım özelliği***

Salt okunur. Kuyruğun ne için kullanıldığını gösterir.

**Tanımlı:** MQQueue sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQUS\_NORMAL
- MQUS\_ILETIMI

**Sözdizimi:** almak için: *usage & = MQQueue .Kullanım*

### ***ClearErrorCodes yöntemi***

CompletionCode 'u MQCC\_OK ve ReasonCode ' yi hem MQQueue sınıfı, hem de MQSession sınıfı için MQRC\_NONE olarak sıfırlar.

**Tanımlı:** MQQueue sınıfı

**Sözdizimi:**

```
Call MQQueue.ClearErrorCodes()
```

### ***Yöntemi kapat***

CloseOptions' un geçerli değerlerini kullanarak bir kuyruğu kapatır.

**Tanımlı:** MQQueue sınıfı

**Sözdizimi:**

```
Call MQQueue.Close()
```

### ***Get yöntemi***

Kuyruktan bir ileti alır.

Bu yöntem, nesne olarak bir MQMessage nesnesini, nesnenin MQMD 'deki bazı alanları giriş deęiřtirgeleri olarak kullanarak alır. Özellikle, MessageId ve CorrelId alanları kullanılır; bu nedenle, bu alanların gerektięi şekilde ayarlandığından emin olun. Bu alanlarla ilgili daha fazla bilgi için bkz. [MsgId \(MQBYTE24\)](#) ve [CorrelId \(MQBYTE24\)](#).

Yöntem başarısız olursa, MQMessage nesnesi deęiřmeden kalır. Başarılı olursa, MQMessage nesnesinin MQMD ve Message Data kısımlarının yerini, gelen iletiden MQMD ve İleti Verileri alır. MQMessage denetim özellikleri ařaęıdaki gibi ayarlandı:

- **MessageLength** , IBM MQ iletisinin uzunluęuna ayarlıdır.
- **DataLength** , IBM MQ iletisinin uzunluęuna ayarlıdır.
- **DataOffset** sıfır olarak ayarlandı

#### Tanımlandığı yer:

MQQueue sınıfı

#### Sözdizimi:

```
Call MQQueue.Get(Message, GetMsgOptions, GetMsgLength)
```

#### Parametreler

İleti:

Alınacak iletiyi gösteren MQMessage nesnesi.

GetMsgSeçenekleri:

Alma işlemini denetlemek için isteęe baęlı MQGetMessageSeçenekleri nesnesi. Bu parametre belirlenmezse, varsayılan MQGetMessageSeçenekleri kullanılır.

GetMsgUzunluęu:

Kuyruktan alınan IBM MQ iletisinin uzunluk üst sınırını denetlemek için isteęe baęlı 2 ya da 4 baytlık uzunluk deęeri.

MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneęi belirtilirse, GET işleminin MQCC\_UYARI tamamlanma koduyla başarılı olur ve ileti büyüklüęü belirtilen uzunluęu ařarsa MQRC\_TRUNCATED\_MSG\_ACCEPTED neden kodu kabul edilir.

MessageData , ilk GetMsgbayt veri byte 'ı içerir.

MQGMO\_ACCEPT\_TRUNCATED\_MSG **is not** belirtilirse ve ileti büyüklüęü belirtilen uzunluęu ařarsa, MQCC\_FAILED işleminin tamamlanma kodu bir arada MQRC\_TRUNCATED\_MESSAGE\_FAILED dönüş koduyla birlikte başarısız oldu.

İleti arabelleęindeki içerik tanımsız olursa, toplam ileti uzunluęu, alınan iletinin tam uzunluęuna ayarlanır.

İleti uzunluęu parametresi belirlenmezse, ileti arabelleęindeki uzunluk, gelen iletinin en az büyüklüęüne göre otomatik olarak ayarlanır.

#### Open yöntemi

řu anki deęerleri kullanarak bir kuyruk açar:

1. QueueName
2. QueueManagerAdı
3. AlternateUserTanıtıcısı
4. DynamicQueueAdı

#### Tanımlandığı yer:

MQQueue sınıfı

## Sözdizimi:

```
Call MQQueue.Open()
```

## Put yöntemi

Kuyruğa ileti yerleştirir.

Bu yöntem, bir MQMessage nesnesini değiştirge olarak alır. Bu nesnenin İleti Tanımlayıcısı (MQMD) özellikleri bu yöntemin bir sonucu olarak değiştirilebilir. Bu yöntem çalıştırıldıktan hemen sonra sahip oldukları değerler, IBM MQ' a konulan değerlerdir.

Koyma işlemi tamamlandıktan sonra MQMessage nesnesinde yapılan değişiklikler, IBM MQ kuyruğunda gerçek iletiyi etkilemez.

### Tanımlandığı yer:

MQQueue sınıfı

## Sözdizimi:

```
Call MQQueue.Put(Message, PutMsgOptions)
```

## Parametreler

İleti

Konmak için iletiyi gösteren MQMessage nesnesi.

PutMsgSeçenekleri

Koyma işlemi denetleyen seçenekleri içerenMQPutMessageSeçenekleri nesnesi. Bu seçenek belirlenmezse, varsayılan PutMessage(PutMessage) seçenekleri kullanılır.

## MQMessage sınıfı

Bu sınıf bir IBM MQ iletisini gösterir. IBM MQ iletisi tanımlayıcısını (MQMD) kapsüllemek için özellikler içerir ve uygulama tarafından tanımlanan ileti verilerini tutmak için bir arabellek sağlar.

Sınıf, bir ActiveX uygulamasından bir MQMessage nesnesine veri kopyalamak için Yazma yöntemlerini içerir. Benzer şekilde, sınıf bir MQMessage nesnesindeki verileri bir ActiveX uygulamasına kopyalamak için Okuma yöntemlerini içerir. Sınıf, arabelleğe otomatik olarak ayrılan bellek ayırma ve serbest bırakma işlemlerini yönetir. Bir MQMessage nesnesi yaratıldığında arabellek büyüklüğünü bildirmek zorunda değildir; arabellek bu nesneye yazılan verileri barındıracak şekilde büyür.

Arabellek büyüklüğü, o kuyruğun MaximumMessageLength özelliğini aşarsa, bir iletiyi IBM MQ kuyruğuna yerleştiremezsiniz.

Oluşturulduktan sonra, bir MQMessage nesnesi, MQQueue.Put yöntemini kullanarak IBM MQ kuyruğuna konabilir. Bu yöntem, nesnenin MQMD ' nin ve ileti veri bölümlerinin bir kopyasını alır ve kuyruğun üzerine kopyalayan yerlerini alır. The application can therefore modify or delete an MQMessage object after the Put, without affecting the message on the IBM MQ queue. The queue manager can adjust some of the fields in the MQMD when it copies the message on the IBM MQ queue.

Gelen bir ileti, MQQueue.Get yöntemini kullanarak bir MQMessage nesnesine okunabilir. Bu işlem, gelen iletiden değerlerle MQMessage nesnesinde önceden bulunan MQMD ya da ileti verilerinin yerini alır. Bu, MQMessage nesnesinin veri arabelleğindeki büyüklüğünü, gelen ileti verilerinin büyüklüğünün eşleştirmek üzere ayarlar.

## Bulundurma

İletiler MQSession sınıfı tarafından içerilir.

## Yaratma

**Yeni** , bir MQMessage nesnesi yaratır. İleti tanımlayıcı özellikleri başlangıçta varsayılan değerlere ayarlanır ve İleti Verileri arabelleği boş olur.

## Sözdizimi

```
Dim msg As New MQMessage
```

ya da

```
Set msg = New MQMessage
```

## Özellikler

Denetim özellikleri şunlardır:

- [“CompletionCode özelliği” sayfa 680](#)
- [“DataLength özelliği” sayfa 680](#)
- [“DataOffset özelliği” sayfa 681](#)
- [“MessageLength özelliği” sayfa 681](#)
- [“ReasonCode özelliği” sayfa 681](#)
- [“ReasonName özelliği” sayfa 681](#)

İleti tanımlayıcı özellikleri şunlardır:

- [“AccountingToken özelliği” sayfa 682](#)
- [“AccountingTokenHex özelliği” sayfa 682](#)
- [“ApplicationIdData özelliği” sayfa 682](#)
- [“ApplicationOriginVeri özelliği” sayfa 682](#)
- [“BackoutCount özelliği” sayfa 682](#)
- [“CharacterSet özelliği” sayfa 683](#)
- [“CorrelationId özelliği” sayfa 683](#)
- [“CorrelationIdHex özelliği” sayfa 683](#)
- [“Kodlama özelliği” sayfa 684](#)
- [“Süre bitimi özelliği” sayfa 684](#)
- [“Feedback özelliği” sayfa 685](#)
- [“Biçim özelliği” sayfa 685](#)
- [“GroupId özelliği” sayfa 685](#)
- [“GroupIdHex özelliği” sayfa 685](#)
- [“MessageData özelliği” sayfa 686](#)
- [“MessageFlags özelliği” sayfa 686](#)
- [“MessageId özelliği” sayfa 686](#)
- [“MessageIdHex özelliği” sayfa 686](#)
- [“MessageSequenceSayı özelliği” sayfa 687](#)
- [“MessageType özelliği” sayfa 687](#)
- [“Görelî konum özelliği” sayfa 687](#)
- [“OriginalLength özelliği” sayfa 687](#)
- [“Kalıcılık özelliği” sayfa 688](#)

- [“Öncelik özelliği” sayfa 688](#)
- [“PutApplicationAd özelliği” sayfa 688](#)
- [“PutApplicationTip özelliği” sayfa 688](#)
- [“PutDateSaat özelliği” sayfa 689](#)
- [“ReplyToQueueManagerAd özelliği” sayfa 689](#)
- [“ReplyToQueueName özelliği” sayfa 689](#)
- [“Rapor özelliği” sayfa 689](#)
- [“TotalMessageUzunluk özelliği” sayfa 689](#)
- [“UserId özelliği” sayfa 690](#)

## **Yöntemler**

- [“ClearErrorCodes yöntemi” sayfa 690](#)
- [“ClearMessage yöntemi” sayfa 690](#)
- [“Okuma yöntemi” sayfa 690](#)
- [“ReadBoolean yöntemi” sayfa 690](#)
- [“ReadByte yöntemi” sayfa 691](#)
- [“ReadDecimal2 yöntemi” sayfa 691](#)
- [“ReadDecimal4 yöntemi” sayfa 691](#)
- [“ReadDouble yöntemi” sayfa 691](#)
- [“ReadDouble4 yöntemi” sayfa 691](#)
- [“ReadFloat yöntemi” sayfa 692](#)
- [“ReadInt2 yöntemi” sayfa 692](#)
- [“ReadInt4 yöntemi” sayfa 692](#)
- [“ReadLong yöntemi” sayfa 692](#)
- [“ReadNullTerminatedString yöntemi” sayfa 692](#)
- [“ReadShort yöntemi” sayfa 693](#)
- [“ReadString yöntemi” sayfa 693](#)
- [“ReadUInt2 yöntemi” sayfa 693](#)
- [“ReadUnsignedByte yöntemi” sayfa 694](#)
- [“ReadUTF yöntemi” sayfa 694](#)
- [“ResizeBuffer yöntemi” sayfa 694](#)
- [“Yazma yöntemi” sayfa 695](#)
- [“WriteBoolean yöntemi” sayfa 695](#)
- [“WriteByte yöntemi” sayfa 695](#)
- [“WriteDecimal2 yöntemi” sayfa 695](#)
- [“WriteDecimal4 yöntemi” sayfa 696](#)
- [“WriteDouble yöntemi” sayfa 696](#)
- [“WriteDouble4 yöntemi” sayfa 696](#)
- [“WriteFloat yöntemi” sayfa 697](#)
- [“WriteInt2 yöntemi” sayfa 697](#)
- [“WriteInt4 yöntemi” sayfa 697](#)
- [“WriteLong yöntemi” sayfa 697](#)
- [“WriteNullTerminatedString yöntemi” sayfa 698](#)

- [“WriteShort yöntemi” sayfa 698](#)
- [“WriteString yöntemi” sayfa 698](#)
- [“WriteUInt2 yöntemi” sayfa 698](#)
- [“WriteUnsignedByte yöntemi” sayfa 699](#)
- [“WriteUTF yöntemi” sayfa 699](#)

## Özellik erişimi

Tüm özellikler her zaman okunabilir.

Denetim özellikleri, okuma-yazma olan DataOffset dışında salt okunurdur. İleti tanımlayıcı özellikleri, yalnızca okunur olan BackoutCount ve TotalMessageUzunluğu dışında, tüm okuma-yazma özellikleridir.

Ancak, ileti bir IBM MQ kuyruğuna konduğunda MQMD özelliklerinin kuyruk yöneticisi tarafından değiştirilebileceğini göz önünde bulundurun. Bunların nasıl değiştirilebileceğiyle ilgili ayrıntılar için [MQMD](#) içindeki alanlara bakın.

## Veri dönüştürme

**CharacterSet** özelliğini, kuyruk yöneticisinin (MQCCSI\_Q\_MGR) Kodlanmış Karakter Kümesi Tanıtıcısı ile eşleşecek şekilde ayarlayarak ve verileri bir dizgi olarak ileterek ikili verileri bir IBM MQ iletisine geçirebilirsiniz. Dizginin Unicode ya da ASCII kod numaralarını içermesi gerekiyorsa, chr işlevini kullanarak bunları dizgi biçimine dönüştürebilirsiniz.

Okuma ve Yazma yöntemleri, veri dönüştürmeyi gerçekleştirir. They convert between the ActiveX internal formats, and the IBM MQ message formats as defined by the Encoding and CharacterSet properties from the message descriptor. Bir ileti yazarken, bir Yazma yöntemi yayınlamadan önce iletinin alıcısının karakteristiğiyle eşleşen Kodlamaya ve CharacterSet ' e değer ayarlayın. Bir ileti okunurken, bu değerler, gelen MQMD ' de bu değerler belirlendiği için olağan durumda zorunlu değildir.

Bu, MQQueue.Get yöntemi tarafından gerçekleştirilen dönüştürmenin ardından gerçekleşen ek bir veri dönüştürme adımdır.

## CompletionCode özelliği

Salt okunur. Bu nesneye yönelik olarak verilen en son yöntem ya da özellik erişimi ile ayarlanan IBM MQ tamamlanma kodunu döndürür.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- MQCC\_OK
- MQCC\_UYARI
- MQCC\_FAILED

**Sözdizimi:** almak için: *completioncode* & = MQMessage .**CompletionCode**

## DataLength özelliği

Salt okunur. Bu özellik şu değeri döndürür:

```
MQMessage.MessageLength - MQMessage.DataOffset
```

Bir Okuma yönteminden önce, arabellekte beklenen karakter sayısının gerçekte var olup olmadığını denetlemek için kullanılabilir.

Başlangıç değeri sıfır.

**Tanımlı:** MQMessage sınıfı



**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *bytessol & = MQMessage .DataLength*

### **DataOffset özelliği**

Okuma-yazma. İleti nesnesinin İleti Verileri bölümü içindeki yürürlükteki konum.

Değer, ileti veri arabelleğinin başlangıcındaki byte göreli konumu olarak ifade edilir; arabelleğindeki ilk karakter, sıfır değerinin DataOffset değerine karşılık gelir.

A read or write method commences its operation at the character referenced by DataOffset. Bu yöntemler, arabelleğindeki verileri bu konumdan başlayarak sırayla işler ve son işlenen son işlemde hemen sonra (varsa) byte 'ı (varsa) işaret edecek şekilde DataOffset güncellemesine bakın.

DataOffset , yalnızca sıfır ile MessageLength aralığındaki değerleri kapsayılabilir. DataOffset = MessageLength sona doğru gösterdiğinde, arabelleğin ilk geçersiz karakteridir. Bu durumda yazma yöntemlerine izin verilir; arabelleğdeki verileri genişletir ve MessageLength değerini, eklenen bayt sayısını artırır. Arabelleğin sona ermesinin ötesinde okuma değeri geçerli değil.

Başlangıç değeri sıfır.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *curcpos & = MQMessage .DataOffset*

Ayarlamak için: *MQMessage .DataOffset = curcupos &*

### **MessageLength özelliği**

Salt okunur. DataOffsetdeğerinden bağımsız olarak, karakter cinsinden ileti nesnesinin İleti Verileri bölümünün toplam uzunluğunu döndürür.

Başlangıç değeri sıfır. Bu ileti nesnesine gönderme yapan bir alma yöntemi çağrısından sonra gelen İleti Uzunluğu olarak ayarlanır. Uygulama nesneye veri eklemek için bir Yazma yöntemi kullanıyorsa, bu değer artırılır. Okuma yöntemlerinden etkilenmez.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** Almak İçin: *msglength & = MQMessage .MessageLength*

### **ReasonCode özelliği**

Salt okunur. Bu nesneye yönelik olarak verilen en son yöntem ya da özellik erişimi tarafından ayarlanan neden kodunu döndürür.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasoncode & = MQMessage .ReasonCode*

### **ReasonName özelliği**

Salt okunur. En son neden kodunun simgesel adını döndürür. Örneğin, "MQRC\_QMGR\_NOT\_AVAILABLE".

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Dize

**Değerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasonname \$= MQMessage .ReasonName*

### ***AccountingToken özelliği***

Okuma-yazma. MQMD AccountingToken -İleti Kimliği Bağlamı 'nın bir parçası.

Başlangıç değeri tüm boş değerlerde olur.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 32 karakterlik dize

**Sözdizimi:** almak için: *actoken \$= MQMessage .AccountingToken*

Ayarlamak için: *MQMessage .AccountingToken = actoken \$*

AccountingToken özelliği yerine AccountingTokenHex 'i kullanmanız gerektiği zaman hakkında daha fazla bilgi için [“İleti Tanımlayıcısı özellikleri” sayfa 641](#) konusuna bakın.

### ***AccountingTokenHex özelliği***

Okuma-yazma. MQMD AccountingToken -İleti Kimliği Bağlamı 'nın bir parçası.

Her iki karakter, tek bir ASCII karakterinin onaltılık değerini gösterir. Örneğin, "6" ve "1" karakter çifti, tek karakteri "A", "6" ve "2" karakter çifti tek karakteri "B", vb. temsil eder.

64 geçerli onaltılı karakter sağlamalısınız.

Başlangıç değeri: "0 ... 0"

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 32 ASCII karakteri simgeleyen 64 onaltılı karakterden oluşan dize

**Sözdizimi:** almak için: *actokenh \$= MQMessage .AccountingTokenHex*

Ayarlamak için: *MQMessage .AccountingTokenHex = actokenh \$*

AccountingToken özelliği yerine AccountingTokenHex 'i kullanmanız gerektiği zaman hakkında daha fazla bilgi için [“İleti Tanımlayıcısı özellikleri” sayfa 641](#) konusuna bakın.

### ***ApplicationIdData özelliği***

Okuma-yazma. MQMD ApplIdentityData-Message Identity Bağlamı 'nın bir parçası.

Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 32 karakterlik dize

**Sözdizimi:** almak için: *applid \$= MQMessage .ApplicationIdData*

Ayarlamak için: *MQMessage .ApplicationIdData = applid \$*

### ***ApplicationOriginVeri özelliği***

Okuma-yazma. MQMD ApplOriginVeri kısmı, ileti kaynağı bağlamının bir parçası.

Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 4 karakter dizgisi

**Sözdizimi:** almak için: *applor \$= MQMessage .ApplicationOriginVerileri*

Ayarlamak için: *MQMessage .ApplicationOriginData = applor \$*

### ***BackoutCount özelliği***

Salt okunur. MQMD BackoutCount.

Başlangıç değeri 0 olur.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *backoutct & = MQMessage .BackoutCount*

### **CharacterSet özelliği**

Okuma-yazma. MQMD CodedCharSetId.

Its initial value is the special value **MQCCSI\_Q\_MGR**.

CharacterSet seçeneği **MQCCSI\_Q\_MGR** olarak ayarlandıysa, WriteString yönteminde, yürürlükteki ülke değerine ilişkin kod sayfası karakter dönüştürmesi için kullanılır. Sunucu uygulamaları için, kullanılan kod sayfası kuyruk yöneticisinin kod sayfasıdır. İstemci uygulamaları için, varsayılan yürürlükteki ülke değeri kod sayfasıdır.

Örneğin:

```
msg.CharacterSet = MQCCSI_Q_MGR
msg.WriteString(chr$(n))
```

burada ' n', sıfırdan büyük ya da sıfıra eşit ve 255 'e eşit ya da daha küçük, arabelleğe yazılacak tek bir değer byte 'ında sonuçlar elde edilir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** Almak İçin: *:30ccid& = MQMessage .CharacterSet*

Ayarlamak için: *MQMessage .CharacterSet = ccid &*

### **Örnek**

Kod sayfası 437 'de yazılan dizginin yazılmasını istiyorsanız, sorun:

```
Message.CharacterSet = 437
Message.WriteString ("string to be written")
```

Set the value you want in the CharacterSet before issuing any WriteString calls.

### **CorrelationId özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQMD ' ye dahil edilmesi için CorrelationId . Ayrıca, kuyruktan ileti alınırken eşleştirilecek tanıtıcı.

Başlangıç değeri boş değerli.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 24 karakter dizgisi

**Sözdizimi:** almak için: *correlid \$= MQMessage .CorrelationId : MQMessage .CorrelationId = correlid \$.*

CorrelationId özelliği yerine CorrelationIdHex 'i kullanmanız gerektiği zaman hakkında daha fazla bilgi için [“İleti Tanımlayıcısı özellikleri” sayfa 641](#) konusuna bakın.

### **CorrelationIdHex özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQMD ' ye dahil edilmesi için CorrelationId . Ayrıca, bir kuyruktan ileti alınırken eşleştirilecek CorrelationId ' nin de eşleşmesi gerekir.

Dizilimin her iki karakteri, tek bir ASCII karakterinin onaltılı değerini gösterir. Örneğin, "6" ve "1" karakter çifti, tek karakteri "A", "6" ve "2" karakter çifti tek karakteri "B", vb. temsil eder.

48 geçerli onaltılı karakter sağlamalısınız.

Başlangıç değeri: "0 ... 0".

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 24 ASCII karakteri temsil eden 48 onaltılı karakterden oluşan dize

**Sözdizimi:** almak için: *correlidh \$= MQMessage .CorrelationIdHex*

Ayarlamak için: *MQMessage .CorrelationIdHex = correlidh \$*

CorrelationId özelliği yerine CorrelationIdHex 'i kullanmanız gerektiği zaman tartışması için bkz. "[İleti Tanımlayıcısı özellikleri](#)" sayfa 641 .

### ***Kodlama özelliği***

Okuma-yazma. Uygulama iletisi verilerinde sayısal değerler için kullanılan gösterimi tanımlayan MQMD alanı.

Başlangıç değeri, platforma göre değişen özel değer MQENC\_NATIVE değeridir.

Bu özellik aşağıdaki yöntemler tarafından kullanılır:

- ReadDecimal2 yöntemi
- ReadDecimal4 yöntemi
- ReadDouble yöntemi
- ReadDouble4 yöntemi
- ReadFloat yöntemi
- ReadInt2 yöntemi
- ReadInt4 yöntemi
- ReadLong yöntemi
- ReadShort yöntemi
- ReadUInt2 yöntemi
- WriteDecimal2 yöntemi
- WriteDecimal4 yöntemi
- WriteDouble yöntemi
- WriteDouble4 yöntemi
- WriteFloat yöntemi
- WriteInt2 yöntemi
- WriteInt4 yöntemi
- WriteLong yöntemi
- WriteShort yöntemi
- WriteUInt2 yöntemi

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *encoding & = MQMessage .Kodlama : MQMessage ögesini ayarlayın.Kodlama = encoding &*

İleti arabelleğiyle veri yazmaya hazırlanıyorsanız, alan kuyruk yöneticisi kendi veri dönüştürmesini gerçekleştiremeyecekse, bu alanı alan kuyruk yöneticisi altyapısının özellikleriyle eşleştirecek şekilde ayarlamalısınız.

### ***Süre bitimi özelliği***

Okuma-yazma. MQMD süre bitimi zaman alanı, saniyenin onda biri olarak beklenir.

Başlangıç değeri, MQE\_UNSNISIT özel değeridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *expiry & = MQMessage .Süre Bitimi*

Ayarlamak için: *MQMessage .Süre Bitimi = süre bitimi ve*

### **Feedback özelliği**

Okuma-yazma. MQMD geribildirim alanı.

İlk değeri, MQFB\_NONE özel değeri.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Bkz. [Geribildirim](#).

**Sözdizimi:** almak için: *feedback & = MQMessage .Geribildirim*

Ayarlamak için: *MQMessage .Geribildirim = geri bildirim ve*

### **Biçim özelliği**

Okuma-yazma. MQMD biçimi alanı. İleti verilerinin niteliyi tanımlayan yerleşik ya da kullanıcı tanımlı bir biçimin adını verir.

İlk değeri, MQFMT\_NONE özel değeridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 8 karakter dizgisi

**Sözdizimi:** almak için: *format \$= MQMessage .Biçim*

Ayarlamak için: *MQMessage .Biçim = biçim \$*

### **GroupId özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQPMR ' ye eklenmesi için GroupId . Ayrıca, kuyruktan ileti alınırken eşleştirilecek tanıtıcı. Başlangıç değeri tüm boş değerlerde olur.

**Tanımlandığı yer:**

MQMessage sınıfı

**Veri Tipi:**

24 karakter dizgisi

**Sözdizimi:** almak için: *groupid \$= MQMessage. GroupId*

Ayarlamak için: *MQMessage. GroupId = groupid \$*

GroupId özelliğinin yerine GroupIdHex 'i kullanmanız gerektiği zaman hakkında daha fazla bilgi için [“İleti Tanımlayıcısı özellikleri”](#) sayfa 641 konusuna bakın.

### **GroupIdHex özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQPMR ' ye eklenmesi için GroupId . Ayrıca, kuyruktan ileti alınırken eşleştirilecek tanıtıcı.

Dizilimin her iki karakteri, tek bir ASCII karakterinin onaltılı değerini gösterir. Örneğin, "6" ve "1" karakter çifti, tek karakteri "A", "6" ve "2" karakter çiftini temsil eder ve tek karakteri "B" vb. gösterir.

48 geçerli onaltılı karakter sağlamalısınız.

Başlangıç değeri: "0 ... 0".

**Tanımlandığı yer:**

MQMessage sınıfı

**Veri Tipi:**

24 ASCII karakteri simgeleyen 48 onaltılı karakterden oluşan dizgi.

**Sözdizimi:** almak için: *groupidh \$= MQMessage. GroupIdHex*

Ayarlamak için: *MQMessage. GroupIdHex = groupidh \$*

GroupId özelliğinin yerine GroupIdHex 'i kullanmanız gerektiği zaman hakkında daha fazla bilgi için [“İleti Tanımlayıcısı özellikleri” sayfa 641](#) konusuna bakın.

**MessageData özelliği**

Okuma-yazma. Bir iletinin tüm içeriğini karakter dizgisi olarak alır ya da ayarlar.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Varyant

**Not:** Bu özellik tarafından kullanılan veri tipi Variant, ancak MQAX bunun bir dizgi çeşitleme tipi olmasını bekliyor. Bu tipten başka bir çeşitleme geçerseniz, MQRC\_OBJECT\_TYPE\_ERROR hatası döndürülür.

**Sözdizimi:** Almak İçin: *String\$ = MQMessage .MessageData*

Ayarlamak için: *MQMessage .MessageData = String\$*

**MessageFlags özelliği**

Okuma yazma. Kesimlere ayırma denetim bilgileri belirten ileti işaretleri. Başlangıç değeri 0 olur.

**Tanımlandığı yer:**

MQMessage sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [MsgFlags \(MQHOT\)](#).

**Sözdizimi:** almak için: *messageflags & = MQMessage. MessageFlags*

Ayarlamak için: *MQMessage. MessageFlags = messageflags &*

**MessageId özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQMD ' ye dahil edilmesi için MessageId . Ayrıca, kuyruktan ileti alınırken eşleştirilecek tanıtıcı.

Başlangıç değeri tüm boş değerlerde olur.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 24 karakter dizgisi

**Sözdizimi:** almak için: *messageid \$= MQMessage .MessageId*

Ayarlamak için: *MQMessage .MessageId = messageid \$*

MessageId özelliği yerine MessageIdHex 'i kullanmanız gerektiği zaman hakkında daha fazla bilgi için [“İleti Tanımlayıcısı özellikleri” sayfa 641](#) konusuna bakın.

**MessageIdHex özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQMD ' ye dahil edilmesi için MessageId . Ayrıca, bir kuyruktan ileti alınırken eşleştirilecek MessageId ' dir.

Dizilimin her iki karakteri, tek bir ASCII karakterinin onaltılı değerini gösterir. Örneğin, "6" ve "1" karakter çifti, tek karakteri "A", "6" ve "2" karakter çifti tek karakteri "B", vb. temsil eder.

48 geçerli onaltılı karakter sağlamalısınız.

Başlangıç değeri: "0 ... 0".

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 24 ASCII karakteri temsil eden 48 onaltılı karakterden oluşan dize

**Sözdizimi:** almak için: *messageidh \$ = MQMessage .MessageIdHex*

Ayarlamak için: *MQMessage .MessageIdHex = messageidh \$*

MessageId özelliği yerine MessageIdHex 'i kullanmanız gerektiği zaman hakkında daha fazla bilgi için "[İleti Tanımlayıcısı özellikleri](#)" sayfa 641 konusuna bakın.

### **MessageSequenceSayı özelliği**

Okuma yazma. Grup içindeki bir iletiyi tanımlayan sıra bilgileri. Başlangıç değeri 1 'dir.

**Tanımlandığı yer:**

MQMessage sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [MsgSeqNumber \(MQUZE\)](#).

**Sözdizimi:** almak için: *sequencenumber & = MQMessage .SequenceNumber*

Ayarlamak için: *MQMessage .SequenceNumber = sequencenumber &*

### **MessageType özelliği**

Okuma-yazma. MQMD MsgType alanı.

İlk değeri MQMT\_DATAGRAM ' tır.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Bkz. [MsgType \(MQUZE\)](#).

**Sözdizimi:** almak için: *msgtype & = MQMessage .MessageType*

Ayarlamak için: *MQMessage .MessageType = msgtype &*

### **Görelî konum özelliği**

Okuma yazma. Bölümlenmiş bir iletteki görelî konum. Başlangıç değeri 0 olur.

**Tanımlandığı yer:**

MQMessage sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [Görelî Konum \(MQUZE\)](#).

**Sözdizimi:** almak için: *offset & = MQMessage .Görelî Konum*

Ayarlamak için: *MQMessage .Kayma = görelî konum ve*

### **OriginalLength özelliği**

Okuma yazma. Kesimlere ayrılmış bir iletinin özgün uzunluğu. İlk değer MQOL\_UNDEFINED değeridir.

**Tanımlandığı yer:**

MQMessage sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [OriginalLength](#) (MQlong).

**Sözdizimi:** almak için: *originallength & = MQMessage. OriginalLength*

Ayarlamak için: *MQMessage. OriginalLength = originallength &*

**Kalıcılık özelliği**

Okuma-yazma. İletinin devamlılığı ayarı.

Başlangıç değeri MQPER\_PERSISTENCE\_AS\_Q\_DEF 'dir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *persist & = MQMessage .Kalıcılık*

Ayarlamak için: *MQMessage .Kalıcılık = kalıcı saklama &*

**Öncelik özelliği**

Okuma-yazma. Mesajın önceliği var.

Başlangıç değeri, MQPRIO\_PRIORITY\_AS\_Q\_DEF özel değeridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *priority & = MQMessage .Öncelik*

Ayarlamak için: *MQMessage .Öncelik = priority &*

**PutApplicationAd özelliği**

Okuma-yazma. İleti kökeni bağlamının MQMD PutApplAdı-kısmı.

Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 28 karakterden oluşan dize

**Sözdizimi:** Almak İçin: *putapplnm \$= MQMessage .PutApplicationAdı*

Ayarlamak için: *MQMessage .PutApplicationName = putapplnm \$*

**PutApplicationTip özelliği**

Okuma-yazma. İleti kökeni bağlamının MQMD PutAppltipi-kısmı.

Başlangıç değeri MQAT\_NO\_CONTEXT ' dir

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Değerler:**

• Bkz. [PutApplTip](#) (MQUZE).

**Sözdizimi:** Almak İçin: *putappltp & = MQMessage .PutApplicationType*

Ayarlamak için: *MQMessage .PutApplicationType = putappltp &*



### **PutDateSaat özelliği**

Okuma/yazma. Bu özellik, MQMD PutDate ve PutTime alanlarını birleştirir. Bunlar ileti kökeni bağlamının, iletinin ne zaman konduğunu gösteren kısımlarıdır.

ActiveX Extension, ActiveX tarih/saat biçimi ile IBM MQ MQMD ' de kullanılan Tarih ve Saat biçimlerini dönüştürür. Geçersiz bir PutDate ya da PutTime(PutTime) içeren bir ileti alındıysa, get yöntemi EMPTY olarak ayarlandıktan sonra PutDateTime özelliği.

İlk değeri BOŞ (EMPTY).

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Tip 7 (tarih/saat) ya da EMPTY tipinde.

**Sözdizimi:** almak için: *datetime* = MQMessage .PutDateTime

Ayarlamak için: MQMessage .PutDateTime = *datetime*

### **ReplyToQueueManagerAd özelliği**

Okuma-yazma. MQMD ReplyToQMgr alanı.

Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: *replytoqmgr \$* = MQMessage .ReplyToQueueManagerName

Ayarlamak için: MQMessage .ReplyToQueueManagerName = *replytoqmgr \$*

### **ReplyToQueueName özelliği**

Okuma-yazma. MQMD ReplyToQ alanı.

Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: *replytoq \$* = MQMessage .ReplyToQueueName

Ayarlamak için: MQMessage .ReplyToQueueName = *replytoq \$*

### **Rapor özelliği**

Okuma-yazma. İletinin Rapor seçenekleri.

İlk değeri MQRO\_NONE olur.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Bkz. [Rapor](#).

**Sözdizimi:** almak için: *report &* = MQMessage .Rapor

Ayarlamak için: MQMessage .Rapor = *rapor ve*

### **TotalMessageUzunluk özelliği**

Salt okunur. MQGET tarafından alınan son iletinin uzunluğunu alır. İleti kısaltılmamışsa, bu değer MessageLength özelliğinin değerine eşittir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *totalmessabelength & = MQMessage .TotalMessageLength*

### ***UserId özelliği***

Okuma-yazma. MQMD UserIdentifier -Message Identity Context (İleti Tanıtıcısı Bağlamı) kısmı.

Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQMessage sınıfı

**Veri Tipi:** 12 karakter dizgisi

**Sözdizimi:** almak için: *userid \$= MQMessage .UserId*

Ayarlamak için: *MQMessage .UserId = userid \$*

### ***ClearErrorCodes yöntemi***

CompletionCode 'u MQCC\_OK ve ReasonCode ' yi hem MQMessage sınıfı, hem de MQSession sınıfı için MQRC\_NONE olarak sıfırlar.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.ClearErrorCodes()
```

### ***ClearMessage yöntemi***

Bu yöntem, MQMessage nesnesinin veri arabelleği b" lmesini temizler. Veri arabelleğindeki herhangi bir ileti verisi kaybedilir; çünkü MessageLength, DataLengthve DataOffset her şey sıfır olarak ayarlanır.

Message Descriptor (MQMD) kısmı etkilenmez; MQMessage nesnesini yeniden kullanmadan önce bir uygulamanın MQMD alanlarının bazılarını değiştirmesi gerekebilir. MQMD alanlarını ayarlamak için nesneyi yeni bir eşgörünüme sahip yeni bir eşgörünüme yenisiyle değiştirmek için Yeni düğmesini kullanın.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.ClearMessage()
```

### ***Okuma yöntemi***

İleti arabelleğinden bayt dizisine kadar bir bayt dizisi okur. DataOffset artırılır ve Veri Uzunluğu okunan bayt sayısı kadar azaltılır.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:** Data = MQMessage. **Okuyun** (len &)

**Parametreler:**

*len &:* Uzun. Okunabilmek için bayt cinsinden veri uzunluğu.

### ***ReadBoolean yöntemi***

İleti arabelleğindeki yürürlükteki konumdan 1 byte 'lık bir Boole değeri okur ve 2 baytlık bir Boole TRUE (-1) /YANLIŞ (0) değerini döndürür. DataOffset , bir birim tarafından artırılır ve Veri Uzunluğu bir birim tarafından azaltılır.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:** *value = MQMessage. ReadBoolean*

## ***ReadByte yöntemi***

DataOffset tarafından gönderme yapılan byte ile başlayarak, ReadByte yöntemi İleti Verileri arabelleğinden 1 byte 'ı okur ve -128-127 aralığında bir Tamsayı (imli 2-bayt) tamsayı değeri olarak döndürür.

The method fails if MQMessage.DataLength is less than 1 when it is issued.

Yöntem başarılı olursa, DataOffset , 1 artırılır ve DataLength değeri 1 ile azaltılır. Yöntem başarılı olursa, 1 değerini artırılır.

İleti verilerinin byte 'ı imzalı bir ikili tamsayı olduğu varsayılır.

### **Tanımlandığı yer:**

MQMessage sınıfı

### **Sözdizimi:**

*integerv%* = MQMessage .ReadByte

## ***ReadDecimal2 yöntemi***

2 baytlık paketlenmiş onlu sayıyı okur ve imzalanmış 2 baytlık bir tamsayı değeri olarak döndürür. DataOffset , iki birim artırılır ve Veri Uzunluğu iki tarafından azaltılır.

### **Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:** *value%* = MQMessage .ReadDecimal2

## ***ReadDecimal4 yöntemi***

4 baytlık paketlenmiş bir ondalık sayıyı okur ve imzalanmış 4 baytlık bir tamsayı değeri olarak döndürür. DataOffset , dört artırılır ve Veri Uzunluğu dört tarafından azaltılır.

### **Tanımlandığı yer:**

MQMessage sınıfı

### **Sözdizimi:**

```
Call value& = MQMessage .ReadDecimal4
```

## ***ReadDouble yöntemi***

DataOffset tarafından gönderme yapılan byte ile başlayarak, ReadDouble yöntemi İleti Verileri arabelleğinden 8 byte 'ı okur ve bunları çift (imli 8 byte) kayan nokta değeri olarak döndürür.

The method fails if MQMessage.DataLength is less than 8 when it is issued.

Yöntem başarılı olursa, DataOffset , 8 artırılır ve DataLength değeri 8 tarafından azaltılır.

İleti verilerinin 8 karakterinin ikili kayar noktalı sayı olduğu varsayılır. Kodlama, MQMessage.Encoding özelliği tarafından belirtilir. System/360 biçiminden dönüştürmenin desteklenmediğini unutmayın.

### **Tanımlandığı yer:**

MQMessage sınıfı

### **Sözdizimi:**

*doublev#* = MQMessage .ReadDouble

## ***ReadDouble4 yöntemi***

ReadDouble4 ve WriteDouble4 yöntemleri, ReadFloat ve WriteFloat yöntemlerine alternatiflerdir. Bunun nedeni, 4 baytlık IEEE kayar noktalı biçim biçimine dönüştürülebilme için çok büyük olan 4 baytlık System/390 kayan noktalı ileti değerlerini destekledikleri için.

DataOffset tarafından başvuru bayt ile başlayarak, ReadDouble4 yöntemi İleti Verileri arabelleğinden 4 bayt okur ve bunları çift (imli 8 bayt) kayan nokta değeri olarak döndürür.

MQMessage.DataLength komutu verildiğinde 4 değerinden küçükse yöntem başarısız olur.

Yöntem başarılı olursa, DataOffset , 4 artırılır ve DataLength değeri 4 tarafından azaltılır.

İleti verilerinin 4 karakterinin bir ikili kayar noktalı sayı olduğu varsayılır. Kodlama, MQMessage.Encoding özelliği tarafından belirtilir. System/360 biçiminden dönüştürmenin desteklenmediğini unutmayın.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

*doublev# = MQMessage.ReadDouble4*

**ReadFloat yöntemi**

DataOffset tarafından başvuru bayt ile başlayarak, ReadFloat yöntemi İleti Verileri arabelleğinden 4 bayt okur ve bunları, Tek (4 baytlık imzalı) bir kayan nokta değeri olarak döndürür.

MQMessage.DataLength komutu verildiğinde 4 değerinden küçükse yöntem başarısız olur.

Yöntem başarılı olursa, DataOffset , 4 artırılır ve DataLength değeri 4 tarafından azaltılır.

İleti verilerinin 4 karakterinin kayan noktalı bir sayı olduğu varsayılır. Kodlama, MQMessage.Encoding özelliği tarafından belirtilir. System/360 biçiminden dönüştürmenin desteklenmediğini unutmayın.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

*singlev! = MQMessage.ReadFloat*

**ReadInt2 yöntemi**

Yöntem, ReadShort yöntemi ile aynıdır.

**Sözdizimi:**

*integerv% = MQMessage.ReadInt2*

**ReadInt4 yöntemi**

Bu yöntem ReadLong yöntemi ile aynıdır.

**Sözdizimi:**

*bigint & = MQMessage.ReadInt4*

**ReadLong yöntemi**

DataOffset tarafından başvuru bayt ile başlayarak, ReadLong yöntemi İleti Verileri arabelleğinden 4 bayt okur ve bunları, Uzun (işaretli 4 baytlık) bir tamsayı değeri olarak döndürür.

MQMessage.DataLength komutu verildiğinde 4 değerinden küçükse yöntem başarısız olur.

Yöntem başarılı olursa, DataOffset , 4 artırılır ve DataLength değeri 4 tarafından azaltılır.

İleti verilerinin 4 karakterinin ikili bir tamsayı olduğu varsayılır. Kodlama, MQMessage.Encoding özelliği tarafından belirtilir.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

*bigint & = MQMessage.ReadLong*

**ReadNullTerminatedString yöntemi**

Bu yöntem, dizgi gömülü boş karakterler içerebiliyorsa, ReadString yerine kullanılır.

This method reads the specified number of bytes from the message data buffer starting with the byte referred to by DataOffset and returns it as an ActiveX string. Dizilim, sondan önce gömülü bir boş değer içeriyorsa, döndürülen dizginin uzunluğu, boş değer olmadan önce yalnızca bu karakterleri yansıtacak şekilde azaltılır.

DataOffset is incremented and DataLength is decremented by the value specified regardless of whether the string contains embedded null characters.

The characters in the message data are assumed to be a string in the code page that is specified by the MQMessage.CharacterSet property. Uygulama için ActiveX gösterimine dönüştürme gerçekleştirilir.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:** *string \$ = MQMessage.ReadNullTerminatedString(uzunluk &)*

**Parametreler:**

uzunluk & *Uzun*. Bayt cinsinden dizgi alanı uzunluğu.

### **ReadShort yöntemi**

DataOffsettarafından başvuru olan bayt ile başlayarak, ReadShort yöntemi İleti Verileri arabelleğinden 2 bayt okur ve bunları bir Tamsayı (imli 2-bayt) değeri olarak döndürür.

The method fails if MQMessage.DataLength is less than 2 when it is issued.

Yöntem başarılı olursa,DataOffset , 2 artırılır ve DataLength değeri 2 tarafından azaltılır.

İleti verilerinin 2 karakterinin ikili bir tamsayı olduğu varsayılır. Kodlama, MQMessage.Encoding özelliği tarafından belirtilir.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

*integerv% = MQMessage.ReadShort*

### **ReadString yöntemi**

This method reads n bytes from the Message Data buffer starting with the byte referred to by DataOffset and returns it as an ActiveX string.

MQMessage.DataLength komutu verildiğinde n değerinden küçükse yöntem başarısız olur.

Yöntem başarılı olursa,DataOffset , n artırılır ve DataLength değeri n tarafından azaltılır.

The n characters of message data are assumed to be a string in the code page that is specified by the MQMessage.CharacterSet property. Uygulama için ActiveX gösterimine dönüştürme gerçekleştirilir.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:** *stringv \$= MQMessage.ReadString (uzunluk & )*

**Değiştirge**

*uzunluk ve Uzun*. Bayt cinsinden dizgi alanı uzunluğu.

### **ReadUInt2 yöntemi**

DataOffsettarafından başvuru olan bayt ile başlayarak, ReadUInt2 yöntemi İleti Verileri arabelleğinden 2 bayt okur ve bunları uzun (imli 4 bayt) bir tamsayı değeri olarak döndürür.

The method fails if MQMessage.DataLength is less than 2 when it is issued.

Yöntem başarılı olursa,DataOffset , 2 artırılır ve DataLength değeri 2 tarafından azaltılır.

İleti verilerinin 2 baytı işaretli bir ikili tamsayı olarak kabul edilir. Kodlama, MQMessage.Encoding özelliği tarafından belirtilir.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

*bigint* & = MQMessage .ReadUInt2

**ReadUnsignedByte yöntemi**

DataOffsettarafından başvuru byte ile başlayarak, ReadUnsignedByte yöntemi İleti Verileri arabelleğinden 1 byte 'ı okur ve 0-255 aralığında bir Tamsayı (imli 2-bayt) tamsayı değeri olarak döndürür.

The method fails if MQMessage.DataLength is less than 1 when it is issued.

Yöntem başarılı olursa,DataOffset , 1 artırılır ve DataLength değeri 1 ile azaltılır. Yöntem başarılı olursa, 1 değerini artırılır.

İleti verilerinin 1 karakterinin imzalanmamış bir ikili tamsayı olduğu varsayılır.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

*integerv%* = MQMessage .ReadUnsignedByte

**ReadUTF yöntemi**

This method reads a UTF format string from the message, starting from the byte referred to by **DataOffset**, and returns the UTF format string as an ActiveX string. Okunmakta olan UTF biçim dizgisi, dizilimin uzunluğunu gösteren 2 baytlık verilerden oluşur ve UTF karakter verileri izler.

**MQMessage.DataLength** komutu verildiğinde dizgi uzunluğundan küçükse yöntem başarısız olur.

**DataOffset** is incremented by the string length and **DataLength** is decremented by the string length if the method succeeds.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

```
value$ = MQMessage.ReadUTF
```

**ResizeBuffer yöntemi**

Bu yöntem, İleti Verileri arabelleğini tutmak için dahili olarak ayrılmış olan depolama miktarını değiştirir. Uygulama, otomatik arabellek yönetimi üzerinde bazı denetim sağlar. Bu durumda, uygulama büyük bir iletiyle başa çıkabileceğini biliyorsa, yeterli büyüklükte bir arabelleğin ayrılmasını sağlayabilirler. Uygulamanın bu çağrıyı kullanması gerekmez; yoksa, otomatik arabellek yönetim kodu arabellek büyüklüğünü sığacak şekilde büyütür.

Arabelleğin büyüklüğünü, yürürlükteki MessageLengthdeğerinden küçük olacak şekilde yeniden boyutlandırırsanız, veri kaybı riskine girmenize neden olur. Veri kaybederse, yöntem MQCC\_UYARY için bir CompletionCode ve MQRC\_DATA\_TRUNCATED için ReasonCode değerini döndürür.

Arabelleğin büyüklüğünü, **DataOffset** özelliğinin değerinden daha küçük olacak şekilde yeniden boyutlandırırsanız:

- **DataOffset** özelliği, yeni arabelleğin sonuna işaret edecek şekilde değiştirildi
- **DataLength** özelliği sıfır olarak ayarlandı
- **MessageLength** özelliği yeni arabellek büyüklüğü olarak değiştirildi

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:** MQMessage .ResizeBuffer ( Uzunluk & )

**Parametre:**

Uzunluk ve Uzun. Karakterlerde gerekli boyut.

**Yazma yöntemi**

Veri Görelî Konumu ile gönderme yapılan konumdaki bayt dizisinden ileti arabelleğine bir bayt dizisi yazar. Gerekirse, arabelleğin uzunluğu (MQMessage.MQMessageLength), bayt dizisinin tam uzunluğuna sığması için genişletilir. DataOffset , yöntem başarılı olursa yazılan bayt sayısı kadar artırılır.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.Write(value)
```

**Parametreler:**

*data*: bir bayt dizisi ya da bir bayt dizisine ilişkin değişken başvurusu

**WriteBoolean yöntemi**

İleti arabelleğindeki, 2 baytlık Boole değerinden 1 byte 'lık bir Boole değerini yazar. DataOffset bir artırılır.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteBoolean(value)
```

**Parametre:**

*değer*: Boole (2-bayt). Yazılacak değer.

**WriteByte yöntemi**

Bu yöntem, imzalanmış 2 baytlık bir tamsayı değerini alır ve bunu Message Data arabelleğiyle DataOffsettarafından belirtilen konumda 1 byte 'lık ikili sayı olarak yazar. Önceden arabelleğindeki konumdaki verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) genişletir.

Yöntem başarılı olursa,DataOffset bir artırılır.

Belirtilen değer -128-127 aralığında olmalıdır. Doğru değilse, yöntem CompletionCode MQCC\_FAILED ve ReasonCode MQRC\_WRITE\_VALUE\_ERROR ile döner.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteByte(value%)
```

**Parametre:** *value%* Tamsayı. Yazılacak değer.

**WriteDecimal2 yöntemi**

2 baytlık bir paketlenmiş onlu sayı olarak imzalı 2 baytlık bir tamsayı yazar. DataOffset , iki tarafından artırılır.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteDecimal2(value%)
```

**Parametre:**

*değer% Tamsayı. Yazılacak değer.*

**WriteDecimal4 yöntemi**

4 baytlık paketlenmiş bir ondalık sayı olarak, imzalı 4 baytlık bir tamsayıyı yazar. DataOffset , dört tarafından artırılır.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WritedDecimal4(value&)
```

**Parametre:**

*değer ve Uzun. Yazılacak değer.*

**WriteDouble yöntemi**

Bu yöntem, imzalanmış 8 baytlık bir kayan noktalı değer alır ve bunu İleti Verileri arabelleğiyle DataOffsettarafından adlandırılan konumdan başlayarak 8 baytlık bir kayan noktalı sayı olarak yazar. Arabelleğindeki bu konumlarda bulunan verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) genişletir.

Yöntem başarılı olursa,DataOffset 8 artırılır.

Yöntem, MQMessage.Encoding özelliği tarafından belirtilen kayan nokta gösterimine dönüştürür. *System/360 biçimine dönüştürme desteklenmiyor.*

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteDouble(value#)
```

**Parametre:**

*değer# Çift. Yazılacak değer.*

**WriteDouble4 yöntemi**

See “ReadDouble4 yöntemi” sayfa 691 for a description of when ReadDouble4 and WriteDouble4 should be used in place of ReadFloat and WriteFloat.

Bu yöntem, imzalanmış 8 baytlık bir kayan noktalı değer alır ve bunu İleti Verileri arabelleğiyle DataOffsettarafından adlandırılan konumdan başlayarak 4 baytlık bir kayan sayı olarak yazar.

Yöntem başarılı olursa,DataOffset , 4 artırılır.

Arabelleğindeki bu konumlarda bulunan verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) genişletir.

Yöntem, MQMessage.Encoding özelliği tarafından belirtilen kayan nokta gösterimine dönüştürür. *System/360 biçimine dönüştürme desteklenmiyor.*

**Tanımlı:** MQMessage sınıfı



**Sözdizimi:**

```
Call MQMessage.WriteDouble4(value#)
```

**Parametre:** *value#* Çift. Yazılacak değer.

**WriteFloat yöntemi**

Bu yöntem, imzalanmış 4 baytlık bir kayan noktalı değer alır ve bunu Message Data arabelleğine, DataOffsettarafından başvuru karakterde başlayan 4 baytlık bir kayan noktalı sayı olarak yazar. Arabelleğindeki bu konumlarda bulunan verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) genişletir.

Yöntem başarılı olursa,DataOffset , 4 artırılır.

Yöntem, MQMessage.Encoding özelliği tarafından belirtilen ikili gösterimine dönüştürür. *System/360 biçimine dönüştürme desteklenmiyor.*

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteFloat(value!)
```

**Parametre** *değer!* Yüzmek. Yazılacak değer.

**WriteInt2 yöntemi**

Bu yöntem WriteShort yöntemi ile aynıdır.

**Sözdizimi:**

```
Call MQMessage.WriteInt2(value%)
```

**Parametre** *değer%* Tamsayı. Yazılacak değer.

**WriteInt4 yöntemi**

Bu yöntem WriteLong yöntemi ile aynıdır.

**Sözdizimi:**

```
Call MQMessage.WriteInt4(value&)
```

**Parametre** *değer ve Uzun.* Yazılacak değer.

**WriteLong yöntemi**

Bu yöntem, imzalanmış 4 baytlık bir tamsayı değerini alır ve Message Data arabelleğiyle DataOffsettarafından adlandırılan bayt 'dan başlayarak 4 baytlık ikili bir sayı olarak yazar. Arabelleğindeki bu konumlarda bulunan verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) genişletir.

Yöntem başarılı olursa,DataOffset , 4 artırılır.

Yöntem, MQMessage.Encoding özelliği tarafından belirtilen ikili gösterimine dönüştürür.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteLong(value&)
```

**Parametre** *değer* ve Uzun. Yazılacak değer.

### ***WriteNullTerminatedString yöntemi***

Bu yöntem, olağan bir WriteString gerçekleştirir ve kalan tüm baytları boş değerle belirtilen uzunluğa kadar destekler. İlk yazma dizgisi tarafından yazılan baytların sayısı belirtilen uzunluğa eşitse, boş değer yazılmaz. Byte sayısı belirtilen uzunluğa aşarsa bir hata oluştu (neden kodu MQRC\_WRITE\_VALUE\_ERROR) ayarlanır.

Yöntem başarılı olursa,DataOffset belirtilen uzunluğa göre artırılır.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteNullTerminatedString(value$, length&)
```

**Parametreler:**

\$String değeri. Yazılacak değer.

uzunluk ve Uzun. Bayt cinsinden dizgi alanı uzunluğu.

### ***WriteShort yöntemi***

Bu yöntem, imzalanmış 2 baytlık bir tamsayı değerini alır ve Message Data arabelleğiyle DataOffsettarafından adlandırılan bayt 'dan başlayarak 2 baytlık ikili bir sayı olarak yazar. Arabelleğindeki bu konumlarda bulunan verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) uzatır.

Yöntem başarılı olursa,DataOffset , 2 artırılır.

Yöntem, MQMessage.Encoding özelliği tarafından belirtilen ikili gösterimine dönüştürür.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteShort(value%)
```

**Parametre** *değer%* Tamsayı. Yazılacak değer.

### ***WriteString yöntemi***

This method takes an ActiveX string and writes it into the Message Data buffer starting at the byte referred to by DataOffset. Arabelleğindeki bu konumlarda bulunan verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) uzatır.

Yöntem başarılı olursa,DataOffset , bayt cinsinden dizgi uzunluğuna göre artırılır.

Yöntem, karakterleri MQMessage.CharacterSet özelliği tarafından belirtilen kod sayfasına dönüştürür.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteString(value$)
```

**Parametre** *değer \$* Dize. Yazılacak değer.

### ***WriteUInt2 yöntemi***

Bu yöntem, imzalanmış 4 baytlık bir tamsayı değerini alır ve Message Data arabelleğine, DataOffsettarafından gönderme yapılan bayttan başlayarak 2 baytlık imzalanmamış bir ikili sayı olarak yazar. Arabelleğindeki bu konumlarda bulunan verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) genişletir.

Yöntem başarılı olursa,DataOffset , 2 artırılır.

Yöntem, MQMessage.Encoding özelliği tarafından belirtilen ikili gösterimine dönüştürür. Belirlenen değer 0-2 \* \*16-1 aralığında olmalıdır. Bu yöntem, CompletionCode MQCC\_FAILED ve ReasonCode MQRC\_WRITE\_VALUE\_ERROR ile döndüren yöntem değildir.

**Tanımlı:** MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteUInt2(value&)
```

**Parametre** *değer* ve Uzun. Yazılacak değer.

### **WriteUnsignedByte yöntemi**

Bu yöntem, imzalanmış 2 baytlık bir tamsayı değerini alır ve Message Data arabelleğine, DataOffsettarafından başvuru karakterde başlayan 1 baytlık imzalanmamış bir ikili sayı olarak yazar. Arabelleğindeki bu konumlarda bulunan verilerin yerine geçer ve gerekirse arabelleğin uzunluğunu (MQMessage.MessageLength) genişletir.

Yöntem başarılı olursa,DataOffset , 1 artırılır.

Belirlenen değer, 0-255 aralığında olmalıdır. Bu yöntem, CompletionCode MQCC\_FAILED ve ReasonCode MQRC\_WRITE\_VALUE\_ERROR ile döndüren yöntem değildir.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:**

```
Call MQMessage.WriteUnsignedByte(value%)
```

**Parametre** *değer%* Tamsayı. Yazılacak değer.

### **WriteUTF yöntemi**

Bu yöntem, bir ActiveX dizgisini alır ve UTF biçiminde geçerli konumda ileti veri arabelleğiyle yazar. Yazılan veriler, 2 baytlık karakter uzunluğunun ve ardından karakter verilerinin izlediği bir uzunluktan oluşur. Yöntem başarılı olursa,DataOffset , dizginin uzunluğuna göre artırılır.

**Tanımlandığı yer:**

MQMessage sınıfı

**Sözdizimi:** Call MQMessage. WriteUTF (değer\$)

**Parametre:**

*değer \$Dizgisi.* Yazılacak değer.

## **MQPutMessageSeçenekleri sınıfı**

This class encapsulates the various options that control the action of putting a message onto an IBM MQ Queue.

### **Bulundurma**

The MQPutMessageOptions class is contained by the MQSession class.

### **Yaratma**

**Yeni** seçeneği, yeni bir MQPutMessageSeçenekleri nesnesi yaratır ve tüm özelliklerini ilk değerlerine ayarlar.

Diğer bir seçenek olarak, MQSession sınıfının AccessPutMessageOptions yöntemini kullanın.

## Sözdizimi

**Dim** *pmo* **Farklı Yeni MQPutMessageSeçenekleri** ya da

**Set** *pmo* = **Yeni MQPutMessageSeçenekleri**

## Özellikler

- “CompletionCode özelliği” sayfa 700.
- “Seçenekler özelliği” sayfa 700.
- “ReasonCode özelliği” sayfa 700.
- “ReasonName özelliği” sayfa 701.
- “RecordFields özelliği” sayfa 701.
- “ResolvedQueueManagerName özelliği” sayfa 701.
- “ResolvedQueueAd özelliği” sayfa 701.

## Yöntemler

- “ClearErrorCodes yöntemi” sayfa 701.

### CompletionCode özelliği

Salt okunur. Nesneye yönelik olarak verilen son yöntem ya da özellik erişimi tarafından ayarlanan tamamlanma kodunu döndürür.

**Tanımlı:** MQPutMessageSeçenekleri sınıfı

**Veri Tipi:** Uzun

#### Değerler:

- MQCC\_OK
- MQCC\_UYARI
- MQCC\_FAILED

**Sözdizimi:** almak için: *completioncode* & = *PutOpts* .**CompletionCode**

### Seçenekler özelliği

Okuma-yazma. MQPMO Seçenekleri alanı. Bu alanın ilk değeri MQPMO\_NONE olur. Ek bilgi için [MQPMO seçenekleribaşlıklı konuya](#) bakın.

**Tanımlı:** MQPutMessageSeçenekleri Sınıfı.

**Veri Tipi:** Uzun

**Sözdizimi:** almak için: *options* & = *PutOpts* .**Seçenekler**

Ayarlamak için: *PutOpts* .**Seçenekler** = *seçenekler* ve

MQPMO\_PASS\_IDENTITY\_CONTEXT ve MQPMO\_PASS\_ALL\_CONTEXT seçenekleri desteklenmiyor.

### ReasonCode özelliği

Salt okunur. Nesneye karşı verilen son yöntem ya da özellik erişimiyle belirlenen neden kodunu döndürür.

**Tanımlı:** MQPutMessageSeçenekleri sınıfı

**Veri Tipi:** Uzun

#### Değerler:

- Bkz. API tamamlama ve neden kodları.

**Sözdizimi:** almak için: *reasoncode & = PutOpts .ReasonCode*

### ***ReasonName özelliği***

Salt okunur. En son neden kodunun simgesel adını döndürür. Örneğin, "MQRC\_QMGR\_NOT\_AVALABILIR".

**Tanımlı:** MQPutMessageSeçenekleri sınıfı

**Veri Tipi:** Dize

#### **Değerler:**

- Bkz. API tamamlama ve neden kodları.

**Sözdizimi:** almak için: *reasonname \$= PutOpts .ReasonName*

### ***RecordFields özelliği***

Okuma-yazma. Dağıtım listesine bir ileti yerleştirilirken, kuyruklar başına hangi alanların uyarlanacağı belirten işaretler. Başlangıç değeri sıfır.

Bu özellik, MQI MQPMO yapısındaki PutMsgRecFields işaretlerine karşılık gelir. MQI 'de, bu işaretler, MQPUT tarafından hangi alanların (MQPMR yapısında) bulunduğunu ve kullanıldığını denetler. Bir MQPutMessageOptions nesnesinde, bu alanlar her zaman vardır ve işaretler, yalnızca put tarafından kullanılan alanları etkiler.

#### **Tanımlandığı yer:**

MQPutMessageSeçenekleri sınıfı

#### **Veri Tipi:**

Uzun

**Sözdizimi:** almak için: *recordfields & = PutOpts .RecordFields*

Ayarlamak için: *PutOpts .RecordFields = recordfields &*

### ***ResolvedQueueManagerName özelliği***

Salt okunur. MQPMO ResolvedQMGrAd alanı. Ayrıntılar için [ResolvedQMGrName \(MQCHAR48\)](#) konusuna bakın. Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQPutMessageSeçenekleri sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: *qmgr \$= PutOpts .ResolvedQueueManagerName*

### ***ResolvedQueueAd özelliği***

Salt okunur. MQPMO ResolvedQName alanı. Ayrıntılı bilgi için [ResolvedQName \(MQCHAR48\)](#) başlıklı konuya bakın. Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQPutMessageSeçenekleri sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** almak için: *qname \$= PutOpts .ResolvedQueueAd*

### ***ClearErrorCodes yöntemi***

CompletionCode ögesini, hem MQPutMessageSeçenekleri sınıfı, hem de MQSession sınıfı için MQCC\_NONE ve MQRC\_NONE MQRC\_NONE değerine döndürür.

#### **Tanımlandığı yer:**

MQPutMessageSeçenekleri sınıfı

## Sözdizimi:

`Arama PutOpts .ClearErrorCodes ()`

## MQGetMessageSeçenekleri sınıfı

Bu sınıf, IBM MQ kuyruğundan ileti alma işlemini denetleyen çeşitli seçenekleri sarsalıyor.

### Bulundurma

The MQGetMessageOptions class is contained by the MQSession class.

### Yaratma

**Yeni** seçeneği, yeni bir MQGetMessageSeçenekleri nesnesi yaratır ve tüm özelliklerini ilk değerlerine ayarlar.

Diğer bir seçenek olarak, MQSession sınıfının AccessGetMessageOptions yöntemini kullanın.

### Özellikler

- [“CompletionCode özelliği” sayfa 702](#)
- [“MatchOptions özelliği” sayfa 702](#)
- [“Seçenekler özelliği” sayfa 703](#)
- [“ReasonCode özelliği” sayfa 703](#)
- [“ReasonName özelliği” sayfa 703](#)
- [“ResolvedQueueAd özelliği” sayfa 703](#)
- [“WaitInterval özelliği” sayfa 703](#)

### Yöntemler

- [“ClearErrorCodes yöntemi” sayfa 704](#)

## Sözdizimi

**Dim gmo Yeni MQGetMessageSeçenekleri Olarak** ya da

**Set gmo = Yeni MQGetMessageSeçenekleri**

### **CompletionCode özelliği**

Salt okunur. Nesneye yönelik olarak verilen son yöntem ya da özellik erişimi tarafından ayarlanan tamamlanma kodunu döndürür.

**Tanımlı:** MQGetMessageSeçenekleri Sınıfı.

**Veri Tipi:** Uzun

### **Değerler:**

- MQCC\_OK
- MQCC\_UYARI
- MQCC\_FAILED

**Sözdizimi:** Alınmak için: `completioncode & = GetOpts .CompletionCode`

### **MatchOptions özelliği**

Okuma-yazma. MQGET için kullanılan seçim ölçütlerini denetleyen seçenekler. Başlangıç değeri: MQMO\_MATCH\_MSG\_ID + MQMO\_MATCH\_COREL\_ID.

**Tanımlandığı yer:**

MQGetMessageSeçenekleri sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [MatchOptions \(MQUZE\)](#).

**Sözdizimi:** Alınmak için: *matchoptions* & = *GetOpts*. **MatchOptions**

Ayarlamak için: *GetOpts*. **MatchOptions** = *matchoptions* &

**Seçenekler özelliği**

Okuma-yazma. MQGMO Seçenekleri alanı. Ayrıntılar için [Seçenekler](#) konusuna bakın. İlk değer MQGMO\_NO\_DUR değeridir.

**Tanımlı:** MQGetMessageSeçenekleri Sınıfı.

**Veri Tipi:** Uzun

**Sözdizimi:** Alınmak için: *options* & = *GetOpts* .**Seçenekler** ayarlamak için: *GetOpts* .**Seçenekler** = *seçenekler* ve

**ReasonCode özelliği**

Salt okunur. Nesneye karşı verilen son yöntem ya da özellik erişimiyle belirlenen neden kodunu döndürür.

**Tanımlı:** MQGetMessageSeçenekleri sınıfı

**Veri Tipi:** Uzun

**Değerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** Alınmak için: *reasoncode* & = *GetOpts* .**ReasonCode**

**ReasonName özelliği**

Salt okunur. En son neden kodunun simgesel adını döndürür. Örneğin, "MQRC\_QMGR\_NOT\_AVALABILIR".

**Tanımlı:** MQGetMessageSeçenekleri sınıfı

**Veri Tipi:** Dize

**Değerler:**

- Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** Alınmak için: *reasonname* \$= *MQGetMessageOptions* .**ReasonName**

**ResolvedQueueAd özelliği**

Salt okunur. MQGMO ResolvedQName alanı. Ayrıntılı bilgi için [ResolvedQName \(MQCHAR48\)](#) başlıklı konuya bakın. Başlangıç değeri boşluk karakteridir.

**Tanımlı:** MQGetMessageSeçenekleri sınıfı

**Veri Tipi:** 48 karakterden oluşan dize

**Sözdizimi:** Alınmak için: *qname* \$= *GetOpts* .**ResolvedQueueAd**

**WaitInterval özelliği**

Okuma/yazma. MQGMO WaitInterval alanı. Options özelliği tarafından bekleme işlemi istendiye, Get 'in uygun bir ileti gelmesi için bekleyeceği süre üst sınırı (milisaniye olarak). Bu alanın başlangıç değeri 0 (0). MQGMO seçeneklerine ilişkin ayrıntılar için bakınız: [MQGMO](#).

**Tanımlı:** MQGetMessageSeçenekleri sınıfı

**Veri Tipi:** Uzun

**Sözdizimi:** Alınmak için: *wait & = GetOpts .WaitInterval*

Ayarlamak için: *GetOpts .WaitInterval = wait &*

### ***ClearErrorCodes* yöntemi**

CompletionCode ögesini, hem MQGetMessageOptions sınıfı, hem de MQSession sınıfı için MQCC\_NONE ve MQRC\_NONE MQRC\_NONE olarak sıfırlar.

**Tanımlandığı yer:**

MQGetMessageSeçenekleri sınıfı

**Sözdizimi:**

**Call** *GetOpts .ClearErrorCodes ()*

## **MQDistributionList sınıfı**

Bu sınıf, çıkış için yerel, uzak ya da diğer ad derlemine sarsalıyor.

### **Yaratma**

**yeni** , yeni bir MQDistributionList nesnesi yaratır.

Diğer bir seçenek olarak, MQQueueManager sınıfının AddDistributionList yöntemini kullanın.

### **Özellikler**

- [“AlternateUserTanıtıcı özelliği” sayfa 704](#)
- [“CloseOptions özelliği” sayfa 705](#)
- [“CompletionCode özelliği” sayfa 705](#)
- [“ConnectionReference özelliği” sayfa 705](#)
- [“FirstDistributionListItem özelliği” sayfa 705](#)
- [“IsOpen özelliği” sayfa 706](#)
- [“OpenOptions özelliği” sayfa 706](#)
- [“ReasonCode özelliği” sayfa 706](#)
- [“ReasonName özelliği” sayfa 706](#)

### **Yöntem**

- [“AddDistributionListItem yöntemi” sayfa 707](#)
- [“ClearErrorCodes yöntemi” sayfa 707](#)
- [“Yöntemi kapat” sayfa 707](#)
- [“Open yöntemi” sayfa 707](#)
- [“Put yöntemi” sayfa 708](#)

### **Sözdizimi**

**dim** *distlist*. **A s New** MQDistributionList ya da **Set** *distlist = New* MQDistributionList

### ***AlternateUserTanıtıcı özelliği***

Okuma-yazma. Açıldığında kuyruklar listesine erişimi doğrulamak için kullanılan diğer kullanıcı kimliği.

**Tanımlandığı yer:**

MQDistributionList sınıfı



**Veri Tipi:**

12 karakterden oluşan dizgi

**Sözdizimi:** almak için: *altuser \$= MQDistributionList. AlternateUserTanıtıcısı*

Ayarlamak için: *MQDistributionList. AlternateUserTanıtıcısı = altuser \$*

**CloseOptions özelliği**

Okuma-yazma. Dağıtım listesi kapatıldığında ne olacağını denetlemek için kullanılan seçenekler. İlk değer MQCO\_NONE olur.

**Tanımlandığı yer:**

MQDistributionList sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

- MQCO\_NONE
- MQCO\_DELETE
- MQCO\_DELETE\_PURGE

**Sözdizimi:** almak için: *closeopt & = MQDistributionList. CloseOptions*

Ayarlamak için: *MQDistributionList. CloseOptions = closeopt &*

**CompletionCode özelliği**

Salt okunur. Nesneye karşı verilen son yöntem ya da özellik erişimi tarafından ayarlanan tamamlanma kodu.

**Tanımlandığı yer:**

MQDistributionList sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

- MQCC\_OK
- MQCC\_UYARI
- MQCC\_FAILED

**Sözdizimi:** almak için: *completioncode & = MQDistributionList. CompletionCode*

**ConnectionReference özelliği**

Okuma-yazma. Dağıtım listesinin ait olduğu kuyruk yöneticisi.

**Tanımlandığı yer:**

MQDistributionList sınıfı

**Veri Tipi:**

MQueueManager

**Sözdizimi:** almak için: *set queuemanager = MQDistributionList. ConnectionReference*

Şu şekilde ayarlayın: *set MQDistributionList. ConnectionReference = queuemanager*

**FirstDistributionListItem özelliği**

Salt okunur. Dağıtım listesiyle ilişkili ilk dağıtım listesi ögesi nesnesi.

**Tanımlandığı yer:**

MQDistributionList sınıfı

**Veri Tipi:**

MQDistributionListÖgesi

**Değerler:**

**Sözdizimi:** almak için: *set distributionlistitem = MQDistributionList. FirstDistributionListItem*

***IsOpen özelliği***

Salt okunur.

**Tanımlandığı yer:**

MQDistributionList sınıfı

**Veri Tipi:**

Boole

**Değerler:**

- TRUE (-1)
- FALSE (0)

**Sözdizimi:** almak için: *IsOpen = MQDistributionList. IsOpen*

***OpenOptions özelliği***

Okuma-yazma. Dağıtım listesi açıldığında kullanılacak seçenekler.

**Tanımlandığı yer:**

MQDistributionList sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [MQPMO seçenekleri](#).

**Sözdizimi:** almak için: *openopt & = MQDistributionList. OpenOptions*

Ayarlamak için: *MQDistributionList. OpenOptions = openopt &*

***ReasonCode özelliği***

Salt okunur. Nesneye karşı verilen son yöntem ya da özellik erişimi tarafından ayarlanan neden kodu.

**Tanımlandığı yer:**

MQDistributionList sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasoncode & = MQDistributionList. ReasonCode*

***ReasonName özelliği***

Salt okunur. ReasonCode için simgesel ad. Örneğin, "MQRC\_QMGR\_NOT\_AVALABILIR".

**Tanımlandığı yer:**

MQDistributionList sınıfı

**Veri Tipi:**

Dizgi

**Değerler:**

Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasonname \$= MQDistributionList. ReasonName*

## **AddDistributionListItem yöntemi**

Yeni bir MQDistributionListItem nesnesi yaratmak ve bunu dağıtım listesi nesnesiyle ilişkilendirmek için bu yöntemi kullanın. Kuyruk adı parametresi zorunludur.

Bu yöntem, varolan bir listenin ilk ögesi olarak yeni bir dağıtım listesi ögesi ekler. Özellikle, bu yöntem aşağıdaki yapılanışı yaratır:

- Dağıtım listesinde, **FirstDistributionListItem** özelliğini yeni dağıtım listesi ögesini işaret edecek şekilde ayarlar.
- Yeni dağıtım listesi ögesinde, aşağıdaki özellikleri ayarlar:
  - **DistributionList** özelliğini dağıtım listesini işaret edecek şekilde ayarlar.
  - **PreviousDistributionListItem** özelliğini boş değer olarak ayarlar.
  - **NextDistributionListItem** özelliğini, önceden önce olan dağıtım listesi ögesini işaret edecek şekilde ayarlar ya da listede önceki öge yoksa boş değer (null) olarak ayarlar.

Dağıtım listesi açık olduğunda yeni bir öge eklemek için bu yöntemi kullanamazsınız.

### **Tanımlandığı yer:**

MQDistributionList sınıfı

**Sözdizimi:** set distributionlistitem = MQDistributionList .AddDistributionListItem (QName\$, QMgrName\$)

### **Parametreler:**

QName\$ Dize. IBM MQ kuyruğunun adı.

QMgrName\$ Dize. IBM MQ kuyruk yöneticisinin adı.

## **ClearErrorCodes yöntemi**

CompletionCode ögesini, hem MQDistributionList sınıfı, hem de MQSession sınıfı için MQCC\_NONE ve MQRC\_NONE MQRC\_NONE olarak sıfırlar.

### **Tanımlandığı yer:**

MQDistributionList sınıfı

### **Sözdizimi:**

```
Call MQDistributionList.ClearErrorCodes()
```

## **Yöntemi kapat**

Geçerli Kapanış seçeneklerinin geçerli değerini kullanarak bir dağıtım listesini kapatır.

### **Tanımlandığı yer:**

MQDistributionList sınıfı

### **Sözdizimi:**

```
Call MQDistributionList. Close ()
```

## **Open yöntemi**

Geçerli nesneyle ilişkili dağıtım listesi öğelerinin **QueueName** ve (uygun olduğu) **QueueManagerName** özelliklerinin, geçerli AlternateUserTanıtıcısının geçerli değerini kullanarak belirlediği kuyrukları açar.

### **Tanımlandığı yer:**

MQDistributionList sınıfı

### **Sözdizimi:**

```
Call MQDistributionList.Open()
```

## **Put yöntemi**

Dağıtım listesiyle ilişkili dağıtım listesi öğeleriyle tanımlanan kuyrukların her birine bir ileti yerleştirir.

### **Tanımlandığı yer:**

MQDistributionList sınıfı

### **Sözdizimi**

MQDistributionList çağrısını arayın. **Koy** (İleti, PutMsgSeçenekleri ve &)

### **Parametreler**

Konulacak iletiyi gösteren *İleti* MQMessage nesnesi.

Koyma işlemini denetleme seçeneklerini içeren *PutMsgSeçenekleri* MQPutMessageOptions nesnesi. Belirtilmemişse, varsayılan PutMessageSeçenekleri kullanılır.

Bu yöntem, bir MQMessage nesnesini değiştirge olarak alır. Aşağıdaki dağıtım listesi öğesi özellikleri, bu yöntemin bir sonucu olarak değiştirilebilir:

- CompletionCode
- ReasonCode
- ReasonName
- MessageId
- MessageIdOnaltılı
- CorrelationId
- CorrelationIdOnaltılı
- GroupId
- GroupIdOnaltılı
- Geribildirim
- AccountingToken
- AccountingTokenAltılı

## **MQDistributionListÖğe sınıfı**

Bu sınıf, MQOR, MQRR ve MQPMR yapılarını sarsalıyor ve bunları sahip olan bir dağıtım listesiyle ilişkilendirir.

### **Yaratma**

MQDistributionList sınıfının AddDistributionListItem Yöntemini kullanın.

### **Özellikler**

#### **Yöntemler**

- [“AccountingToken özelliği” sayfa 709.](#)
- [“AccountingTokenHex özelliği” sayfa 710.](#)
- [“CompletionCode özelliği” sayfa 710.](#)
- [“CorrelationId özelliği” sayfa 710.](#)
- [“CorrelationIdHex özelliği” sayfa 710.](#)
- [“DistributionList özelliği” sayfa 711.](#)

- “Feedback özelliđi” sayfa 711.
- “GroupId özelliđi” sayfa 711.
- “GroupIdHex özelliđi” sayfa 711.
- “MessageId özelliđi” sayfa 712.
- “MessageIdHex özelliđi” sayfa 712.
- “NextDistributionListItem özelliđi” sayfa 712.
- “PreviousDistributionListItem özelliđi” sayfa 712.
- “QueueManagerAd özelliđi” sayfa 712.
- “QueueName özelliđi” sayfa 713.
- “ReasonCode özelliđi” sayfa 713.
- “ReasonName özelliđi” sayfa 713.
- “ClearErrorCodes yöntemi” sayfa 713.

### **Özellikler**

- AccountingToken özelliđi
- AccountingTokenHex özelliđi
- CompletionCode özelliđi
- CorrelationId özelliđi
- CorrelationIdHex özelliđi
- DistributionList özelliđi
- Feedback özelliđi
- GroupId özelliđi
- GroupIdHex özelliđi
- MessageId özelliđi
- MessageIdHex özelliđi
- NextDistributionListItem özelliđi
- PreviousDistributionListItem özelliđi
- QueueManagerAd özelliđi
- QueueName özelliđi
- ReasonCode özelliđi
- ReasonName özelliđi

#### *Yöntemler:*

- ClearErrorCodes yöntemi

#### *Yaratma:*

MQDistributionList sınıfının AddDistributionListItem Yöntemini kullanın.

### **AccountingToken özelliđi**

Okuma-yazma. Bir kuyruđa bulunduđunda iletinin MQPMR ' ye eklenmesi için AccountingToken . Bařlangıç deđeri tüm boş deđerlerde olur.

#### **Tanımlandığı yer:**

MQDistributionListÖđe sınıfı

#### **Veri Tipi:**

32 karakter dizgisi

**Sözdizimi:** almak için: *accountingtoken \$= MQDistributionListÖgesi*. **AccountingToken**

Ayarlamak için: *MQDistributionListÖge*. **AccountingToken** = *accountingtoken \$*

### **AccountingTokenHex özelliği**

Okuma-yazma. Bir kuyruğa bulunduğu iletinin MQPMR ' ye eklenmesi için AccountingToken .

Dizilimin her iki karakteri, tek bir ASCII karakterinin onaltılı değerini gösterir. Örneğin, "6" ve "1" karakter çifti, tek karakteri "A", "6" ve "2" karakter çiftini temsil eder ve tek karakteri "B" vb. gösterir.

64 geçerli onaltılı karakter sağlamalısınız.

Başlangıç değeri: "0 ... 0".

#### **Tanımlandığı yer:**

MQDistributionListÖge sınıfı

#### **Veri Tipi:**

32 ASCII karakteri yeniden sunan, 64 onaltılı karakterden oluşan dizgi.

**Sözdizimi:** almak için: *accountingtokenh \$= MQDistributionListÖgesi*. **AccountingTokenOnaltılı**

Ayarlamak için: *MQDistributionListÖge*. **AccountingTokenHex** = *accountingtokenh \$*

### **CompletionCode özelliği**

Salt okunur. Son açma ya da koyma isteği tarafından belirlenen tamamlanma kodu, sahip olan dağıtım listesi nesnesine ilişkin olarak verildi.

#### **Tanımlandığı yer:**

MQDistributionListÖge sınıfı

#### **Veri Tipi:**

Uzun

#### **Değerler:**

- MQCC\_OK
- MQCC\_UYARI
- MQCC\_FAILED

**Sözdizimi:** almak için: *completioncode \$= MQDistributionListÖgesi*. **CompletionCode**

### **CorrelationId özelliği**

Okuma-yazma. Bir kuyruğa bulunduğu iletinin MQPMR ' ye eklenmesi için CorrelId (CorrelId). Başlangıç değeri tüm boş değerlerde olur.

#### **Tanımlandığı yer:**

MQDistributionListÖge sınıfı

#### **Veri Tipi:**

24 karakter dizgisi

**Sözdizimi:** almak için: *correlid \$= MQDistributionListÖgesi*. **CorrelationId**

Ayarlamak için: *MQDistributionListÖge*. **CorrelationId** = *correlid \$*

### **CorrelationIdHex özelliği**

Okuma-yazma. Bir kuyruğa bulunduğu iletinin MQPMR ' ye eklenmesi için CorrelId (CorrelId).

Dizilimin her iki karakteri, tek bir ASCII karakterinin onaltılı değerini gösterir. Örneğin, "6" ve "1" karakter çifti, tek karakteri "A", "6" ve "2" karakter çiftini temsil eder ve tek karakteri "B" vb. gösterir.

48 geçerli onaltılı karakter sağlamalısınız.

Başlangıç değeri: "0 .. 0".

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

24 ASCII karakteri simgeleyen 48 onaltılı karakterden oluşan dizgi.

**Sözdizimi:** almak için: *correlidh \$= MQDistributionListögesi*. **CorrelationIdHex**

Ayarlamak için: *MQDistributionListÖğe*. **CorrelationIdHex** = *correlidh \$*

**DistributionList özelliği**

Salt okunur. Bu dağıtım listesi ögesinin ilişkilendirildiği dağıtım listesi.

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

MQDistributionList

**Sözdizimi:** almak için: *distributionlist = MQDistributionListÖğe ayarla*. **DistributionList**

**Feedback özelliği**

Okuma-yazma. Bir ileti kuyruğuna konduğunda iletinin MQPMR ' ye eklenmesi için Geribildirim değeri.

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [Geribildirim \(MQUZE\)](#).

**Sözdizimi:** almak için: *feedback & = MQDistributionListItem*. **Geribildirim**

Ayarlamak için: *MQDistributionListÖğe*. **Geribildirim** = *geribildirim &*

**GroupId özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQPMR ' ye eklenmesi için GroupId . Başlangıç değeri tüm boş değerlerde olur.

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

24 karakter dizgisi

**Sözdizimi:** almak için: *groupid \$= MQDistributionListögesi*. **GroupId**

Ayarlamak için: *MQDistributionListÖğe*. **GroupId** = *groupid \$*

**GroupIdHex özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQPMR ' ye eklenmesi için GroupId .

Dizilimin her iki karakteri, tek bir ASCII karakterinin onaltılı değerini gösterir. Örneğin, "6" ve "1" karakter çifti, tek karakteri "A", "6" ve "2" karakter çiftini temsil eder ve tek karakteri "B" vb. gösterir.

48 geçerli onaltılı karakter sağlamalısınız.

Başlangıç değeri: "0 .. 0".

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

24 ASCII karakteri yeniden sunan, 48 onaltılı karakterden oluşan dizgi.

**Sözdizimi:** almak için: *groupidh \$= MQDistributionListögesi*. **GroupIdHex**

Ayarlamak için: *MQDistributionListÖğe*. **GroupIdHex** = *groupidh \$*

### **MessageId özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQPMR ' ye dahil edilmesi için MessageId . Başlangıç değeri tüm boş değerlerde olur.

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

24 karakter dizgisi

**Sözdizimi:** almak için: *messageid \$= MQDistributionListögesi*. **MessageId**

Ayarlamak için: *MQDistributionListÖğe*. **MessageId** = *messageid \$*

### **MessageIdHex özelliği**

Okuma-yazma. Bir kuyruğa konduğunda iletinin MQPMR ' ye dahil edilmesi için MessageId .

Dizilimin her iki karakteri, tek bir ASCII karakterinin onaltılı değerini gösterir. Örneğin, "6" ve "1" karakter çifti, tek karakteri "A", "6" ve "2" karakter çiftini temsil eder ve tek karakteri "B" vb. gösterir.

48 geçerli onaltılı karakter sağlamalısınız.

Başlangıç değeri: "0 .. 0".

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

24 ASCII karakteri simgeleyen 48 onaltılı karakterden oluşan dizgi.

**Sözdizimi:** almak için: *messageidh \$= MQDistributionListögesi*. **MessageIdHex**

Ayarlamak için: *MQDistributionListÖğe*. **MessageIdHex** = *messageidh \$*

### **NextDistributionListItem özelliği**

Salt okunur. Bir sonraki dağıtım listesi ögesi nesnesi, aynı dağıtım listesiyle ilişkilendirildi.

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

MQDistributionListÖgesi

**Sözdizimi:** almak için: *set distributionlistitem = MQDistributionListögesi*. **NextDistributionListItem**

### **PreviousDistributionListItem özelliği**

Salt okunur. Aynı dağıtım listesiyle ilişkilendirilmiş önceki dağıtım listesi ögesi nesnesi.

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

MQDistributionListÖgesi

**Sözdizimi:** almak için: *set distributionlistitem = MQDistributionListItem*. **PreviousDistributionListItem**

### **QueueManagerAd özelliği**

Okuma-yazma. IBM MQ kuyruk yöneticisi adı.

**Tanımlandığı yer:**

MQDistributionListÖğe sınıfı

**Veri Tipi:**

48 karakter dizgisi.



**Sözdizimi:** almak için: *qmname \$= MQDistributionListÖğesi. QueueManagerAdı*

Ayarlamak için: *MQDistributionListÖğesi. QueueManagerAdı = qmname \$*

### **QueueName özelliği**

Okuma-yazma. IBM MQ kuyruk adı.

**Tanımlandığı yer:**

MQDistributionListÖğesi sınıfı

**Veri Tipi:**

48 karakter dizgisi.

**Sözdizimi:** almak için: *qname \$= MQDistributionListÖğesi. QueueName*

Ayarlamak için: *MQDistributionListÖğesi. QueueName = qname \$*

### **ReasonCode özelliği**

Salt okunur. Son açılımın ya da sahip olan dağıtım listesi nesnesine gönderilen neden kodu.

**Tanımlandığı yer:**

MQDistributionListÖğesi sınıfı

**Veri Tipi:**

Uzun

**Değerler:**

Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için:

```
reasoncode& = MQDistributionListItem.ReasonCode
```

### **ReasonName özelliği**

Salt okunur. ReasonCode için simgesel ad. Örneğin, "MQRC\_QMGR\_NOT\_AVAILABLE".

**Tanımlandığı yer:**

MQDistributionListÖğesi sınıfı

**Veri Tipi:**

Dizgi

**Değerler:**

Bkz. [API tamamlama ve neden kodları](#).

**Sözdizimi:** almak için: *reasonname \$= MQDistributionListÖğesi. ReasonName*

### **ClearErrorCodes yöntemi**

CompletionCode ögesini, hem MQDistributionListÖğesi sınıfı, hem de MQSession sınıfı için MQCC\_NONE ve MQRC\_NONE MQRC\_NONE olarak sıfırlar.

**Tanımlandığı yer:**

MQDistributionListÖğesi sınıfı

**Sözdizimi:**

```
Call MQDistributionListItem.ClearErrorCodes
```

## **ActiveX için IBM MQ Otomasyon Sınıflarını İzleme**

Information on the trace facility provided for IBM MQ Automation Classes for ActiveX, common pitfalls and help on how to avoid them.

IBM MQ 9.0' tan Microsoft Active X için IBM MQ desteği kullanımdan kaldırılmıştır. IBM MQ classes for .NET are the recommended replacement technology. Daha fazla bilgi için bkz [Developing .NET applications](#).

Aşağıdaki bölümde, sağlanan izleme olanağı açıklanır ve bunların önüne geçilmesine ilişkin yardım içeren ortak atışlara ilişkin ayrıntılar açıklanmaktadır:

- “ActiveX için IBM MQ Otomasyon Sınıfları İzlemeyi Denetleme” sayfa 714
- “IBM MQ Automation Classes for ActiveX komut dosyası başarısız olduğunda” sayfa 715
- “Reason codes IBM MQ Automation Classes for ActiveX” sayfa 715
- “Kod düzeyi aracı” sayfa 718

## ActiveX için IBM MQ Otomasyon Sınıfları İzlemeyi Denetleme

The IBM MQ Automation Classes for ActiveX (MQAX) includes a trace facility to help the service organization identify what is happening when you have a problem.

Bu, MQAX komut dosyanızı çalıştırdığınızda alınan yolları gösterir. Bir sorun yoksa, sistem kaynaklarının gereksiz bir şekilde kullanılmamasını önlemek için izlemeyi devre dışı bırakmadan çalıştırın.

İzlemeyi denetlemek için ayarladığınız üç ortam değişkeni vardır:

- OMQ\_TRACE
- OMQ\_TRACE\_PATH
- OQ\_IZLEME\_DÜZEYI

**Not:** **OMQ\_TRACE** değişkeni için *any* değerinin belirlenmesi, izleme olanağını açık olarak değiştirir. **OMQ\_TRACE** değişkenini OFF (Kapalı) olarak ayarlasanız bile, izleme hala etkin durumda olur. İzlemeyi kapatmak için **OMQ\_TRACE** için bir değer belirtmeyin.

1. **Başlat** düğmesini tıklatın.
2. **Control Panel** (Denetim Masası)
3. **System** (Sistem) seçeneğini çift tıklatın
4. **Gelişmiş** düğmesini tıklatın.
5. **Ortam** düğmesini tıklatın.
6. **Kullanıcı değişkenleri (kullanıcı adı)** başlıklı bölümde **Yeni** 'yi tıklatın.
7. Değişken adını ve uygun alanlara geçerli bir değer girin ve **Tamam** düğmesini tıklatın.
8. Ortam Değişkenleri penceresini kapatmak için **Tamam** düğmesini tıklatın.
9. Sistem Özellikleri penceresini kapatmak için **Tamam** düğmesini tıklatın.
10. Control Panel (Denetim Masası) penceresini kapatın

İzleme dosyalarının yazılmasını istediğiniz yere karar verirken, okunmanın yanı sıra diskten yazmak için yeterli yetkiye sahip olduğundan emin olun.

İzleme etkinleştirildiğinde, MQAX 'in çalıştırılmalarını yavaşlatır, ancak ActiveX ya da IBM MQ ortamlarınızın performansını etkilemez. Artık bir izleme dosyasına gerek kalmadığında, dosyayı silebilirsiniz.

MQAX çalışırken, OMQ\_TRACE değişkeninin durumunu değiştiremezsiniz.

## İzleme dosyası adı ve dizini

İzleme dosyası adı OMQnnnnn.trc biçimini alır; burada *nnnn* , o sırada çalışan ActiveX işleminin tanıtıcısıdır.

Çizelge 103. Komutlar ve etkileri	
Komut	Efekt
SET OMQ_TRACE_PATH = sürücü: \directory	İzleme dosyasının yazıldığı izleme dizinini belirler.

Çizelge 103. Komutlar ve etkileri (devamı var)

Komut	Efekt
SET OMQ_TRACE_PATH =	İzleme dizini için varolan bir ayarı kaldırır. İzleme dizini ayarlanmadığında, yürürlükteki çalışma dizini ( ActiveX başlatıldığında) kullanılır.
ECHO %OMQ_TRACE_PATH%	Windowsüzzerindeki izleme dizinine ilişkin yürürlükteki ayarı görüntüler.
SET OMQ_TRACE = xxxxxxxx	İzlemeye geçiş yapar. İzlemeyi '=' işaretinden sonra bir ya da daha çok karakter koyarak etkinleştirmenizi sağlar. Örneğin: SET OMQ_TRACE=yes ve SET OMQ_TRACE=no. Bu örneklerin her ikisinde de izleme etkindir. Bu ayar yalnızca tek bir window/session için etkilidir.
SET OMQ_TRACE=	İzlemeyi kapatır.
ECHO %OMQ_TRACE%	Windowsüzzerindeki ortam değişkeninin içeriğini görüntüler.
Belirle	Windowsüzzerindeki tüm ortam değişkenlerinin içeriğini görüntüler.
SET OMQ_TRACE_LEVEL = 9	İzleme düzeyini 9 olarak ayarlar. 9 'dan büyük değerler, izleme dosyasında ek bilgi üretmez.

## IBM MQ Automation Classes for ActiveX komut dosyası başarısız olduğunda

If your IBM MQ Automation Classes for ActiveX script fails, there are a number of sources of information that you can look at.

### İlk hata belirtisi raporu

İzleme olanından bağımsız olarak, beklenmeyen ve iç hatalar için ilk hata belirtisi raporu üretilebilir.

Bu rapor, OMQnnnnn . fdcadlı bir dosyada bulunur; burada nnnnn , o sırada çalışmakta olan ActiveX işleminin numarasıdır. Bu dosyayı, ActiveX ' u başlattığınız çalışma dizininde ya da OMQ\_PATH ortam değişkeninde belirtilen yolda bulursunuz.

### Diğer bilgi kaynakları

IBM MQ , ilgili platforma bağlı olarak çeşitli hata günlükleri ve izleme bilgileri sağlar. Windows uygulama olay günlüğünüze bakın.

## Reason codes IBM MQ Automation Classes for ActiveX

IBM MQ MQI neden kodlarına ek olarak oluşabilecek IBM MQ Automation Classes for ActiveX (MQAX) için neden kodları.

IBM MQ MQI için belgelenenlere ek olarak aşağıdaki neden kodları da olabilir. Diğer kodlar için, IBM MQ uygulama olay günlüğünüze bakın.

Çizelge 104. Neden kodları ve ne anlama geldikleri

Neden Kodu	Açıklama
MQRC_LIBRARY_LOAD_ERROR (6000)	IBM MQ kitaplıklarından biri ya da daha fazlası yüklenemedi. Tüm IBM MQ kitaplıklarının, kullanmakta olduğunuz sistemde doğru arama yolunda olup olmadığını denetleyin. Örneğin, IBM MQ kitaplıklarını içeren dizinlerin PATHiçinde olduğundan emin olun.
MQRC_CLASS_LIBRARY_ERROR (6001)	IBM MQ classlibrary çağrılarında biri beklenmeyen bir ReasonCode ya da CompletionCode değeri döndürdü. Ayrıntılar için First Failure Symptom Raporuna bakın. Kullanılmakta olan son yöntemi/özelliği ve sınıfı not alın ve sorunun IBM Destek birimine bildirmesini bildirin.

Çizelge 104. Neden kodları ve ne anlama geldikleri (devamı var)

Neden Kodu	Açıklama
MQRC_STRING_LENGTH_TOO_BIG (6002)	İleti arabelleğindeki uzunluğu 65,535 byte 'tan büyük bir UTF biçim dizgisi yazmak için girişimde bulunuldu.
MQRC_WRITE_VALUE_ERROR (6003)	Aralık dışında bir değer kullanılır; örneğin, <code>msg.WriteByte (240)</code> .
MQRC_PACKED_DECIMAL_ERROR (6004)	İleti arabelleğinden paketlenmiş bir ondalık sayıyı okuma girişiminde bulunuldu, ancak veri işaretçisinin veri göstergesi geçerli bir paketlenmiş veri biçiminde değil.
MQRC_FLOAT_CONVERSION_ERROR (6005)	İleti arabelleğinden tek ya da çift kayan noktalı sayı okuma girişiminde bulunuldu, ancak veri işaretçisinde veri imleci uygun bir kayar noktalı biçimde değil.
MQRC_REOPEN_EXCL_INPUT_ERROR (6100)	Açık bir nesne doğru <b>OpenOptions</b> ayarlarına sahip değildir ve bir ya da daha fazla ek seçenek gerektirir. Gizli bir yeniden açma gerekli, ancak, kuyruk dışlayıcı giriş ve kapanış için açık olduğu için kapatma engellendi; ancak, kuyruğa alma işlemi, diğer kişilerin kuyruğa erişmesi için bir fırsat penceresi sunacaktır. Set the <b>OpenOptions</b> values explicitly to cover all eventualities so that implicit reopening is not required.
MQRC_REOPEN_INQUIRE_ERROR (6101)	Açık bir nesne doğru <b>OpenOptions</b> ayarlarına sahip değildir ve bir ya da daha fazla ek seçenek gerektirir. An implicit reopen is required but closure has been prevented because one or more characteristics of the object need to be checked dynamically prior to closure, and the <b>OpenOptions</b> values do not already include the <b>MQOO_INQUIRE</b> option. <b>OpenOptions</b> değerlerini belirttik olarak <b>MQOO_INQUIRE</b> seçeneğini içerecek şekilde ayarlayın.
MQRC_REOPEN_SAVED_CONTEXT_ERR (6102)	Açık bir nesne doğru <b>OpenOptions</b> ayarlarına sahip değildir ve bir ya da daha fazla ek seçenek gerektirir. Örtük bir yeniden açma gerekli; ancak, kuyruk <b>MQOO_SAVE_ALL_CONTEXT</b> seçeneğiyle açık olduğu ve daha önce yıkıcı bir Get çağrısı gerçekleştirildiği için kapatma engellendi. Bu, alıkonan durum bilgilerinin açık kuyrukla ilişkilendirilmesine neden oldu ve bu bilgiler kapatılarak yok edilecek. Set the <b>OpenOptions</b> values explicitly to cover all eventualities so that implicit reopening is not required.
MQRC_REOPEN_TEMPORARY_Q_ERROR (6103)	Açık bir nesne doğru <b>OpenOptions</b> ayarlarına sahip değildir ve bir ya da daha fazla ek seçenek gerektirir. Örtük bir yeniden açma gerekli, ancak kuyruk, kapatılarak yok edilecek <b>MQQDT_TEMPORARY_DYNAMIC</b> tanım tipinin yerel bir kuyruğu olduğu için kapatma engellendi. Set the <b>OpenOptions</b> values explicitly to cover all eventualities so that implicit reopening is not required.
MQRC_ATTRIBUTE_LOCKED (6104)	Nesne açıkken, bir nesnenin değerini ya da özniteliğini değiştirmek için girişimde bulunuldu. <b>AlternateUserId</b> gibi bazı öznitelikler, bir nesne açıkken değiştirilemez.
MQRC_CURSOR_NOT_VALID (6105)	Açık bir kuyruk için göz atma imleci, örtük bir yeniden açma tarafından son kullanılandan sonra geçersiz kılındı. Set the <b>OpenOptions</b> values explicitly to cover all eventualities so that implicit reopening is not required.
MQRCENCODING_ERROR (6106)	Sonraki ileti ögesinin kodlaması, okumak için <b>MQENC_NATIVE</b> kodlaması olmalıdır.
MQRC_STRUCID_ERROR (6107)	Veri işaretçisinden başlayarak 4 karakterden türetilen sonraki ileti ögesine ilişkin tanıtıcı eksik ya da ögenin okunmakta olduğu değişken tipiyle tutarsız.
MQRC_NULL_POINTER (6108)	Boş değerli (null) olmayan bir işaretçinin gerekli ya da örtük olarak bulunduğu boş değerli gösterge belirtildi. Bunun nedeni, çağrılacak parametreler olarak Visual Basic ya da Excel 'den kullanılan IBM MQ nesnelere ilişkin belirtik bildirimler kullanılmasından kaynaklanabilir. Örneğin: <ul style="list-style-type: none"> <li>From Visual Basic, <code>dim msg as Object</code> works correctly, whereas <code>dim msg as MqMessage</code> might not work correctly.</li> <li>Bir kuyruk tanımlı ve ayarlanmış Visual Basic 'ten <code>dim msg as MqMessageq.put msg</code> doğru çalışır, oysa Excel 'den bu komut bir <b>MQRC_NULL_POINTER</b> kural dışı durumu oluşturur.</li> </ul>

Çizelge 104. Neden kodları ve ne anlama geldikleri (devamı var)

Neden Kodu	Açıklama
MQRC_NO_CONNECTION_REFERCE (6109)	MQQueue nesnesi, MQQueueManager nesnesiyle bağlantısını kaybetti. Kuyruk yöneticisinin bağlantısı kesilirse bu durum ortaya çıkar. MQQueue nesnesini silin.
MQRC_NO_BUFFER (6110)	Kullanılabilir arabellek yok. Bir MQMessage nesnesi için, nesne durumunda bir iç tutarsızlık olduğu için arabellek ayrılmıyor.
MQRC_BINARY_DATA_LENGTH_ERROR (6111)	İkili verilerin uzunluğu, hedef özniteliğin uzunluğuna göre tutarsız. Sıfır, tüm öznitelikler için doğru bir uzunluktur. 24, bir <b>CorrelationId</b> özniteliği için ve <b>MessageId</b> özniteliği için doğru bir uzunluktur. 32, bir <b>AccountingToken</b> özniteliği için doğru bir uzunluktur.
MQRC_BUFFER_NOT_AUTOMAKS (6112)	Kullanıcı tanımlı ve yönetilen arabellekler yeniden boyutlandırılmaz. İleti arabellekleri sistem tarafından yönetildiğinden, bu bir iç tutarsızlığı gösterir.
MQRC_INSUFFICIENT_BUFFER (6113)	Veri işaretçisinin isteği yerine getirmesinin ardından kullanılabilir arabellek alanı yetersiz. Bunun nedeni, arabelleğin yeniden boyutlandırılmaması olabilir.
MQRC_INSUFFICIENT_DATA (6114)	Veri işaretçisi okuma isteğini yerine getirmek için yeterli veri yok. Arabelleği doğru boyutlara indirin ve verileri yeniden okuyun.
MQRC_DATA_TRUNCATED (6115)	Veriler bir arabellekten diğerine kopyalanırken kesildi. Bunun nedeni, hedef arabelleğin yeniden boyutlandırılmaması ya da bir ya da başka bir arabelleğin adreslenmesiyle ilgili bir sorun olması ya da bir arabellek daha küçük bir değiştirmeye küçülmekte olduğu için olabilir.
MQRC_ZERO_LENGTH (6116)	Pozitif bir uzunluğun gerekli ya da örtük olduğu bir yere sıfır (sıfır) uzunluk değeri verildi.
MQRC_NEGATIVE_LENGTH (6117)	Sıfır ya da pozitif bir uzunluğun gerekli olduğu yerde negatif bir uzunluk belirtildi.
MQRC_NEGATIVE_OFFSET (6118)	Sıfır ya da pozitif görel konum gerekli olduğunda negatif görel konum sağlandı.
MQRC_INCONSISTENT_BİÇİMİ (6119)	Sonraki ileti öğesinin biçimi, öğenin okunmakta olduğu değişkenle tutarsız.
MQRC_INCONSSTENT_OBJECT_STATE (6120)	Bu nesne, açık olan bu nesne ile gönderme yapılan MQQueueManager nesnesi arasında bir tutarsızlık var; bu nesne bağlı değil.
MQRC_CONTEXT_OBJECT_NOT_VALID (6121)	MQPutMessageOptions bağlam başvurusu geçerli bir MQQueue nesnesine gönderme yapmamaktadır. Nesne daha önce yok edildi.
MQRC_CONTEXT_OPEN_ERROR (6122)	MQPutMessageOptions bağlam başvurusu, bağlam oluşturmak için açılmayabilecek bir MQQueue nesnesine gönderme yapıyor. Bunun nedeni, MQQueue nesnesinin uygun olmayan açık seçenekleri olması olabilir. Hatanın nedenini belirlemek için başvuru nesne neden kodunu inceleyin.
MQRC_STRUC_LENGTH_ERROR (6123)	İç veri yapısının uzunluğu, içeriğiyle tutarlı değil. Bir MQRMH üstbilgisi için uzunluk, sabit alanları ve tüm görel konum verilerini içermek için yeterli değildir.
MQRC_NOT_CONNECTED (6124)	Kuyruk yöneticisine gerekli bir bağlantı kullanılmadığından ve bir bağlantı örtük olarak kurulamadığından, yöntem başarısız oldu.
MQRC_NOT_OPEN (6125)	Bir IBM MQ nesnesi açık olmadığı için bir yöntem başarısız oldu ve açma örtük olarak gerçekleştirilemez.
MQRC_DISTRIBUTION_LIST_BOŞ (6126)	Dağıtım listesinde MQDistributionListItem nesnesi olmadığı için MQDistributionList açılmadı. Düzeltilici işlem: Dağıtım listesine en az bir MQDistributionListItem nesnesi ekleyin.

Çizelge 104. Neden kodları ve ne anlama geldikleri (devamı var)

Neden Kodu	Açıklama
MQRC_INCONSISTENT_OPEN_SEÇENEKLERI (6127)	Nesne açık olduğundan ve açma seçenekleri gerekli işlemle tutarsız olduğundan bir yöntem başarısız oldu. Düzeltilici işlem: Uygun açık seçeneklerle nesneyi açın ve yeniden deneyin.
MQRC_WRONG_VERSION (6128)	Belirtilen ya da saptanan bir sürüm numarası yanlış ya da desteklenmediği için yöntem başarısız oldu.

## Kod düzeyi aracı

IBM Hizmet Ekibi tarafından hangi kod düzeyini kurduğunuz sorulacaktır.

Bunu öğrenmek için 'MQAXLEV' yardımcı programını çalıştırın.

Komut isteminden, MQAX200.dll ' i içeren dizine geçin ya da tam yol uzunluğunu ekleyin ve şunu girin:

```
MQAXLev MQAX200.dll > MQAXLEV.OUT
```

Burada MQAXLEV.OUT , çıkış dosyasının adıdır.

Bir çıkış dosyası belirtmezseniz, ayrıntı ekranda görüntülenir.

Aşağıdaki örnekte, kod düzeyi aracından örnek bir çıkış dosyası ayrıntılı olarak açıklanmıştır:

## Kod düzeyi aracından örnek çıkış dosyası

```
5639-B43 (C) Copyright IBM Corp. 1996, 2024. ALL RIGHTS RESERVED.  
***** Code Level is 5.1 *****  
lib/mqole/mqole.cpp, mqole, p000, p000 L981119      1.8 98/08/21  
lib/mqlsx/gmqdyn0a.c, mqlsx, p000, p000 L990212    1.6 99/02/11 16:40:24  
lib/mqlsx/pc/gmqdyn1p.c, mqlsx, p000, p000 L990212  1.6 99/02/11 16:44:14  
lib/mqlsx/xmqcsa.c, mqole, p000, p000 L990216      1.3 99/02/15 13:24:34  
lib/mqlsx/xmqfdca.c, mqlsx, p000, p000 L990212    1.3 99/02/11 16:40:35  
lib/mqlsx/xmqtrca.c, mqlsx, p000, p000 L990212    1.5 99/02/11 16:12:02  
lib/mqlsx/xmqutila.c, mqlsx, p000, p000 L990212    1.3 99/02/11 16:40:40  
lib/mqlsx/xmqutl1a.c, mqlsx, p000, p000 L990212    1.4 99/02/11 16:40:30  
lib/mqlsx/xmqcnv1a.c, mqlsx, p000, p000 L990212    1.9 99/02/11 16:40:56  
lib/mqlsx/xmqmsg.c, mqole, p000, p000 L990219     1.11 99/02/18 12:12:59
```

## MQAI ' yeActiveX arabirimi

COM arabirimlerine ve MQAI ' de kullanımlarına ilişkin kısa bir genel bakış için bkz. [“Component Object Model Interface olanağının kullanılması \( ActiveXiçinIBM MQ Otomasyon Sınıfları\)” sayfa 640.](#)

MQAI, uygulamaların PCF için gereken değişken uzunluklu arabellekleri doğrudan edinmeden ve biçimlendirmeden Programlanır Komut Biçimi (PCF) komutları oluşturmasına ve göndermesine olanak sağlar. MQAI ile ilgili daha fazla bilgi için bkz. [IBM MQ Yönetim Arabirimi \(MQAI\)](#). MQAI ActiveX MQBag sınıfı, MQAI tarafından desteklenen veri torbalarını, COM nesnelerinin yaratılmasını destekleyen herhangi bir dilde (örneğin, Visual Basic, C + +, Javave diğer ActiveX komut dosyası istemcileri) kullanabilecek bir şekilde sarmalıyor.

MQAI ActiveX arabirimi, MQI ' ye bir COM arabirimi sağlayan MQAX sınıflarıyla birlikte kullanılır. MQAX sınıflarına ilişkin ek bilgi için bkz. [“ActiveX dışı uygulamalara erişen MQAX uygulamalarının tasarlanması” sayfa 641.](#)

ActiveX arabirimi, MQBag adında tek bir sınıf sağlar. Bu sınıf, MQAI veri torbaları yaratmak için kullanılır; özellikleri ve yöntemleri, her bir çanta içinde veri öğeleri yaratmak ve bu öğeleri kullanarak çalışmak için kullanılır. MQBag Execute yöntemi, çanta verilerini bir IBM MQ kuyruk yöneticisine bir PCF iletisi olarak gönderir ve yanıtları toplar.

MQBag sınıfı, özellikleri ve yöntemleri hakkında daha fazla bilgi için bkz. [“MQBag sınıfı” sayfa 719](#).

PCF iletisi, belirlenen kuyruk yöneticisi nesnesine, isteğe bağlı olarak belirlenen istek ve yanıt kuyruklarını kullanarak gönderilir. Yanıtlar yeni bir MQBag nesnesi içinde döndürülür. Komutların ve yanıtların tam kümesi, [Programların Komut Biçimlerinin Tanımları](#) başlıklı konu altında açıklanmıştır. Komutlar, uygun istek ve yanıt kuyrukları seçilerek IBM MQ ağındaki herhangi bir kuyruk yöneticisine gönderilebilir.

## MQBag sınıfı

MQBag sınıfı, gerekli olduğu şekilde MQBag nesnelere yaratmak için kullanılır. Somutlaştırıldığında, MQBag sınıfı yeni bir MQBag nesne başvurusu döndürür.

Visual Basic 'te bir MQBag nesnesi yaratmak için aşağıdaki komutu girin:

```
Dim mqbagg As MQBag
Set mqbagg = New MQBag
```

## MQBag Özelliği

MQBag nesnelere ilişkin özellikler aşağıdaki listede açıklanmıştır:

- [“Öğe özelliği” sayfa 720](#).
- [“Sayı özelliği” sayfa 721](#).
- [“Seçenekler özelliği” sayfa 721](#).

## MQBag yöntemleri

MQBag nesnelere ilişkin yöntemler aşağıdaki listede açıklanmıştır:

- [“Yöntem ekle” sayfa 722](#).
- [“AddInquiry yöntemi” sayfa 722](#).
- [“Yöntemi temizle” sayfa 723](#).
- [“Yöntemi yürüt” sayfa 723](#).
- [“FromMessage yöntemi” sayfa 724](#).
- [“ItemType yöntemi” sayfa 724](#).
- [“Yöntemi kaldır” sayfa 725](#).
- [“Seçici yöntemi” sayfa 725](#).
- [“ToMessage yöntemi” sayfa 726](#).
- [“Yöntem kes” sayfa 726](#).

## Hata İşleme

MQBag nesnesindeki bir işlem sırasında, temeldeki MQAX ya da MQAI nesnesi tarafından torbaya döndürülen hatalar da içinde olmak üzere bir hata saptanırsa, hata kural dışı durumu ortaya çıktı. MQBag sınıfı, COM ISupportErrorbilgi arabirimini destekler; bu nedenle, hata işleme yordamınıza aşağıdaki bilgiler kullanılabilir:

- Hata numarası: saptanan hataya ilişkin IBM MQ neden kodundan ve bir COM olanağı kodundan oluşur. Tesis alanı, COM için standart olarak, hatanın sorumluluk alanını gösterir. IBM MQ tarafından saptanan hatalar için her zaman KOLAYITY\_ITF olur.
- Hata kaynağı: hatayı saptayan nesnenin tipini ve sürümünü belirtir. MQBag işlemleri sırasında saptanan hatalar için, hata kaynağı her zaman MQBag.MQBag1' dir.
- Hata açıklaması: IBM MQ neden kodu için simgesel adı veren dizgi.

Hata bilgilerine erişim, komut dosyası yazma dilinize bağlıdır; örneğin, Visual Basic 'te bilgiler Err nesnesinde döndürülür ve IBM MQ neden kodu, Err.Number' tan sabit vbObjectHatası çıkarılarak çıkarılarak elde edilir.

### **ReasonCode = Err.Number - vbObjectHatası**

MQBag Execute yöntemi bir PCF iletisi gönderirse ve bir yanıt alınırsa, gönderilen komut başarısız olsa da işlem başarılı olarak kabul edilir. Bu durumda, yanıt torbasının kendisi, Programlanabilir Komut Biçimlerinin Tanımlamaları içinde açıklandığı şekilde tamamlanma ve hata neden kodlarını içerir.

## **Öge özelliği**

### **Amaç**

Öge özelliği, bir çantadaki bir ögeyi temsil eder. Bir ögenin değerini ayarlamak ya da sorgulamak için kullanılır. Bu özelliğin kullanılması, aşağıdaki MQAI çağrılarına karşılık gelir:

- "mqSetDizgisi"
- "mqSetTamsayı"
- "mqInquireTamsayı"
- "mqInquireDizgi"
- "mqInquireBag"

in the Programlanabilir komut biçimleri başvurusu.

### **Biçim**

Öge (Seçici, ItemIndex, Değer)

### **Parametreler**

#### **Seçici (VARIANT)-giriş**

Ayarlanacak ya da sorgulanacak ögenin seçicisi.

Bir ögeyle ilgili araştırma sırasında, MQSEL\_ANY\_USER\_SELECTOR varsayılan değerdir. Bir öge ayarlarken, varsayılan olarak MQIA\_LIST ya da MQCA\_LIST değeri olur.

Selector uzun tipte bir tipse, MQRC\_SELECTOR\_TYPE\_ERROR sonuçları.

Bu parametre isteğe bağlıdır.

#### **ItemIndex (LONG)-giriş**

Bu değer, belirlenecek ya da sorgulanacak belirlenen seçiciye ilişkin ögenin oluşumunu tanımlar. MQIND\_NONE varsayılan değerdir.

Bu parametre isteğe bağlıdır.

#### **Değer (VARIANT)-giriş/çıkış**

Döndürülecek değer ya da ayarlanacak değer. Bir öge sorulması sırasında, dönüş değeri long, string ya da MQBag tipinde olabilir. Ancak, bir öge ayarlanırken, değer long ya da string tipinde olmalıdır; değilse, MQRC\_ITEM\_VALUE\_ERROR sonuçları.

## **Visual Basic Dili Çağırma**

Bir çantanın içindeki bir ögenin değerini ne zaman sorup sorulduğunuzda:

```
Value = mqbag[.Item]([Selector],  
[ItemIndex])
```

MQBag başvuruları için:



```
Set abag = mqbagg[.Item]([Selector],  
[ItemIndex])
```

Bir öğenin bir çantadaki değerini ayarlamak için:

```
mqbag[.Item]([Selector],  
[ItemIndex]) = Value
```

## Sayı özelliği

### Amaç

Count özelliği, bir çanta içindeki veri öğelerinin sayısını temsil eder. Bu özellik, [Programlanabilir komut biçimleri başvurusu](#) içindeki "mqCountÖğeleri" MQAI çağrısına karşılık gelir.

### Biçim

Sayı (*Selector*, *Value*)

### Parametreler

#### Seçici (VARIANT)-giriş

Sayıma dahil edilecek veri öğelerinin seçicisi.

MQSEL\_ALL\_USER\_SELECTORS, varsayılan değer olarak kullanılan varsayılan değerdir.

Selector uzun tipli değilse, MQRC\_SELECTOR\_TYPE\_ERROR döndürülür.

#### Değer (LONG)-çıkış

The number of items in the bag included by the *Selector*.

## Visual Basic Dili Çağırma

Bir çantadaki öğelerin sayısını döndürmek için:

```
ItemCount = mqbagg.Count([Selector])
```

## Seçenekler özelliği

### Amaç

Seçenekler özelliği, bir çantanın kullanımına ilişkin seçenekleri ayarlar. This property corresponds to the **Options** parameter of the MQAI call, "mqCreateBag," in the [Programlanabilir komut biçimleri başvurusu](#).

### Biçim

Seçenekler (*Options*)

### Parametreler

#### Seçenekler (LONG)-giriş/çıkış

Çanta seçenekleri.

**Not:** The bag options must be set **önce** data items are added to or set within the bag. Çanta boş değilse seçenekler değiştirilirse, MQRC\_OPTIONS\_ERROR sonuçları gelir. Bu, çanta daha sonra temizlenmiş olsa bile geçerlidir.

## Visual Basic Dili Çağırma

Bir çantanın içindeki bir öğenin seçenekleri ne zaman sorulabiliyor:

```
Options = mqbagg.Options
```

Çantadaki bir öğenin bir seçeneğini ayarlamak için:

```
mqbagg.Options = Options
```

## MQBag yöntemleri

MQBag nesnelere ilişkin yöntemler, aşağıdaki sayfalar üzerinden açıklanır.

### *Yöntem ekle*

#### Amaç

Add yöntemi, bir çantaya veri ögesi ekler. Bu yöntem, [Programlanabilir komut biçimleri başvurusu](#) içindeki MQAI çağrılarında, "mqAddInteger" ve "mqAddDizgisi" ögesine karşılık gelir.

#### Biçim

**Ekle (Value, Selector)**

#### Parametreler

##### Değer (VARIANT)-giriş

Veri ögesinin tamsayı ya da dizgi değeri.

##### Seçici (VARIANT)-giriş

Eklenecek öğeyi tanımlayan seçici.

Value tipine bağlı olarak, varsayılan olarak MQIA\_LIST ya da MQCA\_LIST olur. **Selector** parametresi uzun tipte değilse, MQRC\_SELECTOR\_TYPE\_ERROR sonuçları.

## Visual Basic Dili Çağırma

Bir torbaya öğe eklemek için:

```
mqbagg.Add(Value, [Selector])
```

### *AddInquiry yöntemi*

#### Amaç

AddInquiry yöntemi, bir SORGULAMA komutu yürütmek için bir denetim paketi gönderildiğinde döndürülecek özniteliği belirten bir seçici eklemenizi sağlar. Bu yöntem, [Programlanabilir komut biçimleri başvurusu](#) içindeki "mqAddSorgusu" MQAI çağrısına karşılık gelir.

#### Biçim

**AddInquiry (Inquiry)**

## Parametreler

### Sorgu (LONG)-giriş

Selector of the IBM MQ attribute to be returned by the INQUIRE administration command.

## Visual Basic Dili Çağırma

AddInquiry yöntemini kullanmak için:

```
mqbag.AddInquiry(Inquiry)
```

## Yöntemi temizle

### Amaç

Temizleme yöntemi, bir torbadan tüm veri öğelerini siler. Bu yöntem, [Programlanabilir komut biçimleri başvurusu](#) içinde MQAI çağrısına ("mqClearBag") karşılık gelir.

### Biçim

#### Temizle

## Visual Basic Dili Çağırma

Bir paketten tüm veri itmelerini silmek için:

```
mqbag.Clear
```

## Yöntemi yürüt

### Amaç

Execute yöntemi, komut sunucusuna bir denetim komutu iletisi gönderip, herhangi bir yanıt iletisi için bekler. Bu yöntem, [Programlanabilir komut biçimleri başvurusu](#) içindeki "mqExecute," MQAI çağrısına karşılık gelir.

### Biçim

**Execute (QueueManager, Command, OptionsBag, RequestQ, ReplyQ, ReplyBag)**

## Parametreler

### QueueManager (MQQueueManager)-giriş

Uygulamanın bağlandığı kuyruk yöneticisi.

### Komut (LONG)-giriş

Yürütülecek komut.

### OptionsBag (MQBag)-giriş

Aramanın işlenmesini etkileyen seçenekleri içeren çanta.

### RequestQ (MQQueue)-giriş

Denetim komut iletisinin konacağı kuyruk.

### ReplyQ (MQQueue)-giriş

Herhangi bir yanıt iletisinin alındığı kuyruk.

### ReplyBag (MQBag)-çıkış

Yanıt iletilerinden alınan verileri içeren bir çanta başvurusu.

## Visual Basic Dili Çağırma

Bir denetim komutu iletişi göndermek ve yanıt iletilerini beklemek için:

```
Set ReplyBag = mqbag.Execute(QueueManager, Command,  
[OptionsBag], [RequestQ], [ReplyQ])
```

### *FromMessage yöntemi*

#### Amaç

FromMessage yöntemi, verileri bir iletiden çantaya yükler. Bu yöntem, Programlanabilir komut biçimleri başvurusu içindeki "mqBufferToBag" MQAI çağrısına karşılık gelir.

#### Biçim

**FromMessage (Message, OptionsBag)**

#### Parametreler

##### İleti (MQMessage)-giriş

Dönüştürülecek verileri içeren ileti.

##### OptionsBag (MQBag)-giriş

Aramanın işlenmesini denetleyen seçenekler.

## Visual Basic Dili Çağırma

Bir iletiden bir çantaya veri yüklemek için:

```
mqbag.FromMessage(Message, [OptionsBag])
```

### *ItemType yöntemi*

#### Amaç

ItemType yöntemi, bir çantada belirtilen bir öğede değerin tipini döndürür. Bu yöntem, Programlanabilir komut biçimleri başvurusu içindeki "mqInquireItemInfo" MQAI çağrısına karşılık gelir.

#### Biçim

**ItemType (Selector, ItemIndex, ItemType)**

#### Parametreler

##### Seçici (VARIANT)-giriş

Sorgulanacak öğeyi tanımlayan seçici.

MQSEL\_ANY\_USER\_SELECTOR varsayılan değerdir. **Selector** parametresi uzun tipte değilse, MQRC\_SELECTOR\_TYPE\_ERROR sonuçları.

##### ItemIndex (LONG)-giriş

Sorgulanacak öğelerin dizini.

MQIND\_NONE varsayılan değerdir.

##### ItemType (LONG)-çıkış

Belirtilen öğenin veri tipi.

**Not: Selector** parametresi, **ItemIndex** parametresi ya da her ikisi de belirtilmelidir. Her iki deđiřtirge de varsa, MQRC\_PARAMETER\_MISSING sonuları.

## Visual Basic Dili ađırma

Bir deđerin tipini dndrmek iin:

```
ItemType = mqbag.ItemType([Selector],  
[ItemIndex])
```

### *Yntemi kaldır*

#### Ama

Kaldırma (Remove) yntemi, bir geyi antadan siler. Bu yntem, Programlanabilir komut biimleri bařvurusuiindeki "mqDeleteItem" adlı MQAI ađırısına karřılık gelir.

#### Biim

**Remove (Selector, ItemIndex)**

#### Parametreler

##### Seici (VARIANT)-giriř

Silinecek geyi tanımlayan seici.

MQSEL\_ANY\_USER\_SELECTOR varsayılan deđerdir. **Selector** parametresi uzun tipte deđilse, MQRC\_SELECTOR\_TYPE\_ERROR sonuları.

##### ItemIndex (LONG)-giriř

Silinecek genin dizini.

MQIND\_NONE varsayılan deđerdir.

**Not: Selector** parametresi, **ItemIndex** parametresi ya da her ikisi de belirtilmelidir. Her iki deđiřtirge de varsa, MQRC\_PARAMETER\_MISSING sonuları.

## Visual Basic Dili ađırma

antadan bir geyi silmek iin:

```
mqbag.Remove([Selector],[ItemIndex])
```

### *Seici yntemi*

#### Ama

Seici yntemi, bir anta iinde belirtilen bir genin seicisini dndrr. Bu yntem, Programlanabilir komut biimleri bařvurusuiindeki "mqInquireItemInfo" MQAI ađırısına karřılık gelir.

#### Biim

**Seici (Selector, ItemIndex, OutSelector)**

#### Parametreler

##### Seici (VARIANT)-giriř

Sorgulanacak geyi tanımlayan seici.

MQSEL\_ANY\_USER\_SELECTOR varsayılan değerdir. **Selector** parametresi uzun tipte değilse, MQRC\_SELECTOR\_TYPE\_ERROR sonuçları.

#### **ItemIndex (LONG)-giriş**

Sorgulanacak öğenin dizini.

MQIND\_NONE varsayılan değerdir.

#### **OutSelector (VARIANT)-çıkış**

Belirtilen öğenin seçicisi.

**Not: Selector** parametresi, **ItemIndex** parametresi ya da her ikisi de belirtilmelidir. Her iki değıştirge de varsa, MQRC\_PARAMETER\_MISSING sonuçları.

## **Visual Basic Dili Çağırma**

Bir öğenin seçicisini döndürmek için:

```
OutSelector = mqbag.Selector([Selector],  
[ItemIndex])
```

## **ToMessage yöntemi**

### **Amaç**

ToMessage yöntemi, bir MQMessage nesnesine başvuru döndürür. Başvuru, bir çantadan veri içeriyor. Bu yöntem, Programlanabilir komut biçimleri başvurusu içinde MQAI çağrısına ("mqBagToBuffer") karşılık gelir.

### **Biçim**

**ToMessage** (*OptionsBag*, *Message*)

### **Parametreler**

#### **OptionsBag (MQBag)-giriş**

Yöntemin işlenmesini denetleyen seçenekleri içeren bir çanta.

#### **İleti (MQMessage)-çıkış**

Çantadan veri içeren bir MQMessage nesnesi başvurusu.

## **Visual Basic Dili Çağırma**

ToMessage Yöntemini kullanmak için:

```
Set Message = mqbag.ToMessage([OptionsBag])
```

## **Yöntem kes**

### **Amaç**

Kısaltma yöntemi, bir çantadaki kullanıcı öğelerinin sayısını azaltır. Bu yöntem, Programlanabilir komut biçimleri başvurusu içindeki "mqTruncateBag" adlı MQAI çağrısına karşılık gelir.

### **Biçim**

**Truncate** (*ItemCount*)

## Parametreler

### ItemCount (LONG)-giriş

Kesilme gerçekleştikten sonra, çantada kalacak kullanıcı öğelerinin sayısı.

## Visual Basic Dili Çağırma

Bir çantadaki kullanıcı öğelerinin sayısını azaltmak için:

```
mqbag.Truncate(ItemCount)
```

## ActiveX Starter örnekleri için IBM MQ Otomasyon Sınıfları Hakkında

This appendix describes the IBM MQ Automation Classes for ActiveX Starter samples, and explains how to use them.

IBM MQ for Windows , aşağıdaki Visual Basic örnek programlarını sağlar:

- MQAXTRIV.VBP
- MQAXBSRV.VBP
- MQAXDLST.VBP
- MQAXCLSS.VBP

Bu örnekler Visual Basic 4 ya da Visual Basic 5 üzerinde çalışır. Bunları dizinde bulacaksınız ... \tools\mqax\samples\vb.

Aynı dizinde ayrıca Microsoft Excel ve html için örnekler de bulacaksınız. Bu bilgiler şunlardır:

- MQAX.XLS
- MQAXTRIV.XLS
- MQAXTRIV.HTM

**Not:** If using Visual Basic 5, you **gerekir** select and install Visual Basic component grid32.ocx.

## Örneklerde gösterilen nedir?

The samples demonstrate how to use IBM MQ Automation Classes for ActiveX to:

- Kuyruk yöneticisine bağlan
- Kuyruğa erişim
- Kuyruğun üzerine ileti koy
- Kuyruktan ileti al

Visual Basic (Visual Basic) örneğinin merkezi bir bölümü aşağıdaki sayfalarda g " nderilir.

[“Örnekleri çalıştırma hazırlığı yapılıyor” sayfa 728](#) ve

[“Örneklerde işlenirken hata oluştu” sayfa 728](#)

## ActiveX Starter örneklerinin çalıştırılması

IBM MQ Automation Classes for ActiveX Starter örnekleri çalıştırılmadan önce, çalışan bir varsayılan kuyruk yöneticiniz olduğunu ve gereken kuyruk tanımlarını yarattığınız denetçisini denetleyin. Kuyruk yöneticisi yaratıp çalıştırıp kuyruk yaratılmasına ilişkin ayrıntılar için [Administeringdosyasına](#) bakın. The sample uses the queue SYSTEM.DEFAULT.LOCAL.QUEUE which should be defined on any normally set up IBM MQ server.

Veri torbalarını kullanmanın farklı yolları aşağıdaki listede gösterildiği gibidir:

- Kuyruk yöneticisine bağlan

- Kuyruğa erişim
- Kuyruğun üzerine ileti koy
- Kuyruktan ileti al

For information on the MQAX starter samples for Microsoft Basic 4 or later, see [“MQAXTRIV örneğinin çalıştırılması” sayfa 728](#)

Kuyruk yöneticileri ve kuyruk nesneleri özelliklerine ve yöntemlerine göz atmanıza olanak tanıyan bir örnek hakkında bilgi almak için bkz. [“MQAXCLSS örneğinin başlatılması” sayfa 730](#)

MQAXDLST örneğine ilişkin bilgi edinmek için [“MQAXDLST örneği” sayfa 730](#)

For information on running the MQAX starter sample for Microsoft Excel 95 or later, MQAXTRIV.XLS, see [“MQAXTRIV.XLS örneği” sayfa 730](#).

Banka gösterisini MQAX.XLS ile çalıştırma hakkında bilgi için bkz. [“Running the Bank demonstration with MQAX.XLS” sayfa 730](#)

ActiveX uyumlu bir WWW tarayıcısını kullanarak başlangıç örneğine ilişkin bilgi için bkz. [“ActiveX uyumlu bir WWW tarayıcısını kullanarak başlangıç örneği” sayfa 731](#)

## Örnekleri çalıştırma hazırlığı yapılıyor

Örnekleri çalıştırmak için, çalıştırmak istediğiniz örneklerin hangisine bağlı olarak aşağıdakilerden birine gerek duyarsınız.

- Microsoft Visual Basic 4 (ya da üstü)
- Microsoft Excel 95 (ya da üstü)
- Bir Web tarayıcısı

Ayrıca aşağıdakiler de gerekir:

- Bir IBM MQ kuyruk yöneticisi çalışıyor.
- Bir IBM MQ kuyruğu önceden tanımlanmış.

## Örneklerde işlenirken hata oluştu

ActiveX paketi için IBM MQ Otomasyon Sınıflarında sağlanan örneklerin çoğu, küçük ya da hiç hata işleme yoktur. Hata işleme hakkında daha fazla bilgi için bkz. [“Hata işleme” sayfa 645](#).

## MQAXTRIV örneğinin çalıştırılması

1. Kuyruk yöneticisini başlatın.
2. Windows Explorer ya da File Manager' da örneğin, MQAXTRIV.VBP (Visual Basic Proje dosyası) ve dosyayı açın.  
Visual Basic programı, MQAXTRIV.VBP.
3. Visual Basic 'te, numuneyi çalıştırmak için 5 (F5) tuşuna basın.
4. Pencere biçiminde herhangi bir yeri tıklatın, **MQAX önemsiz test aracı**.

Her şey doğru şekilde çalışıyorsa, pencere arka planı yeşil olarak değişmelidir. Kurulumunuzda bir sorun varsa, pencere arka planı kırmızı olarak değişmeli ve hata bilgileri görüntülenecektir.

Aşağıdaki şekil, Visual Basic örneğinin merkezi kısmını göstermektedir.

```
Option Explicit

Private Sub Form_Click()

'*****
'* This simple example illustrates how to put and get an IBM MQ message to
'* and from an IBM MQ message queue. The data from the message returned by the
```



```

'* get is read and compared with that from the original message.
'*****
Dim MQSess As MQSession           '* session object
Dim QMgr As MQQueueManager        '* queue manager object
Dim Queue As MQQueue             '* queue object
Dim PutMsg As MQMessage          '* message object for put
Dim GetMsg As MQMessage          '* message object for get
Dim PutOptions As MQPutMessageOptions '* put message options
Dim GetOptions As MQGetMessageOptions '* get message options
Dim PutMsgStr As String          '* put message data string
Dim GetMsgStr As String          '* get message data string
'*****
'* Handle errors
'*****
On Error GoTo HandleError

'*****
'* Initialize the current position for the form
'*****
CurrentX = 0
CurrentY = 0

'*****
'* Create the MQSession object and access the MQQueueManager and (local) MQQueue
'*****
Set MQSess = New MQSession
Set QMgr = MQSess.AccessQueueManager("")
Set Queue = QMgr.AccessQueue("SYSTEM.DEFAULT.LOCAL.QUEUE", _
                             MQOO_OUTPUT Or MQOO_INPUT_AS_Q_DEF)

'*****
'* Create a new MQMessage object for use with put, add some data then create an
'* MQPutMessageOptions object and put the message
'*****
Set PutMsg = MQSess.AccessMessage()
PutMsgStr = "12345678 " & Time
PutMsg.MessageData = PutMsgStr
Set PutOptions = MQSess.AccessPutMessageOptions()
Queue.Put PutMsg, PutOptions

'*****
'* Create a new MQMessage object for use with get, set the MessageId (to that of
'* the message that was put), create an MQGetMessageOptions object and get the
'* message.
'*
'* Note: Setting the MessageId ensures that the get returns the MQMessage
'* that was put earlier.
'*****

Set GetMsg = MQSess.AccessMessage()
GetMsg.MessageId = PutMsg.MessageId
Set GetOptions = MQSess.AccessGetMessageOptions()
Queue.Get GetMsg, GetOptions
'*****
'* Read the data from the message returned by the get, compare it with
'* that from the original message and output a suitable message.
'*****
GetMsgStr = GetMsg.MessageData
Cls
If GetMsgStr = PutMsgStr Then
    BackColor = RGB(127, 255, 127) '* set to green for ok
    Print
    Print "Message data comparison was successful."
    Print "Message data: "" & GetMsgStr & """"
Else
    BackColor = RGB(255, 255, 127) '* set to amber for compare error
    Print "Compare error: "
    Print "The message data returned by the get did not match the " & _
    "input data from the original message that was put."
    Print
    Print "Input message data:      "" & PutMsgStr & """"
    Print "Returned message data: "" & GetMsgStr & """"
End If

Exit Sub
'*****
'* Handle errors
'*****
HandleError:
Dim ErrMsg As String
Dim StrPos As Integer

```

```

Cls
BackColor = RGB(255, 0, 0) '* set to red for error
Print "An error occurred as follows:"
Print ""
If MQSess.CompletionCode <> MQCC_OK Then
  ErrMsg = Err.Description
  StrPos = InStr(ErrMsg, " ") '* search for first blank
  If StrPos > 0 Then
    Print Left(ErrMsg, StrPos) '* print offending MQAX object name
  Else
    Print Error(Err) '* print complete error object
  End If
  Print ""
  Print "IBM MQ Completion Code = " & MQSess.CompletionCode
  Print "IBM MQ Reason Code = " & MQSess.ReasonCode
  Print "(" & MQSess.ReasonName & ")"
Else
  Print "Visual Basic error: " & Err
  Print Error(Err)
End If

Exit Sub

End Sub

```

## **MQAXCLSS örneğinin başlatılması**

Bu örnek, kuyruk yöneticileri ve kuyruk nesneleri özelliklerine ve yöntemlerine göz atmanızı sağlar.

1. Kuyruk yöneticisini başlatın.
2. Open the file, MQAXCLSS.VBP, by double-clicking on the document icon in Windows Explorer or by clicking File - Open from the file menu in Visual Basic.
3. Örneği başlatın.
4. Uygun kuyruk yöneticisini ve kuyruk adlarını girin ve ilgili düğmeleri tıklayın.

## **MQAXDLST örneği**

Visual Basic MQAXDLST Sample, bir dağıtım listesinin, aynı iletiyi bir put ile iki kuyruğa göndermek için kullanılmasını gösterir. Örneği çalıştırmak için, MQAXCLSS örneğiyle aynı işlemi gerçekleştirin.

## **MQAX Starter sample for Microsoft Excel 95 or later**

This section explains how to run the MQAX starter sample for Microsoft Excel 95 or later, MQAXTRIV.XLS.

### ***MQAXTRIV.XLS örneği***

1. Kuyruk yöneticisini başlatın.
2. Explorer 'da ya da File Manager'da, MQAX örneği MQAXTRIV.XLS.
3. Elektronik sayfadaki düğmeyi tıklayın.
4. Ekran, bir başarı (ya da başarısızlık) iletişiyle güncellenir.

### ***Running the Bank demonstration with MQAX.XLS***

Banka gösterisini yürütmek için bu adımları izleyin.

1. Kuyruk yöneticisini başlatın.
2. IBM MQ MQSC komut dosyasını ( BANK.TST. Bu, gerekli IBM MQ kuyruk tanımlamalarını ayarlar.  
Bir MQSC komut dosyasını nasıl kullanacağını öğrenmek için [MQSC komutlarını kullanarak yönetimbaşlıklı konuya](#) bakın.
3. MQAXBSRV.VBPkomutunu çalıştırın. Bu örnek program, bir arka uç uygulamasının benzetimini yapan sunucudur ve bu uygulamanın Microsoft Excel ile çalışması gerekir.
4. MQAX.XLS. Bu örnek, istemci IBM MQ gösterimidir.

5. Listeden bir müşteri seçin.
6. **Gönder** düğmesini tıklatın.

Kısa bir duraksama (yaklaşık 3 saniye) sonra, değerler ile doldurulan alanlar ve bir çubuk grafiği görüntülenir.

## ActiveX uyumlu bir WWW tarayıcısını kullanarak başlangıç örneği

**Not:** Bu örneği çalıştırmak için, ActiveX uyumlu bir Web tarayıcısı çalıştırıyor olmanız gerekir. Microsoft Edge (ancak Netscape Navigatordeğil) uyumlu bir Web tarayıcısıdır.

### HTML örneğini çalıştırma

Bu örnek, hem VBScript hem de JavaScript' den MQAX 'i nasıl çağırabileceğiniz gösterilir.

1. Kuyruk yöneticisini başlatın.
2. Open the file, "MQAXTRIV.HTM", in your ActiveX compatible Web browser.  
Bunu, Windows Explorer 'da dosya simgesini çift tıklatarak ya da ActiveX uyumlu Web tarayıcınızın Dosya menüsünden Dosya-Aç seçeneğini belirleyebilirsiniz.
3. Ekranaya gelen yönergeleri izleyin.

ULW

## AMQP istemci uygulamalarının geliştirilmesi

MQ Light API 'si de dahil olmak üzere 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 UNIX, Linuxya da Windowsüzerine kurabilirsiniz; bu kanal IBM i ya da z/OSüzerinde kullanılamaz.

MQ Light API, Oasis AMQP 1.0 iletişim kuralını temel alır. Node.js, Java, Ruby ve Python için ileti alışverişi API ' leri vardır.

MQ Light API 'sini kullanmak üzere geliştirilen bir uygulama, MQ Light yürütme ortamına, AMQP kanalı olan bir IBM MQ kuyruk yöneticisine ya da IBM Cloud (formerly Bluemix) içindeki bir MQ Light hizmetinin eşgörünümüne bağlanabilir.

### AMQP istemcilerinin geliştirilmesi

MQ Light API 'si, iş uygulamalarının hızlı bir şekilde prototipini ve geliştirilmesini kolaylaştırmayı amaçlar. <https://github.com/mqlight> adresinde Node.js, Java, Ruby ve Python için MQ Light API ' leri bulunmaktadır.

### Örnek AMQP istemcileri karşıdan yükleniyor

IBM MQ , MQ Light istemcilerini göndermez, ancak aşağıdaki MQ Light istemcilerini karşıdan yükleyip kurabilirsiniz:

#### Node.js

MQ Light Node.js API ' yi npm kullanarak çalışma dizininize kurun: `npm install mqlight@1.0`

#### Java

Gerekli sürümüne ilişkin mqlight dağıtım paketini Maven Central 'dan yükleyin ve içeriği açın. [Maven Central](#) ' da mqlight dağıtım paketlerinin kullanılabilir sürümlerini bulabilirsiniz.

#### Ruby

MQ Light Ruby API ' yı gem kullanarak çalışma dizininize kurun: `gem install mqlight --pre`

#### Python

MQ Light Python API ' yı pip kullanarak çalışma dizininize kurun: `pip install mqlight --pre`

MQ Light istemcisi karřıdan yüklemelerinin tümü, farklı ileti sistemi özelliklerini gösteren birkaç örnek içerir:

- Örnek gönder
- Alma örneđi
- UI Workout örneđi

Apache Qpid kitaplıklarına dayalı olarak diđer 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 desteđi sağlayamıyor ve herhangi bir kullanım sorusu veya kod hatası raporu ilgili projelere yönlendirilmelidir.

## AMQP istemcilerinin güvenliđinin sađlanması

MQ Light uygulamalarının güvenliđini sađlama hakkında bilgi için bkz. [AMQP istemcilerinin güvenliđini sađlama](#).

## AMQP istemcilerinin IBM MQ ' a konuşlandırılması

Bir uygulama devreye alınmaya hazır olduđunda, diđer kurumsal uygulamaların tüm izleme, güvenilirlik ve güvenlik yeteneklerini gerektirir. Ayrıca diđer kurumsal uygulamalarla da veri alışveriři yapabilir. MQ Light uygulamalarını bir IBM MQ kuyruk yöneticisine konuşlandırabilirsiniz. Bkz. [“Deploying MQ Light applications to an on-premises IBM MQ environment” sayfa 748](#).

Bir AMQP istemcisi konuşlandırıđınızda, IBM MQ uygulamalarıyla ileti alışveriři yapabilirsiniz. Örneđin, bir JavaScript dizgi iletisi göndermek için MQ Light Node.js istemcisini kullanırsanız, IBM MQ uygulaması bir MQ iletisi alır; burada MQMD biçim alanı MQSTR olarak ayarlanır.

## AMQP kanalının yönetilmesi

AMQP kanalı, diđer 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 kullanılması alanında, istemcilerin bir kuyruk yöneticisine bağlanmasını tanımlamak ve başlatmak için örnek komutlar sađlanır.

Bir AMQP kanalı başlatıldıđında, ařađdaki yöntemlerden birini kullanarak MQ Light uygulamasını bağlayarak bunu sınavabilirsiniz:

- Node.js ve Java için IBM MQ Light istemcisini kullanma.
- Ruby ve Python için IBM MQ Light istemcisini kullanma.
- Başka bir AMQP 1.0 istemcisi kullanılıyor. Örneđin, Apache Qpid Proton.

### İlgili görevler

[AMQP kanallarının oluşturulması ve kullanılması](#)

[AMQP istemcilerinin güvenliđinin sađlanması](#)

ULW

## MQ Light ve AMQP (Geliřmiş İleti Kuyruđa Yollama İletişim Kuralı)

IBM MQ Light API 'si, OASIS Standardı AMQP 1.0 aktarım iletişim kuralına dayalıdır. AMQP, iletilerin gönderenler ve alıcılar arasında nasıl gönderileceđini belirtir. Uygulama, IBM MQ gibi Message Broker 'a bir ileti gönderdiđinde, bir uygulama gönderen olarak işlev görür. IBM MQ , bir AMQP uygulamasına ileti gönderdiđinde gönderen olarak hareket eder.

AMQP ' nin bazı avantajlarından bazıları řunlardır:

- Açık standartlařtırılmış bir iletişim kuralı
- Diđer açık kaynak AMQP 1.0 istemcileriyle uyumluluk
- Birçok açık kaynak istemcisi somutlaması var

AMQP 1.0 istemcisi bir AMQP kanalına bağlanabilirse de, bazı AMQP özellikleri desteklenmez; örneğin, işlemler ya da birden çok oturum.

Daha fazla bilgi için bkz. [AMQP.org](http://AMQP.org) web sitesi ve [OASIS Standard AMQP 1.0 PDF](#).

MQ Light ileti sistemi API 'si, AMQP 1.0' a dayalıdır. API, yayınlama/abone olma ve noktadan noktaya ileti alışverişi akışlarının çoğunluğu için gereken ileti alışverişi yeteneinden çoğunu sağlar.

MQ Light API, aşağıdaki ileti alışverişi özelliklerine sahiptir:

- En-en-bir kez ileti teslimi
- En az bir kez ileti teslimi
- Konu dizgi hedefi adreslemesi
- İleti ve hedef dayanıklılık
- Birden çok abonenin iş yükünü paylaşmasına izin vermek için paylaşılan hedefler
- Askılı müşterilerin kolay çözülmesi için Müşteri tarafından devralınması
- İletiler öncesinde okunabilen okuma
- İletilere ilişkin yapılandırılabilir alındı bildirimini

MQ Light API ' ye ilişkin eksiksiz bilgi için aşağıdaki web sitelerine bakın:

- Node.js API belgeleri için bkz. <https://www.npmjs.org/package/mqlight>
- Ruby API belgeleri için bkz. <https://www.rubydoc.info/github/mqlight/ruby-mqlight>
- Python API belgeleri için bkz. <https://python-mqlight.readthedocs.org>
- Java API belgeleri için bkz. <https://mqlight.github.io/java-mqlight>

### İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

ULW

## AMQP 1.0 desteği

AMQP kanalları, AMQP 1.0-compliant uygulamalar için bir destek düzeyi sağlar.

AMQP kanalları, AMQP 1.0 protokolünün bir alt kümesini destekler. MQ Light istemcilerini ya da diğer AMQP 1.0 uyumlu istemcilerini bir IBM MQ AMQP kanalına bağlayabilirsiniz. AMQP kanalları tarafından desteklenen tüm ileti alışverişi özelliklerini kullanmak için, belirli AMQP 1.0 alanlarının değerini doğru bir şekilde ayarlamamız gerekir.

Bu bilgiler, AMQP alanlarının biçimlendirilmesi ve AMQP kanallarının desteklemediği AMQP 1.0 belirtiminin özelliklerini listelemektedir.

AMQP 1.0 belirtiminin aşağıdaki özellikleri desteklenmez ya da kullanımları için sınırlanmıştır:

### Bağlantı adları

AMQP kanalları, üç biçimden birini izlemek için bir AMQP bağlantısının adını bekler:

- Düz bir konu (yayınlama ve abone olma için)
  - Publishing messages: a plain topic string (for example, a link name of "/sports/football") causes a message to be published on the /sports/football topic.
  - İleti almak için bir konuya abone olma: düz bir konu dizgisi (örneğin, "/sports/football" bağlantı adı, bir aboneliğin /sports/football konusu üzerinde tanımlanmasına neden olur.
- Özel bir ayrıntılı konu (abone olmak için)
  - Şu formdaki özel bir aboneliği açıklayan ayrıntılı bir konu dizgisi: "private:topic string" (örneğin: "private:/sports/football"). Davranış, düz bir konu dizgisiyle aynı. private bildirimini, istemciler arasında paylaşılan bir abonelikten belirli bir AMQP istemcisine özgü bir aboneliği ayırır.

- Paylaşılan ayrıntılı konu (abone olunması için)
  - Formdaki bir paylaşılan aboneliği açıklayan ayrıntılı bir konu dizgisi: "share:share name:topic string" (örneğin: "share:bbc:/sports/football").

AMQP iletilerinin eşlendiği ve IBM MQ iletilerinden more iletilerine ilişkin daha fazla bilgi için bkz. [AMQP alanlarının IBM MQ alanlarına eşlenmesi \(gelen iletiler\)](#).

## Konu dizgileri, paylaşım adları ve istemci tanıtıcıları için uzunluk üst sınırı

Konu dizgisi, paylaşım adı ve istemci tanıtıcısı 10237 byte içinde yer almalıdır. Buna ek olarak, istemci tanıtıcısının uzunluk üst sınırı 256 karakterdir.

Bu uzunluk üst sınırı, aşağıdakilerden birine sahip olmak anlamına gelir:

- paylaşım adının kısa olması koşuluyla çok uzun bir konu dizisi.
- uzun bir paylaşım adı, ancak kısa bir konu dizisi

## Taşıyıcı Tanıtıcıları

AMQP kanalları, bir AMQP Open performative adlı kapsayıcı tanıtıcısının benzersiz bir MQ Light istemci tanıtıcısını içermesini bekler. Bir MQ Light istemci tanıtıcısının uzunluk üst sınırı 256 karakterdir ve tanıtıcı alfasayısal karakterler, yüzde işareti (%), eğik çizgi (/), nokta (.) ve alt çizgi (\_) içerebilir.

## Oturumlar

AMQP kanalları yalnızca tek bir AMQP oturumu destekler. Birden çok AMQP oturumu yaratma girişiminde bulunan bir AMQP istemcisi, bir hata iletileri alır ve kanaldan bağlantılıdır.

## İşlemler

AMQP kanalları AMQP işlemlerini desteklemez. Yeni bir hareketi ya da yeni bir hareketi bildirmeyi deneyen bir AMQP aktarım çerçevesini koordine etmeyi deneyen bir AMQP bağlantı çerçevesi hata iletileriyle reddedilir.

## Teslim durumu

AMQP kanalları, kabul edilen yok etme çerçeveleri için yalnızca bir teslim durumunu destekler.

## İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

ULW

## AMQP ve IBM MQ ileti alanlarının eşlenmesi

AMQP iletileri, bir üstbilgi, teslim açıklamalarından, ileti ek açıklamalarından, özelliklerden, uygulama özelliklerinden, gövdesinden ve altbilgiden oluşur.

AMQP iletileri aşağıdaki kısımlardan oluşur:

### Üstbilgi

İsteğe bağlı üstbilgi, iletinin beş sabit özneliğini içerir:

- **dayanıklı** - dayanıklılık gereksinimlerini belirtir
- **priority** - göreliliği
- **tll** - milisaniye olarak yaşamının zamanı
- **first-acquirer** - bu doğruysa, ileti başka herhangi bir bağlantı tarafından satın alınmamış olabilir
- **delivery-count** - önceki, başarısız teslim girişimlerinin sayısı.

### Teslim-ek açıklamalar

İsteğe Bağlı. İletinin farklı amaçlanan hedef kitlelere ilişkin standart dışı üstbilgi özniteliklerini belirtir. Teslim ek açıklamaları, gönderen eşten alma eşlerine bilgi iletir.

### İleti-ek açıklamalar

İsteğe Bağlı. İletinin farklı amaçlanan hedef kitlelere ilişkin standart dışı üstbilgi özniteliklerini belirtir. İleti-ek açıklama kısmı, altyapıya yönelik iletinin özellikleri için kullanılır ve her teslim adımında yayılması gerekir.

### Özellikler

İsteğe Bağlı. Bu parça, MQ ileti tanımlayıcısına eşdeğerdir. Aşağıdaki sabit alanları içerir:

- **ileti-kimliği** -uygulama iletisi tanıtıcısı
- **user-id** -Kullanıcı yaratma tanıtıcısı
- - -İletinin kaderinde olduğu düğümün adresi
- **subject** -İletinin konusu
- **yanıt-kaynağı** -gönderme yanıtının gönderileceği düğüm
- **korelasyon-tnt** -uygulama ilinti tanıtıcısı
- **content-type** -MIME içerik tipi
- **content-encoding** -MIME içerik tipi. İçerik tipi için bir değiştirici olarak kullanılır.
- **mutlak-süre bitimi-saat** -bu iletinin süresi dolmuş olarak kabul edilen saat
- **yaratılma zamanı** -bu iletinin yaratıldığı saat
- **grup-tnt** -bu iletinin ait olduğu grup
- **grup-sırası** -bu iletinin grup içindeki sıra numarası.
- **yanıt-grubu-tnt** -yanıt iletisinin ait olduğu grup

### Uygulamalar-özellikler

MQ ileti özellikleri ile eşdeğerdir.

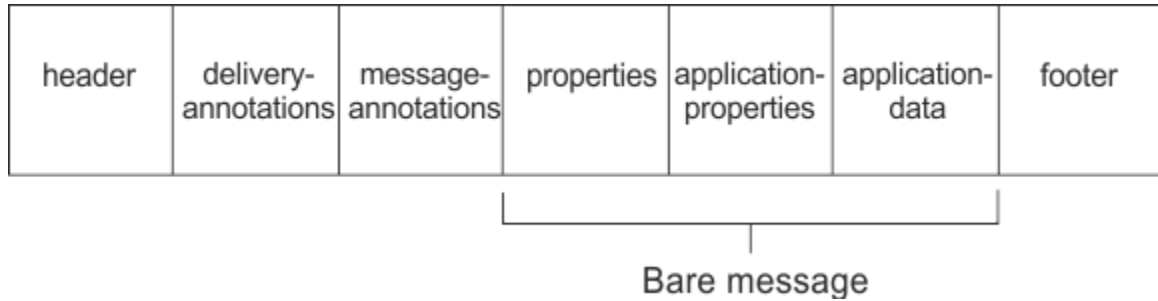
### Gövde

MQ kullanıcı bilgi yükünün eşdeğeridir.

### Altbilgi

İsteğe Bağlı. Altbilgi, yalnızca tüm yalın ileti oluşturulduktan sonra ya da görüldükten sonra hesaplanabilen ya da değerlendirilebilecek (örneğin, ileti hes, HMAC, imzalar ve şifreleme ayrıntıları gibi) iletiyle ya da değerlendirmeye ilgili ayrıntılar için kullanılır.

AMQP ileti biçimi aşağıdaki şekilde gösterilmektedir:



Özellikler, uygulama-özellikleri ve uygulama-veri bölümü "yalın ileti"olarak bilinir. Bu ileti, gönderen tarafından gönderilen ve değişmez olduğu iletidir. Alıcı, üstbilgi, altbilgi, teslim-ek açıklamalar ve ileti-ek açıklamalar da içinde olmak üzere tüm iletiyi görür.

AMQP 1.0 ileti biçimiyle ilgili tam açıklama için <https://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf>adresindeki OASIS Standardı (OASIS Standardı) başlıklı konuya bakın.

### İlgili görevler

AMQP kanallarının yaratılması ve kullanılması

AMQP istemcilerinin güvenliğini sağlama

Bir IBM MQ iletisi yayınlandığında ve IBM MQ bunu bir AMQP tüketicisine gönderdiğinde, IBM MQ iletisinin bazı özniteliklerini eşdeğeri AMQP ileti özniteliklerine geçirecektir.

## üstbilgi

Üstbilgi yalnızca, üstbilgideki beş alandan biri varsayılan olmayan bir değer içeriyorsa içerilir. Üstbilgiye yalnızca, varsayılan olmayan bir değer içeren alanlar eklenir. Beş üstbilgi alanı başlangıçta eşdeğer mq\_amqp.Hdr özelliğinden türetilir (ayarlandıysa) ve aşağıdaki çizelgede gösterildiği gibi değiştirildiyse:

<i>Çizelge 105. Üstbilgi alanı eşlemeleri</i>		
Alan	Varsayılan değer	Değer
dayanıklı	yanlış	MQMD.Persistence değeri MQPER_PERSISTENT ise true, tersi durumda false (yanlış).
öncelik	4	From mq_amqp.Hdr.Pri, if set, or otherwise from MQMD.Priority, if set. Her ikisi de ayarlanmazsa, 4 değerine ayarlayın.
ttl	yok	MQMD.Expiry (milisaniye). MQMD.Expiry değeri MQII_UNSI NSIT ise, AMQP ttl alanı için değer üst sınırı değerine ayarlanır.
Birinci alıcı banka	yanlış	mq_amqp.Hdr.Fac, ayarlandıysa, tersi durumda false (yanlış) ya da false.
teslim sayısı	0	mq_amqp.Hdr.Dct' dan (set) ayarlanmışsa, tersi durumda 0 değeri belirleyin.

## teslim-ek açıklama

AMQP kanalı için gereken şekilde ayarlayın.

## ileti-ek açıklama

Dahil değil.

## özellikler

Bunlar ayarlandıysa, **özellikler**, eşdeğer mq\_amqp.Prp özelliklerinden değiştirilmeden gelir. İleti ilk olarak bir AMQP iletisi değilse (yani, PutApplTipi MQAT\_AMQP değilse), aşağıdaki tabloda açıklandığı gibi bir özellikler bölümü oluşturulur:

<i>Çizelge 106. Özellikler alanı eşlemeleri</i>	
Ad	Değer
ileti-tnt	MQMD.MsgId ikili olarak ayarlıdır.
kullanıcı-tnt	MQMD.UserIdentifier ' in UTF-8 biçimi, ağ bayt sırasının ikili olarak ayarlıdır.
-	İletinin bulunduğu kuyruk ya da bir yayın için konu dizgisi.
subject	Ayarlanmamış.
yanıtlama	The MQMD.ReplyToQ if non-blank, otherwise not set.
ilinti tanıtıcısı	The MQMD.CorrelId is set as binary if non-blank, otherwise not set.



Çizelge 106. Özellikler alanı eşlemeleri (devamı var)

Ad	Değer
İçerik Tipi	Ayarlanmamış.
içerik kodlaması	Ayarlanmamış.
mutlak süre bitimi-süre	Ayarlanmamış.
yaratma zamanı	MQMD.PutDate ve MQMD.PutTime alanları, zaman damgası oluşturmak için kullanılır.
grup-tnt	Ayarlanmamış.
grup-sırası	Ayarlanmamış.
yanıtlama-grup-tanıtıcı	Ayarlanmamış.

## uygulama-özellikleri

"usr" grubundaki tüm IBM MQ özellikleri, **uygulama-özellikleri** olarak eklenir.

## gövde

AMQP kanalı, IBM MQ bilgi yükünü UTF-8' e dönüştürmek için dönüştürme işlemi gerçekleştirir.

If the IBM MQ payload does not contain an AMQP message, then the IBM MQ payload is set in the body as a single string data section for Format MQFMT\_STRING (provided conversion to UTF-8 was successful), or as a single binary data section otherwise.

Bir AMQP biçim iletisi içerilirse, bu, gövde olarak ayarlanır. Gövde bir AMQP Dizisi ise, AMQP iletisinin öncesinde gelen IBM MQ üstbilgileri (ileti tanıtıcısında döndürülen ileti özellikleri dahil değildir), ikili değer olarak önlenir. Ters durumda, IBM MQ üstbilgileri atılır.

## altbilgi

Altbilgi içerilmedi.

### İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

### İlgili başvurular

[MQMD-İleti tanımlayıcısı](#)

## AMQP alanlarını IBM MQ alanlarına eşleme (gelen iletiler)

AMQP kanalı bir ileti aldığı anda ve bunu IBM MQ' a yerleştirdiğinde, AMQP iletisinin bazı özniteliklerini eşdeğer IBM MQ ileti özniteliklerine geçirir.

Gelen bir AMQP iletisi eşlenirken aşağıdaki kısıtlamalar geçerli olur:

- Özellikler kısmındaki 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 IBM MQ iletisine yayılmaz.

Aşağıdaki alt kısımlarda, bir AMQP iletisinin IBM MQ ifadesi gösterilmektedir.

## İleti tanımlayıcısı

Çizelge 107. AMQP iletisi için ileti tanımlayıcısı	
Alan	Değer
StrucId	MQMD_STRUC_ID
S\u00fcr\u00fcm	MQMD_VERSION_1
Rapor	MQRO_NONE
MsgType	MQMT_DATAGRAM
Son kullanma tarihi	AMQP ileti üstbilgisindeki ttl alanından alınan değer
Geribildirim	MQFB_YOK
Kodlama	MQENC_NORMAL
CodedCharSetId	1208 (UTF-8)
Biçim	Bkz. Bilgi Yüğü
Öncelik	AMQP ileti üstbilgisindeki priority alanından alınan değer. Ayarlanırsa, en çok 9 ile sınırlanır. Ayarlanmazsa, varsayılan değer olan 4 değerini alır.
Kalıcılık	AMQP ileti üstbilgisindeki durable alanı true değerine ayarlanırsa, MQPER_PERSISTENT olarak ayarlayın. Ters durumda, MQPER_NOT_PERSISTENT olarak ayarlayın.
MagId	Kuyruk yöneticisi benzersiz bir 24 baytlık MsgIdayıyor.
Koreli	Değer belirlendiyse, AMQP özelliklerinde correlation-id alanından alınan değer. 24 baytlık bir ikili değere ayarlayın. Ters durumda, MQCI_NONE/ değerine ayarlayın.
BackoutCount	0
ReplyToQ	""
ReplyToQMgr	""
UserIdentifier	AMQP kanalına bağı olan kimliğı doğrulanmış kullanıcının tanıtıcısını ayarla
AccountingToken	MQACT_NONE
AppIdentityVerileri	Onaltılı dizilim. AMQP kanalının MQ bağılantı tanıtıcısının son 8 baytına ayarlayın.
PutApplTipi	MQAT_AMQP
PutApplAdı	
PutDate	Değer belirlendiyse, AMQP özelliklerinin creation-time alanından alınan değer. Ters durumda, yürürlükteki tarihe ayarlanır.
PutTime	Değer belirlendiyse, AMQP özelliklerinin creation-time alanından alınan değer. Ters durumda, yürürlükteki saate ayarlanır.
ApplOriginVerileri	""

## İleti Özellikleri

İleti özelliklerinin ayarlanmasını sağlamak için iki neden vardır:

- AMQP iletisinin bazı kısımlarına, iletinin bilgi yükünü etkilemeden kuyruk yöneticisi aracılığıyla akmasına izin vermek için.
- application-properties' in seçilmesine izin vermek için.

Aşağıdaki çizelge, AMQP iletisinden ayarlanan özellikleri göstermektedir:

<i>Çizelge 108. AMQP ileti özellikleri</i>			
Özellik adı	MQRFH2 adı	Tip	Tanım
AMQPListener	mq_amqp.Lis	MQTYPE_STRING	AMQP kanalı için tanımlayıcı bir dize. İletiyi oluşturmak için kullanılır, böylece ilgili taraflara hangi sürümün iletiyi yerleştirdiğini (örneğin, sorunları tanımlarken hizmet ekibi) söyleyebilmeleri gerekir. Değer, kuyruk yöneticisi tarafından doğrulanmaz ve dışarıdan belgelendirilmemelidir.
AMQPVersion	mq_amqp.Ver	MQTYPE_STRING	AMQP iletisinin sürümü. Yoksa, "1.0" varsayılan değer olarak kabul edilir. Değer, kuyruk yöneticisi tarafından doğrulanmaz.
AMQPClient	mq_amqp.Cli	MQTYPE_STRING	API için tanımlayıcı bir dize. Bu ileti, AMQP iletisini kanala göndermek için kullanılır; böylece, ilgili taraflara hangi sürümün iletiyi yerleştirdiğini (örneğin, sorunları tanımlarken hizmet ekibi) söyleyebilmeleri gerekir. Değer, kuyruk yöneticisi tarafından doğrulanmaz ve dışarıdan belgelendirilmemelidir.
AMQPDUable	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	Bir dizgi olarak ayarlandıysa, AMQP özelliklerinde message-id alanının değeri.
		MQTYPE_BYTE_STRING	AMQP özelliklerinde, byte dizgisi olarak ayarlanırsa, message-id alanının değeri.

Çizelge 108. AMQP ileti özellikleri (devamı var)			
Özellik adı	MQRFH2 adı	Tip	Tanım
AMQPUserId	mq_amqp.Prp.Uid	MQTYPE_BYTE_STRING	Ayarlandıysa, AMQP özelliklerinde user-id alanının değeri.
AMQPTo	mq_amqp.Prp.To	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde to alanının değeri.
AMQPSubject	mq_amqp.Prp.Sub	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde subject alanının değeri.
AMQPReplyTo	mq_amqp.Prp.Rto	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde reply-to alanının değeri.
AMQPCorrelationId	mq_amqp.Prp.Cid	MQTYPE_STRING	Bir dizgi olarak ayarlandıysa, AMQP özelliklerinde correlation-id alanının değeri.
		MQTYPE_BYTE_STRING	AMQP özelliklerinde, byte dizgisi olarak ayarlanırsa, correlation-id alanının değeri.
AMQPContentType	mq_amqp.Prp.Cnt	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde content-type alanının değeri.
AMQPContentEncoding	mq_amqp.Prp.Cne	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde content-encoding alanının değeri.
AMQPAbsoluteExpirySaati	mq_amqp.Prp.Aet	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde absolute-expiry-time alanının değeri.
AMQPCreationTime	mq_amqp.Prp.Crt	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde creation-time alanının değeri.
AMQPGroupId	mq_amqp.Prp.Gid	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde group-id alanının değeri.
AMQPGroupSequence	mq_amqp.Prp.Gsq	MQTYPE_INT64	Ayarlandıysa, AMQP özelliklerinde group-sequence alanının değeri.
AMQPReplyToGroupId	mq_amqp.Prp.Rtg	MQTYPE_STRING	Ayarlandıysa, AMQP özelliklerinde reply-to-group-id alanının değeri.

AMQP iletisinden her uygulama özelliği, bir IBM MQ ileti özelliği olarak ayarlanır. application-properties bölümünün, aynı byte için byte 'ı yeniden oluşturulması gerekir ve bu nedenle aşağıdaki kısıtlamalar geçerli olur:

- Bir uygulama özelliği MQSETMP geçerlilik denetimi kodu tarafından reddedilirse, reddedilecek ileti gelir. Örneğin:
  - Özellik adı, MQ\_MAX\_PROPERTY\_NAME\_LENGTH uzunluğuna göre sınırlanmış.
  - Özellik adı, Java Tanıtıcıları için Java Dil Belirtilimi (Java Language Specification) tarafından tanımlanan kurallara uygun olmalıdır.
  - Ayarlanabilen belgelenen JMS özellikleri dışında, özellik adı JMS ya da usr . JMS değerine başlamamalıdır.
  - Özellik adı bir SQL anahtar sözcüğü olmamalıdır.
- Unicode karakterini ( U+002E ) içeren bir uygulama özelliği (".") iletinin reddedilmesine neden olur. Özellik, JMS tarafından kullanılan "usr" özelliklerinde ifade edilebilir olmalıdır.

- Yalnızca null, boolean, byte, short, int, long, float, double, binary ve string özellikleri desteklenir. Başka tipte bir uygulama özelliği, iletinin reddedilmesine neden olur.

## **bilgi yükü**

- Tek bir ikili veri bölümü olan bir AMQP body için, ikili veriler (AMQP bitleri hariç), IBM MQ bilgi yükü olarak MQFMT\_NONE biçimiyle birlikte ortaya konmaktadır.
- Tek bir dizgi verisi bölümü olan bir AMQP body için, dizgi verileri (AMQP bitleri hariç), IBM MQ bilgi yükü olarak MQFMT\_STRING biçimiyle konmaktadır.
- Ters durumda, AMQP body , bilgi yükünü MQFMT\_AMQP biçimiyle birlikte oluşturur.

## **İlgili görevler**

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

**ULW**

## **AMQP ile ileti teslim güvenilirliği**

IBM MQ Light API ' nın, ileti tesliminin güvenilirliğini ve AMQP uygulamalarını denetlemenizi sağlayan dört özelliği vardır.

Bu bilgiler şunlardır:

- [“Hizmet kalitesi \(QOS\)” sayfa 741](#)
- [“Abonenin otomatik olarak doğrulanması” sayfa 742](#)
- [“Abonelik süresi \(abonelik süresi\)” sayfa 742](#)
- [“İleti kalıcılığı” sayfa 742](#)

## **Hizmet kalitesi (QOS)**

MQ Light API , iki hizmet niteliği sunar:

- En çok bir kez
- En az bir kez

Yayıncıların ve abonelerin kullanmasını istediğiniz hizmetin kalitesini seçebilirsiniz.

If you are using an MQ Light client, set the client or subscribe **qos** option to *QOS\_AT\_MOST\_ONCE* or *QOS\_AT\_LEAST\_ONCE*.

Farklı bir AMQP istemcisi kullanıyorsanız, elde etmek istediğiniz hizmet kalitesine bağlı olarak, aktarma çerçevesinin **settled** özneliğini (yayıncılar için) ya da yok etme çerçevesini (aboneler için) *true* ya da *false* olarak ayarlayın.

Bir iletinin sending tarafından ne zaman bir ileti atıldığı zaman hizmet kalitesi belirler.

## **Yayınlama**

Bir yayıncı **QOS 0** (en çok bir kez) ögesini seçerse, yayıncı, ileti kopyasını atmadan önce kuyruk yöneticisinden bir onay beklemez.

Kuyruk yöneticisine yönelik bağlantı, gönderme işlemi tamamlanmadan önce başarısız olursa, ileti aboneler tarafından alınmayabilir.

Bir yayıncı **QOS 1** (en az bir kez) ögesini seçerse, yayıncı, ileti kopyasını atmadan önce ileti kuyruklarının abone kuyruklarına yazıldığını kabul etmek için kuyruk yöneticisini bekler.

Gönderme sırasında kuyruk yöneticisine yönelik bağlantı başarısız olursa, yayıncı iletiyi kuyruk yöneticisine yeniden bağlandıktan sonra yeniden gönderir.

## **abone olunması**

Bir abone **QOS 0** ögesini seçerse, kuyruk yöneticisi iletinin kopyasını atmadan önce aboneden bir onay beklemesini beklemez.

Aboneye yönelik bağlantı, abonenin iletiyi almasından önce başarısız olursa, bu ileti kaybolabilir.

Bir abone **QOS 1** 'i seçerse, kuyruk yöneticisi iletinin kopyasını atmadan önce aboneden bir alındı bildirimini bekler.

Aboneye yönelik bağlantı, abonenin iletiyi almasından önce başarısız olursa, ileti kuyruk yöneticisi tarafından alıkonur. Kuyruk yöneticisi, kuyruk yöneticisi yeniden bağlandığında iletiyi aboneye ya da abonelik paylaşıyorsa başka bir aboneye gönderir.

## Abonenin otomatik olarak doğrulanması

Bir abone, **QOS 1** (en az bir kez) seçeneğini seçerse, kuyruk yöneticisi kopyayı atmadan önce her iletinin alınmasını onaylamalıdır. Abonenin, iletileri ne zaman onaylayacağına karar verebilir.

**auto-confirm** değeri *true* değerine ayarlandıktan sonra, istemci iletiyi ağ üzerinden başarıyla aldıktan sonra, MQ Light istemcisi her iletinin teslimini otomatik olarak kabul eder.

Bu, bir ağ hatası olduğunda, iletinin uygulamaya yeniden teslim edilmesini sağlar. Ancak, uygulama iletiyi kabul eden MQ Light istemcisi arasında başarısız olursa ve uygulamayı işleyen uygulama arasında başarısız olursa, uygulamanın iletiyi kaybetmesi yine de mümkündür.

**auto-confirm** değeri *false* değerine ayarlandığında, MQ Light istemcisi iletinin teslimini otomatik olarak onaylamaz, ancak ne zaman onaylanacağına karar vermek için uygulamayı uygulamaya bırakır.

Bu, bir uygulamanın, iletinin artık işlendiği ve atılabilmesi için kuyruk yöneticisine kabul edilmeden önce, bir veritabanı ya da dosya gibi bir dış kaynağa güncelleme gerçekleştirmesini sağlar.

## Abonelik süresi (abonelik süresi)

Bir uygulama abone olduğunda, aboneliğin ve iletilerin o abonelik için saklanıp saklanmayacağını seçer, uygulama bağlantısını kestikten sonra var olmaya devam eder.

MQ Light abone olma seçeneği **ttl**, uygulamanın bağlantıyı kestikten sonra aboneliğin var olmaya devam ettiği süreyi (milisaniye cinsinden) belirtmek için kullanılır. Uygulama bu zamandan önce yeniden bağlanırsa, abonelik sürdürülür ve uygulama bu abonelikten iletileri tüketmeye devam edebilir.

Uygulama yeniden bağlanmadan canlı zaman dönemi geçirirse, abonelik kaldırılır ve kalıcı iletiler olsa bile, hedef üzerinde saklanan iletiler kaybedilir.

İletileri kaybetmemeniz önemliyse, uygulama için bir zaman kaybı değeri belirlemeniz gerekir; bu değer, iletiler bir kesinti sırasında kaybedilmediğinden emin olmak için yeterince yüksek olur.

## İleti kalıcılığı

İletilerin kalıcılığı, yayınlama ve abone olma uygulamaları ve IBM MQ konu nesnelerinin yapılandırılarak denetlenir.

AMQP abonesi **QOS 0** (en çok bir kez) kullanıyorsa ve kalıcı olmayan bir abonelik yarattıysa, AMQP kanalı, aşağıdaki metinde açıklanan diğer seçenekler dikkate alınmaksızın, her zaman için kalıcı olmayan iletileri abone kuyruğuna koyar.

Kuyruk yöneticisi hem abonelik durdurursa, hem de iletiler kaybedilecekse, bunu not edin.

AMQP yayıncısı, AMQP  **durable** üstbilgisini *true* olarak ayarlarsa, AMQP kanalı kalıcı iletileri abone kuyruklarına yerleştirir.

Herhangi bir nedenle kuyruk yöneticisi durdurulduysa, kuyruk yöneticisi yeniden başlatıldığında, iletiler abonelere kullanılabilir olmaya devam eder.

**durable** üstbilgisi ayarlanmadıysa, AMQP kanalı, yayınlanan iletilerin kalıcılığına, ilgili IBM MQ konu nesnesinin **DEFPSIST** özniteliğine dayalı olarak kalıcılık seçeneğini belirler.

Varsayılan olarak bu, *Hayır* 'un **DEFPSIST** özniteliğini kullanan (kalıcı olmayan) **SYSTEM.BASE.TOPIC** 'dir.



**Uyarı:** MQ Light istemcisinin sonraki sürümleri, AMQP dayanıklı üstbilgisini ayarlanmasını desteklemez.

## İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

## ULW IBM MQ ile AMQP istemcileri için topolojiler

IBM MQ ile çalışmak üzere AMQP istemcilerinizi geliştirmenize yardımcı olan örnek topolojiler.

### İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

## ULW AMQP clients communicating over IBM MQ

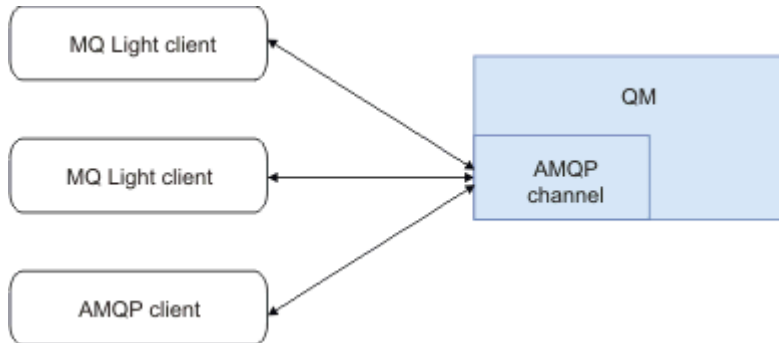
You can use IBM MQ as the messaging provider for IBM MQ Light or any application that complies with AMQP 1.0. AMQP 1.0 istemcisi bir AMQP kanalına bağlanabilirse de, bazı AMQP özellikleri desteklenmez; örneğin, işlemler ya da birden çok oturum.

Bir ya da daha çok AMQP kanalı tanımlayarak, AMQP 1.0 istemcileri kuyruk yöneticisine bağlanabilir ve bir konu dizgisine ileti gönderebilir. İstemciler, örüntüle eşleşen iletileri almak için bir konu örüntülerine de abone olabilirler.

Aşağıdaki senaryoda, ileti gönderen ve alan tek uygulamalar MQ Light ya da AMQP 1.0 uygulamalarıdır.

Uygulamalar, bir konu dizgisine abone olarak yaratılan hedeflerin kalıcı olup olmadığını seçebilir; böylece, uygulama kuyruk yöneticisiyle olan bağlantısını geçici olarak kaybederse, bu iletiler kaybolmaz.

Ayrıca, uygulamalar hedeften temizlenmeden önce iletilerin ne kadar süreyle alıkonacağını da seçebilir.



### İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

## ULW AMQP clients exchanging messages with IBM MQ applications

Bir AMQP kanalı tanımlayarak ve başlatarak, MQ Light ya da AMQP 1.0 uygulamaları, var olan MQ uygulamaları tarafından alınan iletileri yayınlayabilir. Bir AMQP kanalı aracılığıyla yayınlanan iletiler MQ kuyruklarına değil, MQ konularına gönderilir. An MQ application that has created a subscription using the MQSUB API call receives messages published by AMQP 1.0 applications, provided that the topic string or topic object used by the MQ application matches the topic string published by the AMQP client.

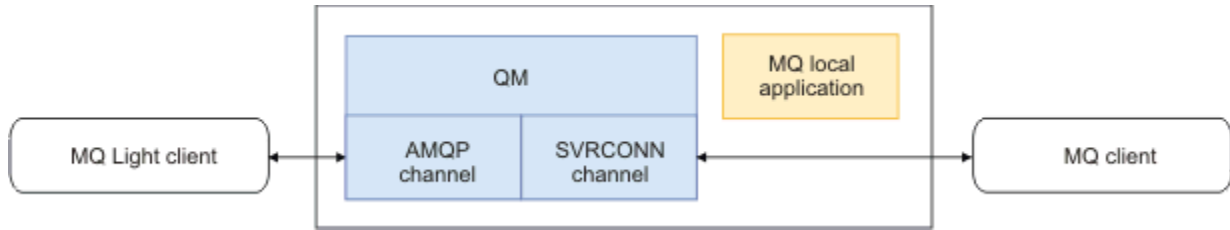
AMQP iletileri verileri, öznitelikleri ve özellikleri MQ uygulaması tarafından alınan MQ iletisine ayarlanır. AMQP ile MQ iletileri eşlemeleri hakkında daha fazla bilgi için bakınız: [Mapping AMQP fields to IBM MQ fields \(gelen iletiler\)](#).

MQ uygulaması dayanaklı bir abonelik oluşturduysa, AMQP uygulaması tarafından yayınlanan iletiler, aboneliği destekleyen kuyruklarda saklanır. The messages are then received by the MQ application when the application resumes its subscription. AMQP uygulaması canlı olarak bir ileti süresi belirtiyorsa ve MQ uygulaması zaman içinde yeniden bağlantı kurulamazsa, iletilerin süresi kuyruktan dolur.

MQ Light ya da AMQP 1.0 uygulamaları, var olan MQ uygulamaları tarafından yayınlanan iletileri de tüketebilirler. Uygulamanın, yayınlanan konu dizgisiyle eşleşen bir konu örüntüyle abone olması koşuluyla, MQ uygulamaları tarafından bir MQ konusuna ya da konu dizgisine yayınlanan iletiler, bir AMQP 1.0 uygulaması tarafından alınmakta.

AMQP 1.0 uygulaması, abonelik için bir zaman aşımı değeri belirtiyorsa ve AMQP uygulaması, yaşamak için süreden büyük bir değere bağlıysa, abonelik süresi sona ermiş olur ve abonelik kuyruğunda saklanan iletiler kaybedilir.

MQMD alanları, ileti özellikleri ve uygulama verileri, AMQP uygulaması tarafından alınan AMQP iletisine ayarlanır. MQ ile AMQP ileti eşlemeleriyle ilgili daha fazla bilgi için bkz. [AMQP alanlarını IBM MQ alanları \(giden iletiler\) üzerine eşleme](#).



### İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

## ULW IBM MQ kuyruklarındaki uygulamalarla doğrudan etkileşimde bulunabilmek için AMQP istemcilerinin yapılandırılması

IBM MQ AMQP somutlaması yalnızca yayınlama/abone olma özelliğini destekler. AMQP istemcileri, doğrudan IBM MQ kuyruklarına ileti koyamaz ya da bu kuyruklardan ileti alamıyor. AMQP istemcilerinin varolan bir kuyruğa ileti göndermesine izin vermek ya da bir kuyruğa koyulan iletileri almak için AMQP istemcilerini abone etmek için bazı ek nesnelere yaratmanız gerekir.

### Genel Bakış

For example, assume there is an application getting messages from an input queue IN\_QUEUE and putting those messages to an output queue OUT\_QUEUE. It is possible for AMQP clients to put messages to IN\_QUEUE, and get messages from OUT\_QUEUE

**Not:** Uygulamanın kendisi için herhangi bir değişiklik gerekli değildir.



Bir AMQP yayıncısı için, bir kuyruğa ileti yerleştirmesi için, AMQP istemcisinin yayınlandığı konu dizgisi için, amaçlanan kuyruğun bir hedefi olan bir yönetici aboneliği yaratmanız gerekir; bkz. [“Uygulamaya ileti gönderiliyor:” sayfa 745](#).

Bir kuyruktan ileti almak için bir AMQP abonesi için, AMQP istemcisinin abone olduğu konu dizgisini simgeleyen bir konu nesnesiyle aynı adı taşıyan bir diğer ad kuyruğuyla birlikte kuyruğu değiştirmeniz gerekir; bkz. [“Uygulamadan ileti alınıyor:” sayfa 745](#)



## Uygulamaya ileti gönderiliyor:

Uygulama zaten IN\_QUEUE ' tan iletileri alıyor ve bir AMQP istemcisinin, bu kuyruğa uygulama tarafından işlenmek üzere iletileri yayınlatabilmesi için iletileri yayınlatabilmesini istiyor.

Bunu yapmak için, bu aboneliğin ileti aldığı konu dizisinin, AMQP istemcisinin yayınladığı konu dizisiyle birlikte yeni bir yönetim aboneliği oluşturursun. The destination queue of this subscription is the input queue for the application, IN\_QUEUE.

Any messages that are published to the defined topic string for that administrative subscription are routed to the defined destination, in this case IN\_QUEUE.

Assuming the AMQP client publishes to a topic string /application/in, you can create an administrative subscription APP\_IN, using the following MQSC command:

```
DEF SUB(APP_IN) TOPICSTR('/application/in') DEST('IN_QUEUE')
```

When you have defined this object, all messages published to /application/in are routed to the destination IN\_QUEUE, where they are picked up by the application in the same way as any other messages put to this queue by other applications.

## Uygulamadan ileti alınıyor:

Uygulama, diğer istemciler tarafından alınabileceği ve işlenebileceği OUT\_QUEUE ' e ileti yerleştiriyor.

Ancak, bu durumda, iletilerin bunun yerine bir AMQP istemcisine teslim edilmesini istersiniz, ancak AMQP istemcileri yalnızca yayınlama/abone olma seçeneğini kullanır ve doğrudan bir kuyruktan ileti seçemez.

Daha önce abone olunan AMQP istemcisiyle ileti alan istemcileri değiştirmek için, AMQP istemcisinin abone olduğu konu dizisi ve diğer ad kuyruğu için bir konu nesnesi yaratmanız gerekir.



**Uyarı:** Diğer ad kuyruğunu tanımlayıp, daha sonra, herhangi bir AMQP istemcisine abone olma şansına sahip olmadan önce üretme uygulamasını başlarsanız, hiçbir abone olmadığı için, üreten uygulamanın "kuyruğa" (şimdi bir konu) göndereceği iletiler kaybedilir.

Bu metinde açıklanan değişiklikler, yalnızca abone olunan AMQP istemcisiyle ileti alan istemcilerin yerini alır. İletileri almak için AMQP ve diğer istemcilerin bir bileşimini kullanmak için daha kapsamlı değişiklikler gerekir.

Assuming the AMQP client subscribes to a topic string /application/out, you can define the topic object APP\_OUT using the following MQSC command:

```
DEF TOPIC(APP_OUT) TOPICSTR('/application/out')
```

Bu konu nesnesine teslim edilen iletiler, aynı konu dizisine abone olan AMQP istemcisine teslim edilir.

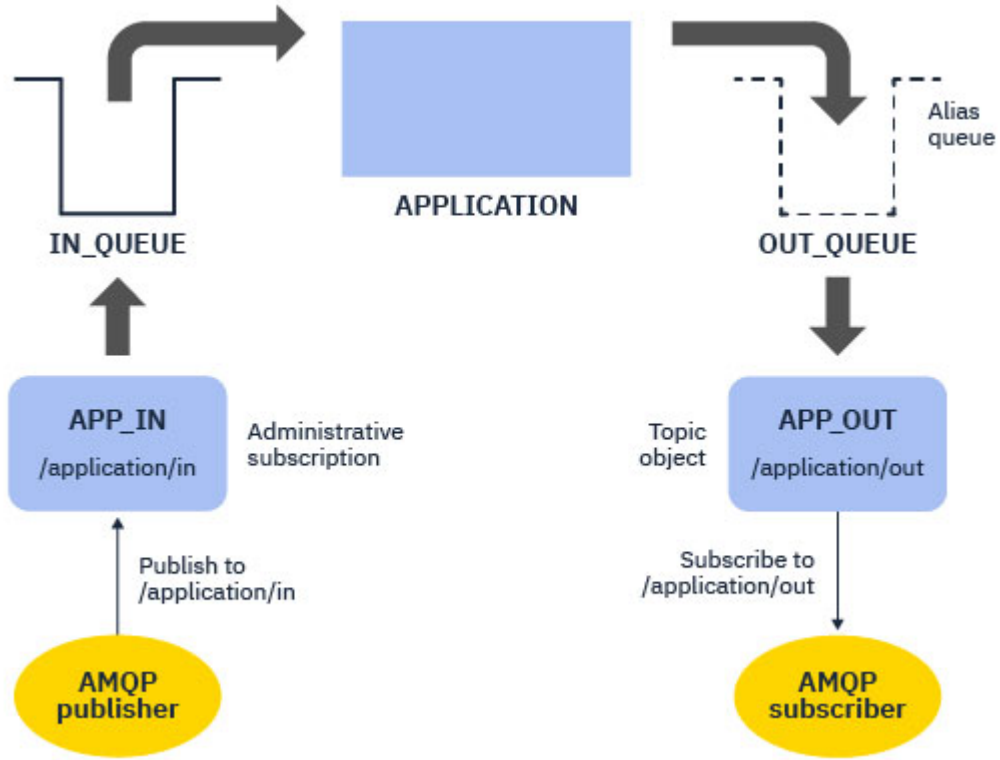
Daha sonra, uygulama tarafından OUT\_QUEUE ' a gönderilen iletilerin, abone olan istemciye gönderilmeleri için bu yeni konu nesnesine teslim edildiğinden emin olmanız gerekir.

To do this, replace the existing queue OUT\_QUEUE with an alias queue of the same name, with a target type of the topic object just created, using the following MQSC command:

```
DEF QALIAS(OUT_QUEUE) TARGTYPE(TOPIC) TARGET(APP_OUT)
```

Şimdi, uygulamanın OUT\_QUEUE ' e koyduğu iletiler kuyruğun çekilmesini beklemez; bunun yerine, bu diğer ad kuyruğunun hedefine ulaştırılır; yani, yeni konu nesnesi APP\_OUT.

Bu konu nesnesi /application/outtarafından temsil edilen konu dizisine abone olan AMQP istemcisi, diğer ad kuyruğundan bu konu nesnesine gönderilen iletileri alır.



### İlgili görevler

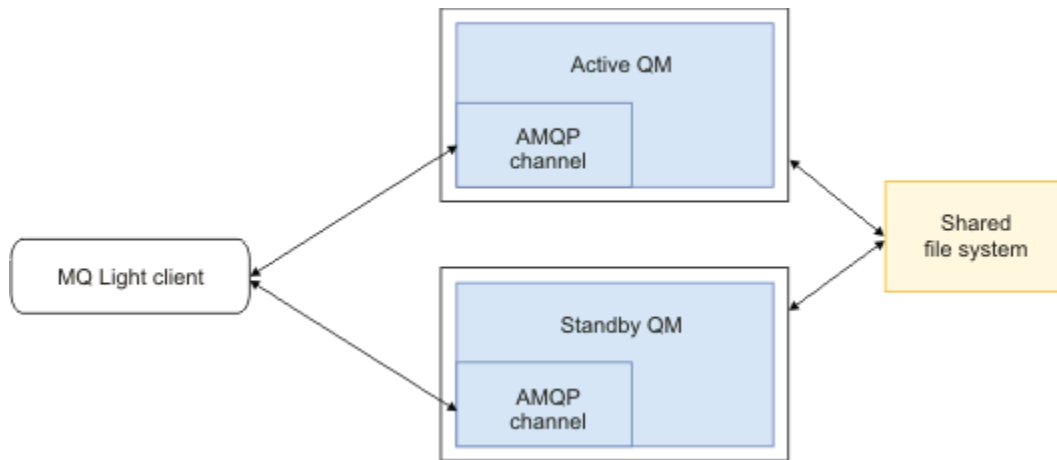
AMQP kanallarının yaratılması ve kullanılması

AMQP istemcilerinin güvenliğini sağlama

### ULW Yüksek kullanılabilirlik için bir AMQP istemcisi yapılandırılması

MQ Light ya da AMQP 1.0 uygulamalarını, IBM MQ çok eşgörümlü bir kuyruk yöneticisinin etkin eşgörümlerine bağlamak ve yüksek kullanılabilirlikli (HA) çift çiftindeki çok eşgörümlü kuyruk yöneticisinin yedek yönetim ortamına geçebilmek için yapılandırabilirsiniz. Bunu yapmak için, AMQP uygulamasını iki IP adresi ve kapı çiftiyle yapılandırınız.

MQ Light API 'y' özel bir işleyle yapılandırabilirsiniz; bu işlem, istemci sunucuya olan bağlantısını kaybederse çağrılır. İşlev alternatif bir IP adresine (örneğin, beklemedeki bir IBM MQ kuyruk yöneticisi ya da özgün IP adresi) bağlanabilir. Diğer AMQP istemcileri için, istemci birden çok bağlantı uç noktasının yapılandırılmasını destekliyorsa, uygulamayı iki anasistem-kapı çiftiyle yapılandırın ve AMQP kitaplığı tarafından sağlanan yeniden bağlanma özelliklerini, beklemedeki kuyruk yöneticisine geçmek için kullanın.



## İlgili görevler

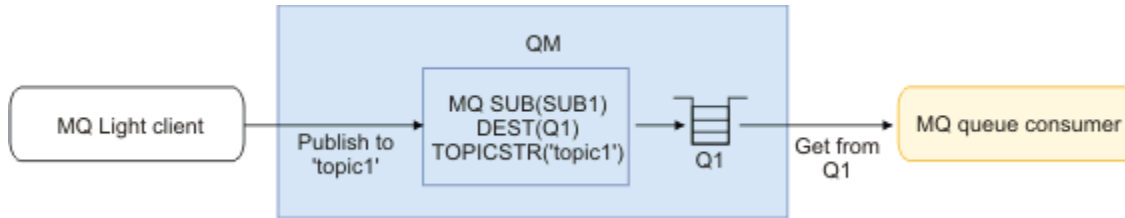
[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

## ULW AMQP istemcileri için yayınlama/abone olma özelliğinin yapılandırılması

AMQP istemcileri, iletileri var olan bir uygulama tarafından okunan bir IBM MQ kuyruğuna yönlendiren bir IBM MQ aboneliğine sahip bir konuyu yayınlabilir. If you want an MQ Light or AMQP 1.0 application to send messages to an existing IBM MQ application that is configured to read from a queue, you must define an administered IBM MQ subscription on the queue manager.

Aboneliği, AMQP uygulaması tarafından kullanılan konu dizgisiyle eşleşen bir konu kalıbını kullanacak şekilde yapılandırın. Abonelik hedefini, IBM MQ uygulamasının iletileri aldığı ya da göz attıran kuyruğun adına ayarlayın.



## İlgili görevler

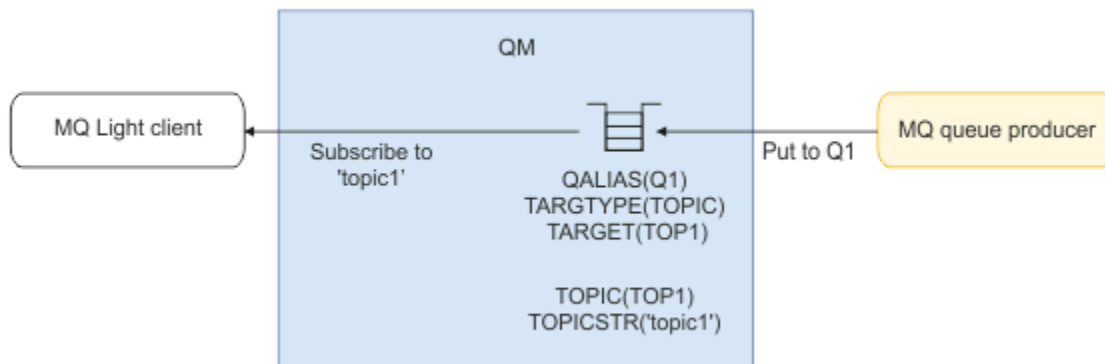
[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

## ULW Bir IBM MQ uygulamasından ileti almak için kuyruk diğer adını kullanan ASMQP istemcisi

An AMQP client can subscribe to a topic and receive messages put to an alias queue by an IBM MQ application. If you want an MQ Light or AMQP 1.0 application to receive messages from an existing IBM MQ application that is configured to put messages on a queue, you must define a queue alias (QALIAS) on the queue manager.

The queue alias must have the same name as the queue that the IBM MQ application opens for putting. Kuyruk diğer adı, AMQP uygulamasının abone olduğu konu örüntüyle eşleşen bir konu dizisine sahip bir temel nesne (TOPIC) ve bir IBM MQ konu nesnesinden oluşan temel nesne belirtmelidir.



## İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

## ULW Bir uygulama sunucusundan gelen yanıtları ve yanıt alan yanıtları içeren bir MQP istemcisi

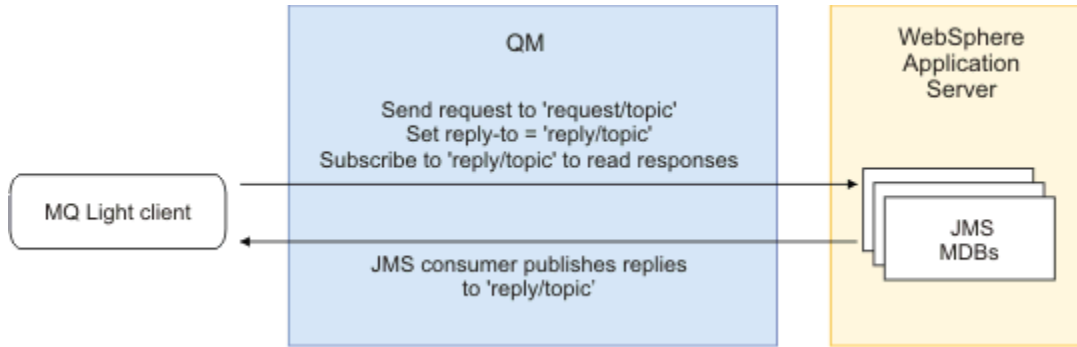
Bir MQ Light ya da diğer AMQP istemcisi, bir uygulama sunucusunda çalışan ileti odaklı bir bean 'e istek gönderebilir ve yanıt konusundan yanıtları tüketebilirler. IBM MQ , AMQP 1.0 uygulamalarını, IBM MQ ' un yayınladığı iletilerde bir yanıtlama konusu ayarlamayı destekler. Bir AMQP iletisi, yanıtlama özneliği kümesiyle birlikte yayınlandığında, yanıtlanacak yanıt alanının değeri, JMS tüketicilerinin alması için JMS özelliği olarak ayarlanır. Bu ayar, JMS kullanıcılarının iletiden gelen yanıtı okumasını ve AMQP istemcisine bir yanıt iletisi göndermesini sağlar.

JMS özelliği **JMSReplyTo**. AMQP yanıtlama dizilimi, aşağıdaki tiplerden biri olmalıdır:

- Bir konu dizgisi. Örneğin, 'reply/topic'
- An AMQP address URL in the form `amqp://host:port/[topic-string]`. Örneğin, `amqp://localhost:5672/reply/topic`

If you specify an AMQP address URL as the reply-to field, everything except the topic-string at the end of the URL is removed before setting the **JMSReplyTo** property.

Bir AMQP yanıtından bir **JMSReplyTo** özelliğine eşlemelerle ilgili daha fazla bilgi için bakınız: [Mapping AMQP fields to IBM MQ fields \(gelen iletiler\)](#)



### İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

## ULW Deploying MQ Light applications to an on-premises IBM MQ environment

IBM MQ supports the IBM MQ Light messaging API, so you can use IBM MQ to deploy your MQ Light application to an on-premises IBM MQ environment.

MQ Light uygulamalarını bir IBM MQ kuyruk yöneticisiyle konuşlandırabilirsiniz; MQ Light uygulamalarının, aşağıdaki şemada gösterildiği gibi, önceden IBM MQ ile bağlantılı olan var olan kurumsal uygulamalarla iletişim kurmasını sağlar:



MQ Light uygulamaları, var olan IBM MQ uygulamalarıyla bir konu alanını paylaşabilir ve bu, var olan kurumsal sistemlerle etkileşimli çalışabilmelerini sağlar.

### İlgili görevler

[AMQP kanallarının yaratılması ve kullanılması](#)

[AMQP istemcilerinin güvenliğini sağlama](#)

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 göndermek için messaging REST API komutunu kullanabilirsiniz. messaging REST API varsayılan olarak etkindir.

Ortak ileti özelliklerini ayarlamak için kullanılacak 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 APIkomutunu kullanarak ileti alışverişi” sayfa 750. 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 Connect EE

IBM z/OS Connect EE (zCEE), CICS ya da IMS işlemleri ve IBM MQ kuyrukları ve konuları gibi var olan z/OS varlıkları üzerinde REST API ' leri oluşturmanızı 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.

zCEE ile, JSON verilerini gönderip alacak ve isteğe bağlı olarak bu verileri birçok ana bilgisayar uygulaması tarafından beklenen daha geleneksel dil yapılarına dönüştürecek bir REST API oluşturabilirsiniz. Örneğin, COBOL kopya kitapları.

Eclipse tabanlı bir API düzenleyicisi, sorgu değiştirgelerinden ve URL yol bölümlerinden yararlanan kapsamlı bir RESTful API oluşturmak için kullanılabilir ve JSON biçimini zCEE yürütme ortamı boyunca akar.

zCEE , IBM MQ kuyruklarını ve konularını hizmet olarak göstermek için kullanılabilir. İki farklı hizmet tipi desteklenir:

- Tek yönlü hizmetler: Sağlanan işlev, HTTP POST 'un bir ileti gönderdiği, HTTP DELETE' in yıkıcı bir şekilde bir ileti aldığı HTTP için IBM MQ köprüsü ile sağlanan işleve benzer. Köprü üzerindeki temel avantajlar, yerleşik veri dönüştürme desteğidir ve daha kapsamlı bir RESTful API oluşturmak için API düzenleyicisini kullanabilirsiniz.
- İki yönlü hizmetler: Bu, arka uç istek yanıt biçimi uygulaması tarafından kullanılan bir kuyruk çiftinin üzerinde bir REST API sağlar. Çağrılar, iki yönlü hizmete HTTP POST işlemi gerçekleştirir ve JSON verilerini gönderir. Bu veriler, arka uç uygulaması tarafından işlendiği ve yanıt kuyruğuna bir yanıt konduğu bir istek kuyruğuna yerleştirilir. Bu yanıt alınır ve çağrının POST yanıt gövdesi olarak geri gönderilir.

zCEE , IBM MQ 8 ve sonraki yayın düzeylerinde desteklenir. Daha fazla bilgi için bkz. [IBM MQ for z/OS Service Provider for z/OS Connect.](#)

## 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 kullanılacak *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](#).

V 9.1.0

## REST APIkomutunu kullanarak ileti alışverişi

Basit bir noktadan noktaya ve yayınlama ileti sistemini gerçekleştirmek için messaging REST API ' i kullanabilirsiniz. İletileri bir konuya yayınlayabilir, bir kuyruğa ileti gönderebilir, kuyruklardaki iletilere göz atabilir ve iletileri, kuyruktan yok edici olarak alabilirsiniz. Information is sent to, and received from, the messaging REST API in plain text format.

## Başlamadan önce

### Not:

- messaging REST API varsayılan olarak etkindir. Tüm ileti alışverişlerini önlemek için messaging REST API ' i devre dışı bırakabilirsiniz. messaging REST API ' ı etkinleştirme ya da devre dışı bırakma hakkında daha fazla bilgi için bkz. [messaging REST API' in yapılandırılması](#).
- messaging REST API , IBM MQ güvenliği ile bütünleştirilmiştir. messaging REST API' u kullanmak için kullanıcıların kimliğinin mqweb sunucusunda kimliği doğrulanmalıdır ve MQWebUser rolünün bir üyesi olmalıdır. Kullanıcının, belirtilen kuyruğaya da konuyaerişme yetkisi de olmalıdır. REST APIgüvenlik güvenliğiyle ilgili ek bilgi için [IBM MQ Console and REST API security](#) başlıklı konuya bakın.
- messaging REST APIile Advanced Message Security (AMS) kullanıyorsanız, tüm iletilerin, iletiyi gönderen kullanıcının bağlamından değil, mqweb sunucusunun bağlamı kullanılarak şifrelendiğini unutmayın.
- When receiving veya göz atma a message only IBM MQ MQSTR formatted messages are supported. Daha sonra, tüm iletiler zamanuyumlu olarak zamanuyumlu olarak alınır ve işlenmeyen iletiler kuyruğun üzerinde bırakılır. IBM MQ kuyruğu, bu zehirli iletileri alternatif bir hedefe taşımak için yapılandırılabilir. Daha fazla bilgi için bkz. ["IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi"](#) sayfa 209.
- messaging REST API , yalnızca işlemsel destek içeren iletilerin teslimi için size bir kez ve bir kez vermez. If an HTTP POST is issued and the connection fails before an HTTP response is received by the client, the client can not immediately tell if the message was sent to the specified queue, ya da belirtilen konuyla ilgili olarak yayınlandı. Bir HTTP DELETE komutu verilirse ve istemci tarafından bir HTTP yanıtı

alınmadan bağlantı başarısız olursa, bir ileti, yıkıcı bir şekilde kuyruktan alınmış ve kaybedilmiş olabilir; bu durumda, yıkıcı geri alma işlemi geri döndürülemez.

- Gelen dizgilerdeki yeni satırlar, HTTP POST işleminde çıkarılır. REST uygulamaları

REST API 'si kullanılarak gönderilen ya da yayınlanan iletilerde yeni satırlar kullanılmamalıdır; bunlar kaybolacak şekilde.

## Yordam

- [“messaging REST API ile çalışmaya başlama” sayfa 751](#)
- [“messaging REST API komutunu kullanma” sayfa 753](#)
- [REST API hata işleme](#)
- [REST API keşfi](#)
- [REST API ulusal dil desteği](#)

## İlgili başvurular

[Messaging REST API başvurusu](#)

## İlgili bilgiler

[Eğitmen: IBM MQ Messaging REST API ile çalışmaya başlayın](#)

V 9.1.0

## messaging REST API ile çalışmaya başlama

messaging REST API ile hızlı bir şekilde çalışmaya başlayın ve cURL komutunu kullanarak birkaç örnek komutu deneyin.

## Başlamadan önce

messaging REST API komutunu kullanmaya başlamanız için, bu görevdeki örneklerin aşağıdaki gereksinimleri vardır:

- Örnekler, iletileri bir kuyruktan almak ve kuyruktan almak için REST isteklerini göndermek için cURL 'yi kullanır. Bu nedenle, bu görevi tamamlamak için sisteminizde kurulu olan cURL 'ye gereksinim duyarsınız.
- The examples use a queue manager QM1. Aynı adı taşıyan bir kuyruk yöneticisi yaratın ya da sisteminizde varolan bir kuyruk yöneticisini değiştirin. Kuyruk yöneticisi, mqweb sunucusu ile aynı makinede olmalıdır.
- Bu görevi tamamlamak için, [dspmweb](#) komutunu kullanabilmek üzere belirli ayrıcalıklara sahip bir kullanıcı olmanız gerekir:

- **z/OS** z/OS işletim sistemlerinde, [dspmweb](#) komutunu çalıştırma ve mqwebuser.xml kütüğüne yazma erişiminiz olmalıdır.
- **Multi** Diğer tüm işletim sistemlerinde [ayrıcalıklı kullanıcı](#) olmanız gerekir.

IBM i

IBM üzerinde, komutların QSHELL içinde çalıştırılması gerekir.

## Yordam

1. Ensure that the mqweb server is configured for the messaging REST API:

- mqweb sunucusu administrative REST API, MFT, messaging REST API ya da IBM MQ Console için administrative REST API tarafından kullanılmak üzere yapılandırılmadıysa, mqweb sunucusunu yapılandırın. Temel bir kayıt dosyasıyla mqweb sunucusu için temel yapılandırma oluşturma hakkında daha fazla bilgi için bkz. [mqweb sunucusu için temel yapılandırma](#).
- mqweb sunucusu önceden yapılandırıldıysa, 5. adımda ileti sistemini etkinleştirmek için uygun kullanıcıları eklediğinizden emin olun.

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* , cırtmqweb' e geçirilen dizinin adıdır. Örneğin:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

Daha fazla bilgi için bkz. [mqweb sunucusu yaratılması](#).

### 3. Aşağıdaki komutu girerek REST API URL ' yi belirleyin:

```
dspmqweb status
```

Aşağıdaki adımlardaki örnekler, REST API URL ' nin varsayılan URL <https://localhost:9443/ibmmq/rest/v1/olduğunu> varsayar. URL 'iniz varsayılan değerden farklıysa, aşağıdaki adımlarda URL ' yi değiştirin.

### 4. Create a queue, MSGQ, on queue manager QM1. Bu kuyruk ileti alışverişi için kullanılır. Aşağıdaki yöntemlerden birini kullanın:

- mqadmin kullanıcısı olarak kimlik doğrulamak için administrative REST API kaynağının queue kaynağında bir POST isteği kullanın:

```
curl -k https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1/queue -X POST -u mqadmin:mqadmin -H "ibm-mq-rest-csrf-token: value" -H "Content-Type: application/json" --data "{\"name\": \"MSGQ\"}"
```

- MQSC komutlarını kullan:

**z/OS** z/OS üzerinde, **runmqsc** komutu yerine bir 2CR kaynağı kullanın. Daha fazla bilgi için bkz. [z/OS üzerinde MQSC komutları yayınlayabileceğiniz kaynaklar](#).

- Start **runmqsc** for the queue manager by entering the following command:

```
runmqsc QM1
```

- Kuyruğu yaratmak için **DEFINE QLOCAL** MQSC komutunu kullanın:

```
DEFINE QLOCAL(MSGQ)
```

- Aşağıdaki komutu girerek **runmqsc** ' dan 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. Substitute your user where myuser is used:

- z/OS** z/OS'ta:

- Kuyruğa kullanıcı erişiminizi verin:

```
RDEFINE MQQUEUE h1q.MSGQ UACC(NONE)
PERMIT h1q.MSGQ CLASS(MQQUEUE) ID(MYUSER) ACCESS(UPDATE)
```

- Kuyruğun tüm bağlamını ayarlamak için mqweb 'in görev kullanıcı kimliği erişimini başlatmasını isteyin:

```
RDEFINE MQADMIN h1q.CONTEXT.MSGQ UACC(NONE)
PERMIT h1q.CONTEXT.MSGQ CLASS(MQADMIN) ID(mqwebStartedTaskID) ACCESS(CONTROL)
```

- Multi** Diğer tüm işletim sistemlerinde, kullanıcılarınız mqm grubunda yer aldıysa, yetki zaten verilir. Ters durumda, aşağıdaki komutları girin:



- a. Start **runmqsc** for the queue manager by entering the following command:

```
runmqsc QM1
```

- b. Kullanıcı göz atma, sorgulama, alma ve kuyruğa alma yetkilerinizi vermek için **SET AUTHREC** MQSC komutunu kullanın:

```
SET AUTHREC PROFILE(MSGQ) OBJTYPE(Queue) +  
PRINCIPAL(myuser) AUTHADD(BROWSE, INQ, GET, PUT)
```

- c. Aşağıdaki komutu girerek **runmqsc** ' dan çıkın:

```
end
```

6. Put a message with the content Hello World! on the queue MSGQ on queue manager QM1, by using a POST request on the message resource. Substitute your user ID and password from the mqwebuser.xml for myuser and mypassword:

Basic authentication is used, and an `ibm-mq-rest-csrf-token` HTTP header with an arbitrary value is set in the cURL REST request. Bu ek üstbilgi, POST, PATCH ve DELETE istekleri için gereklidir.

```
curl -k https://localhost:9443/ibmmq/rest/v1/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. Destructively get the message from queue Hello World! on the queue MSGQ on queue manager QM1, by using a DELETE request on the message resource. Substitute your user ID and password from the mqwebuser.xml for myuser and mypassword:

```
curl -k https://localhost:9443/ibmmq/rest/v1/messaging/qmgr/QM1/queue/MSGQ/message -X DELETE  
-u myuser:mypassword -H "ibm-mq-rest-csrf-token: value"
```

Hello World! iletisi döndürüldü.

## Sonraki adım

- Örnekler, isteğin güvenliğini sağlamak 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 Console ile istemci sertifikası kimlik doğrulamasını kullanma ve REST API ile belirteç tabanlı kimlik doğrulamasını kullanma](#).
- Learn more about using the messaging REST API and constructing URLs with query parameters: [“messaging REST API komutunu kullanma” sayfa 753](#).
- **V9.1.2** messaging REST API' u kullandığınızda, performans eniyilemek için kuyruk yöneticisine yönelik bağlantılar havuza gönderilmektedir. Havuz büyüklüğü üst sınırını yapılandırabilir ve havuzdaki tüm bağlantılar kullanımda olduğunda hangi işlem alınabileceğini yapılandırabilirsiniz: [messaging REST API' i yapılandırma](#).
- Kullanılabilir messaging REST API kaynakları ve kullanılabilir tüm sorgu parametreleri için başvuru bilgilerine göz atın: [messaging REST API başvurusu](#).
- IBM MQ denetimi için RESTful arabirimi olan administrative REST API' u (RESTful) keşfedin: [REST API komutunu kullanarak yönetim](#).
- Discover the IBM MQ Console, a browser-based GUI: [IBM MQ Console komutunu kullanarak yönetim](#).

## **V9.1.0** messaging REST API komutunu kullanma

messaging REST API' i kullandığınızda, IBM MQ iletileri göndermek ve almak için URL' lerde HTTP yöntemlerini çağırıyorsunuz. HTTP yöntemi (POST gibi), URL ile gösterilen nesnede gerçekleştirilecek işlem tipini temsil eder. İşlem hakkında daha fazla bilgi sorgu parametrelerinde kodlanabilir. İşlemin gerçekleştirilmesiyle ilgili bilgi, HTTP yanıtının gövdesi olarak döndürülebilir.

## Başlamadan önce

messaging REST API' u kullanmadan önce şu şeyleri göz önünde bulundurun:

- messaging REST API' i kullanabilmek için mqweb sunucusu ile kimlik doğrulaması gerekir. HTTP temel kimlik doğrulaması, istemci sertifikası kimlik doğrulaması ya da belirteç tabanlı kimlik doğrulaması kullanarak kimliğinizi doğrulayabilirsiniz. Bu kimlik doğrulama yöntemlerinin nasıl kullanılacağı hakkında daha fazla bilgi için [IBM MQ Console ve REST API security](#) başlıklı konuya bakın.
- REST API , büyük/küçük harfe duyarlıdır. Örneğin, kuyruk yöneticisi qmgr1olarak adlandırıldıysa, aşağıdaki URL ' deki bir HTTP POST işlemi hatayla sonuçlanır.

```
/ibmmq/rest/v2/messaging/qmgr/QMGR1/queue/Q1/message
```

- IBM MQ nesne adlarında kullanılacak karakterler, bir URL ' de doğrudan kodlanabilir değil. Bu karakterleri doğru bir şekilde kodlamak için uygun URL kodlamasını kullanmanız gerekir:
  - Eğik çizgi %2Folarak kodlanmalıdır.
  - Yüzde imi, %25olarak kodlanmalıdır.
  - Nokta, %2Eşeklinde kodlanmalıdır.
  - Bir soru işareti %3Folarak kodlanmalıdır.
- When receiving veya göz atma a message only IBM MQ MQSTR formatted messages are supported. Daha sonra, tüm iletiler zamanuyumlu olarak zamanuyumlu olarak alınır ve işlenmeyen iletiler kuyruğun üzerinde bırakılır. IBM MQ kuyruğu, bu zehirli iletileri alternatif bir hedefe taşımak için yapılandırılabilir. Daha fazla bilgi için bkz. [“IBM MQ classes for JMS içinde zehirli iletilerin işlenmesi” sayfa 209.](#)

## Bu görev hakkında

When you use the REST API to perform a messaging action on an IBM MQ queue object, you first need to construct a URL to represent that object. Her URL, isteğin gönderileceği anasistem adını ve kapıyı tanımlayan bir önekle başlar. URL ' nin geri kalanı belirli bir nesneyi tanımlar ya da bir kaynak olarak bilinen bu nesneye yönelir.

Kaynak üzerinde gerçekleştirilecek ileti alışverişi işlemi, URL ' nin sorgu parametrelerinin gerekip gerekmediğini tanımlar mı, yoksa sorgu değiştiricileri mi yoksa? Ayrıca, kullanılan HTTP yöntemini ve URL adresine ek bilgilerin gönderilip gönderilmediğini ya da bu yöntemin döndürdüğü bilgileri de tanımlar. Ek bilgiler, HTTP isteğinin bir parçası olabilir ya da HTTP yanıtının bir parçası olarak döndürülebilirler.

URL 'yi oluşturduktan sonra, HTTP isteğini IBM MQ' e gönderebilirsiniz. İsteğinizin programlama diline yerleşik HTTP uygulamasını kullanarak isteği gönderebilirsiniz. İsteği, cURLgibi komut satırı araçlarını ya da bir web tarayıcısı ya da web tarayıcısı eklentisi gibi kullanarak da gönderebilirsiniz.

**Önemli:** You must, as a minimum, carry out steps [“1.a” sayfa 754](#) and [“1.b” sayfa 754](#).

## Yordam

1. URL ' yi oluşturun:

a) Aşağıdaki komutu girerek URL öneğini belirleyin:

```
dspmweb status
```

Kullanmak istediğiniz URL, /ibmmq/rest/ sözcük grubunu içerir.

b) URL yoluna ileti sistemi için kullanılacak kuyruğu ve ilişkili kuyruk yöneticisi kaynaklarını ekleyin.

In the messaging reference, the variable segments can be identified in the URL by the braces that surround it { }. For further information, see [/message/qmgr/{qmgrName}/queue/{queueName}/message](#).

For example, to interact with queue Q1 associated with queue manager QM1, add /qmgr and /queue to the prefix URL to create the following URL:

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message
```

c) İsteğe bağlı: URL ' ye isteğe bağlı bir sorgu parametresi ekleyin.

Soru işareti ekleme,?, sorgu parametresi, eşittir işareti = ve bir değer URL ' ye değer.

Örneğin, bir sonraki iletinin kullanılabilir duruma gelmesi 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 daha fazla isteğe bağlı sorgu parametreleri ekleyin.

URL 'ye ve URL' ye ve işareti ekleyin ve adım 1c' yi yineleyin.

2. URL ' de ilgili HTTP yöntemini çağırın. İsteğe bağlı olan herhangi bir ileti bilgi yükünü belirtin ve kimlik doğrulaması için uygun güvenlik kimlik bilgilerini sağlayın. Örneğin:

- Seçtiğiniz programlama dilinizin HTTP/REST somutlamasını kullanın.
- REST istemci tarayıcısı eklentisi ya da cURL gibi bir araç kullanın.

## Developing MQI applications with IBM MQ

IBM MQ , C, Visual Basic, COBOL, Assembler, RPG, pTALve PL/I için destek sağlar. Bu yordamsal diller, ileti kuyruğa alma hizmetlerine erişmek için ileti kuyruğu arabirimini (MQI) kullanır.

Uygulamalarınızı seçtiğiniz dilde nasıl yazılabilmeye ilişkin ayrıntılı bilgi için alt başlıklara bakın.

Yordamsal dillere ilişkin çağrı arabirimine genel bakış için Çağrı açıklamaları başlıklı konuya bakın. Bu konu, MQI çağrılarının bir listesini içerir ve her bir çağrı size bu dillerin her birindeki çağrılar kodlarının nasıl kodlanacağını gösterir.

IBM MQ , uygulamalarınızı yazmanıza yardımcı olacak veri tanımlama dosyaları sağlar. Tam açıklama için bkz. “IBM MQ veri tanımlama dosyaları” sayfa 756.

Programlarınızı kodlamak istediğiniz yordamsal dili seçmenize yardımcı olmak için, programlarınızın işleyeceği iletilerin uzunluk üst sınırını göz önünde bulundurun. Programlarınız yalnızca bilinen bir uzunluk üst sınırına ilişkin iletileri işleyecekse, bunları desteklenen dillerden herhangi birinde kodlayabilirsiniz. Programların işlemesi için gereken iletilerin uzunluk üst sınırını bilmiyorsanız, seçtiğiniz dil CICS, IMS ya da toplu iş uygulaması mı yazıyorsanız, bu dil için değişir.

### IMS ve toplu iş

Programları C, PL/I ya da çevirici dilinde kodlamak için, bu dillerin isteğe bağlı bellek miktarlarını elde etmek ve serbest bırakmak için sunduğu olanakların kullanılmasını sağlar. Diğer bir seçenek olarak, programlarınızı COBOL ' de kodlayabilirsiniz, ancak depolama alanını almak ve serbest bırakmak için çevirici dili, PL/I ya da C alt kenar çizgileri kullanabilirsiniz.

### CICS

Programları, CICS tarafından desteklenen herhangi bir dilde kodlayın. EXEC CICS arabirimi gerekirse, belleği yönetmek için gereken çağrılar sağlar.

### İlgili kavramlar

“Nesne yönelimli uygulamalar” sayfa 14

IBM MQ , JMS, Java, C + +, .NET ve ActiveX için destek sağlar. Bu diller ve çerçeveler, IBM MQ çağrılarını ve yapılarıyla aynı işlevselliği sağlayan sınıfları sağlayan IBM MQ Object Model ' i kullanır.

Teknik genel bakış

“Uygulama geliştirme kavramları” sayfa 7

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz.

IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.

## İlgili başvurular

[Uygulama geliştirme başvurusu](#)

## IBM MQ veri tanımlama dosyaları

IBM MQ , uygulamalarınızı yazmanıza yardımcı olacak veri tanımlama dosyaları sağlar.

Veri tanımlama dosyaları aşağıdaki gibi de bilinir:

Dil	Veri tanımlamaları
C	Dosyaları ya da üstbilgi dosyalarını dahil et
Visual Basic	Modül dosyaları (yalnızca 32 bitlik sürümler)
COBOL	Dosyaları kopyala
Çevirici	Makrolar
PL/I	İçerme dosyaları

Kanal çıkışları yazmanıza yardımcı olacak veri tanımlama dosyaları, [IBM MQ COPY, HEADER, include, and module files](#) içinde anlatılır.

Kurulabilir hizmet çıkışlarını yazmanıza yardımcı olacak veri tanımlama dosyaları [“Kullanıcı çıkışları, API çıkışları ve IBM MQ kurulabilir hizmetleri”](#) sayfa 968 içinde açıklanmıştır.

C + + üzerinde desteklenen veri tanımlama dosyaları için bkz. [Using C++](#).

### IBM i

RPG ' de desteklenen veri tanımlama dosyaları için, [IBM i Application Programming Reference \(ILE/RPG\)](#) belgesine bakın.

Veri tanımlama dosyalarının adları CMQ öneğine ve programlama diline göre belirlenen bir sonektir:

Sonek	Dil
a	Çevirici dili
b	Visual Basic
c	C
l	COBOL (kullanıma hazırlanmamış değerler)
p	PL/I
v	COBOL (varsayılan değerler kümesi ile)

## Kuruluş kitaplığı





The name **thlqual** is the high-level qualifier of the installation library on z/OS.

Bu konuda, aşağıdaki başlıklar altında IBM MQ veri tanımlama dosyaları tanıtılır:

- [“C dili, dosyaları içerir” sayfa 757](#)
- [“Visual Basic modül dosyaları” sayfa 757](#)
- [“COBOL kopya dosyaları” sayfa 757](#)
- [z/OS “System/390 çevirici dili makroları” sayfa 758](#)
- [z/OS “PL/I içerme dosyaları” sayfa 759](#)

## C dili, dosyaları içerir

The IBM MQ C include files are listed in [C üstbilgi dosyaları](#). Bunlar, aşağıdaki dizinlere ya da kitaplıklara kurulur:

Platform	Kuruluş dizini ya da kitaplığı
 IBM i	QMQM/H
 UNIX	<i>MQ_INSTALLATION_PATH/inc/</i>
 Windows sistemleri	<i>MQ_INSTALLATION_PATH\Tools\c\include</i>
 z/OS	<b>thlqual.SCSQC370</b>

Burada *MQ\_INSTALLATION\_PATH*, IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder.

**Not:** UNIX için, içerme dosyaları sembolik olarak `/usr/include` ile bağlantılıdır.

Dizinlerin yapısı hakkında daha fazla bilgi için [Planning file system support](#) başlıklı konuya bakın.

## Visual Basic modül dosyaları

IBM MQ for Windows, dört Visual Basic modülü dosyası sağlar.

Bunlar [Visual Basic modül dosyaları](#) içinde listelenir ve içinde kurulur.


```
MQ_INSTALLATION_PATH\Tools\Samples\VB\Include
```

## COBOL kopya dosyaları



COBOL için, IBM MQ, adlandırılmış değişmezleri içeren ayrı kopya dosyaları ve her bir yapı için iki kopya dosyası sağlar.

Her bir yapı için iki kopya dosyası vardır. Bunun nedeni, her ikisinin de başlangıç değerleri olması ve başlangıç değerleri olmaksızın sağlanması:

- Bir COBOL programının ÇALIŞMA-STORAGE Bölümünde, yapı alanlarını varsayılan değerlere ilk kullanıma hazırlatan dosyaları kullanın. Bu yapılar, V harfi (değerler) ile suffixed adlarına sahip kopya dosyalarda tanımlanır.
- Bir COBOL programının LINKUB Bölmesinde, başlangıç değerleri olmayan yapıları kullanın. Bu yapılar, L (bağ) harfi ile suffixed adlarına sahip olan kopyalarda tanımlanır.

 Copy files containing data and interface definitions for IBM i are provided for ILE COBOL programs using prototyped calls to the MQI. Dosyalar QMQM/QCBLLESRC içinde L (başlangıç değerleri olmayan yapılar için) soneki ya da V soneki (başlangıç değerleri olan yapılar için) olan üye adlarıyla var olur.

IBM MQ COBOL kopya dosyaları, [COBOL COPY dosyaları](#) içinde listelenir. Bunlar aşağıdaki dizinlere kurulur:

Platform	Kuruluş dizini ya da kitaplığı
 Diğer UNIX altyapıları	<i>MQ_INSTALLATION_PATH/inc/</i>
 IBM i	QMQM/QCBLLESRC

Platform	Kuruluş dizini ya da kitaplığı
Windows	<i>MQ_INSTALLATION_PATH</i> \Tools\cobol\copybook (Micro Focus COBOL için) <i>MQ_INSTALLATION_PATH</i> \Tools\cobol\copybook\VAcobol ( IBM VisualAge COBOL için)
z/OS	<b>thlqual.SCSQCOBC</b>

*MQ\_INSTALLATION\_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Programınıza yalnızca gereksinim duyduğunuz dosyaları ekleyin. Bunu bir level-01 bildiriminden sonra bir ya da daha çok COPY deyimiyle gerçekleştirin. Bu, gerekirse bir programdaki yapıların birden çok sürümünü de içerebileceğiniz anlamına gelir. CMQV ' nin büyük bir dosya olduğunu unutmayın.

CMQMDV kopyalama dosyasını içermek için COBOL kodunun bir örneği aşağıda yer alır:

```
01 MQM-MESSAGE-DESCRIPTOR.  
COPY CMQMDV.
```

Her yapı bildirimini bir level-01 ögesiyle başlar; level-01 bildirimini kodlayarak, yapı bildirimini geri kalanına kopyalanacak bir COPY deyiminin ardından, yapının birkaç örneğini bildirebilirsiniz. Uygun örneğe başvurmak için IN anahtar sözcüğünü kullanın.

Aşağıda, CMQMDV ' nin iki eşgörünümünü içermek için COBOL kodu örneği bulunmaktadır:

```
* 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.
```

Yapıları 4 baytlık sınırlarla hizalayın. If you use the COPY statement to include a structure following an item that is not the level-01 item, ensure that the structure is a multiple of 4-bytes from the start of the level-01 item. Bunu yapmazsanız, uygulamanızın performansını düşürebilirsiniz.

Yapılar, MQI ' da kullanılan veri tipleri içinde açıklanmıştır. Yapılardaki alanların tanımları, öneki olmayan alanların adlarını gösterir. COBOL programlarında, alan adlarına, COBOL bildirimlerinde gösterildiği gibi, yapının adını bir tire işareti ile önek olarak ekleyin. Yapı kopyalama dosyalarındaki alanlar bu şekilde öneki olur.

Yapı kopyalama dosyalarındaki bildirimlerdeki alan adları büyük harfle karakterdir. Bunun yerine küçük harf ya da küçük harf ya da küçük harf kullanabilirsiniz. Örneğin, MQGMO yapısının *StrucId* alanı COBOL bildiriminde ve kopyalama dosyasında MQGMO-STRUCID olarak gösterilir.

V-sonek yapıları, tüm alanlar için başlangıç değerleriyle bildirilir; bu nedenle, yalnızca gereken değer başlangıç değerinden farklı olduğu alanları ayarlamanız gerekir.

## System/390 çevirici dili makroları

z/OS

IBM MQ for z/OS , adlandırılmış değişmezleri içeren iki adet çevirici dil makrosu ve her bir yapıyı oluşturmak için bir makro sağlar.

Bunlar, [z/OS Assembler COPY files](#) içinde listelenir ve **thlqual.SCSQMACS** ' de kurulur.

Bu makroların adı şu şekilde kullanılarak çağrılır:

```
MY_MQMD CMQMDA EXPIRY=0,MSGTYPE=MQMT_DATAGRAM
```

## PL/I içerme dosyaları

z/OS

IBM MQ for z/OS , PL/I içinde IBM MQ uygulamaları yazdığınızda gereksinim duyduğunuz tüm tanımlamaları içeren dosyaları içerir.

Dosyalar, [PL/I include dosyaları](#) listesinde yer alır ve **thlqual.SCSQPLIC** dizinine kurulur:

IBM MQ sınırlı kod öbeğini programınıza bağladığınızda, programınıza bu dosyaları ekleyin (bkz. “Programınızı çalıştırmak üzere hazırlama” sayfa 1059 ). IBM MQ çağrılarını dinamik olarak bağlamak istiyorsanız yalnızca CMQP ' yi dahil edin (bkz. “IBM MQ sınırlı kod öbeğini devingen olarak çağırma” sayfa 1065 ). Dinamik bağlantı oluşturma, yalnızca toplu iş ve IMS programları için gerçekleştirilebilir.

## Kuyruğa alma için bir yordamsal uygulama yazma

Kuyruk yöneticisi, yayınlama/abone olma, nesnelere açma ve kapama nesnelere bağlanma ve bağlantıyı kesme, kuyruğa alma ve bağlantı kesme hakkında bilgi edinmek için bu bilgileri kullanın.

Uygulama yazma hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- [“Message Queue Interface-Genel Bakış” sayfa 760](#)
- [“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)
- [“Nesnelerin açılması ve kapatılması” sayfa 781](#)
- [“İletileri Kuyruğa Koyma” sayfa 791](#)
- [“Kuyruktan İleti Alınması” sayfa 806](#)
- [“Yayınlama/abone olma uygulamaları yazılıyor” sayfa 845](#)
- [“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)
- [“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)
- [“Starting IBM MQ applications using triggers” sayfa 899](#)
- [“MQI ve kümelerle çalışma” sayfa 918](#)
- [z/OS “Uygulamaların IBM MQ for z/OS üzerinde kullanılması ve yazılması” sayfa 922](#)
- [z/OS “IBM MQ for z/OS üzerinde IMS ve IMS köprüsü uygulamaları” sayfa 65](#)

### İlgili kavramlar

[“Uygulama geliştirme kavramları” sayfa 7](#)

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz.

IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.

[“IBM MQ için uygulama geliştirilmesi” sayfa 5](#)

İletileri göndermek ve almak, kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için uygulamalar geliştirebilirsiniz. IBM MQ , birçok farklı dil ve çerçeveye içinde yazılmış uygulamaları destekler.

[“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#)

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğinize karar verdiğinizde, IBM MQ tarafından sunulan özelliklerin nasıl kullanılacağına karar vermeniz gerekir.

[“İstemci yordamsal uygulamaları yazılıyor” sayfa 945](#)

Bir yordamsal dil kullanarak IBM MQ üzerinde istemci uygulamaları yazmak için bilmeniz gerekenler.

[“Yordamsal uygulama oluşturulması” sayfa 1030](#)

Bir IBM MQ uygulamasını birden çok yordamsal dilden birine yazabilir ve uygulamayı farklı platformlarda çalıştırabilirsiniz.

[“Yordamsal program hatalarının işlenmesi” sayfa 1073](#)

Bu bilgiler, bir çağrı yaparken ya da iletisi son hedefine teslim edildiğinde, uygulama MQI çağrılarınızla ilişkili hataları açıklar.

## İlgili görevler

“IBM MQ örnek yordamsal programlarının kullanılması” sayfa 1092

Bu örnek programlar yordamsal dillere yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımları gösterir. Farklı platformlardaki IBM MQ programları.

## Message Queue Interface-Genel Bakış

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

İleti Kuyruğu Arabirimi aşağıdaki öğelerden oluşur:

- Programların kuyruk yöneticisine ve tesislerine erişebileceği *Çağrılar*
- queue, veri aktarmak ve kuyruk yöneticisinden veri almak için kullanılan *Yapılar*
- Veri aktarma ve kuyruk yöneticisinden veri almak için *temel veri tipleri*

**z/OS** IBM MQ for z/OS ayrıca şunları sağlar:

- z/OS toplu iş programlarının işleyebileceği ve değişiklikleri geri alabileceği iki ek çağrı.
- IBM MQ for z/OS ile sağlanan değişmezlerin değerlerini tanımlayan *Veri tanımlama dosyaları* (bazen kopya dosyaları, makrolar, içerme dosyaları ve üstbilgi dosyaları olarak bilinir).
- *Sınırlı Kod Öbeği programları*, uygulamalarınıza bağlantı düzenleme olanağı sağlar.
- A suite of sample programs that demonstrate how to use the MQI on the z/OS platform. Bu örneklerle ilgili daha fazla bilgi için bkz. “z/OS için örnek programların kullanılması” sayfa 1193.

**IBM i** IBM MQ for IBM i ayrıca şunları sağlar:

- IBM MQ for IBM i ile sağlanan değişmezlerin değerlerini tanımlayan *Veri tanımlama dosyaları* (bazen kopya dosyaları, makrolar, içerme dosyaları ve üstbilgi dosyaları olarak bilinir).
- ILE C, ILE COBOL ve ILE RPG uygulamalarınıza bağlantı düzenlemek için üç sınırlı kod öbeği programı.
- A suite of sample programs that demonstrate how to use the MQI on the IBM i platform.

UNIX and Linux sistemlerinde IBM MQ for Windows ve IBM MQ ürünleri de sağlar:

- Calls through which IBM MQ for Windows and IBM MQ on UNIX and Linux systems programs can commit and back out changes.
- *İçerme dosyaları* bu altyapılarda sağlanan değişmezlerin değerlerini tanımlar.
- Uygulamalarınızı bağlamak için *Kitaplık dosyaları*.
- Bu altyapılarda MQI 'yi nasıl kullanacağını gösteren örnek programlar grubu. Bu örneklerle ilgili daha fazla bilgi için bkz. “Örnek Programların çoklu Platformlar Üzerinde Kullanılması” sayfa 1093.
- Dış hareket yöneticilerine bağ tanımları için örnek kaynak ve yürütülebilir kod.

MQI hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- “MQI çağrıları” sayfa 761
- “Eşitleme noktası aramaları” sayfa 762
- “Veri dönüştürme, veri tipleri, veri tanımları ve yapılar” sayfa 763
- “IBM MQ kod parçası programları ve kitaplık dosyaları” sayfa 763
- “Tüm çağrılar için ortak olan parametreler” sayfa 768
- “Arabelleklerin belirtilmesi” sayfa 769
- **z/OS** “z/OS toplu iş konuları” sayfa 769
- “UNIX and Linux sinyal işleme” sayfa 770

## İlgili kavramlar

“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773



IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabılır alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OSüzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

[“IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

## **MQI çağruları**

Message Queue Interface (MQI) içindeki çağruları öğrenmek için bu bilgileri kullanın.

MQI ' deki çağrılar aşağıdaki gibi gruplanabilir:

### **MQCONN, MQCONNX ve MQDISC**

Bu çağruları kullanarak, (seçeneklerle ya da seçeneksiz) bir programı, kuyruk yöneticisinden bir program bağlantısını kesin (bir program ile). z/OSiçin CICS programları yazarsanız, bu çağruları kullanmanız gerekmez. Ancak, uygulamanızı diğer altyapılara kapılamak istiyorsanız, bunları kullanmanız önerilir.

### **MQOPEN ve MQCLOSE**

Bir nesneyi (kuyruk gibi) açmak ve kapatmak için bu çağruları kullanın.

### **MQPUT ve MQPUT1**

Bir ileti kuyruğuna ileti koymak için bu çağruları kullanın.

### **MQGet**

Bir kuyruktaki iletilere göz atmak ya da kuyruktan iletileri kaldırmak için bu çağrıyı kullanın.

### **MQSUB, MQSUBRQ**

Bir konuya abonelik kaydetmek ve abonelikle eşleşen yayınları istemek için bu çağruları kullanın.


### **MQINQ**

Bir nesnenin özniteliklerine ilişkin bilgi edinmek için bu aramayı kullanın.

### **MQSET**

Bir kuyruğun özniteliklerinin bazılarını ayarlamak için bu çağrıyı kullanın. Diğer nesne türlerinin özniteliklerini ayarlayamazsınız.

### **MQBEGIN, MQCMIT ve MQBACK**

IBM MQ , bir iş biriminin eşgüdümçüsü olduğunda bu çağruları kullanın. MQBEGIN, iş birimini başlatır. MQCMIT ve MQBACK, iş birimi sırasında yapılan güncellemeleri kesinleştirerek ya da geri döndürerek iş birimini sona erdirir.  IBM i taahhüt denetleyicisi, IBM MQ for IBM iüzerinde genel

çalışma birimlerini koordine etmek için kullanılır. Yerel başlatma kesinleştirme denetimi, kesinleştirme ve geri alma komutları kullanılır.

### **MQCRMH, MQBUFMH, MQMHBUF, MQDLTMH**

Bir ileti tanıtıcısı yaratmak, ileti tanıtıcısını bir arabelleğe ya da arabelleğe bir ileti tanıtıcısı oluşturmak ve bir ileti tanıtıcısını silmek için bu çağrılarını kullanın.

### **MQSETMP, MQINQMP, MQDLTMP**

İleti tanıtıcısı üzerinde bir ileti özelliği ayarlamak, bir ileti özelliği sorgulamak ve bir özelliği ileti tanıtıcısından silmek için bu çağrılarını kullanın.

### **MQCB, MQCB\_FUNC, MQCTL**

Geri bildirme işlevini kaydettirmek ve denetlemek için bu çağrılarını kullanın.

### **MQSTAT**

Önceki zamanuyumsuz koyma işlemlerine ilişkin durum bilgilerini almak için bu çağrıyı kullanın.

MQI çağrılarının bir açıklaması için [Çağrı açıklamaları](#) başlıklı konuya bakın.

## ***Eşitleme noktası aramaları***

Farklı platformlarda eşitleme noktası çağrılarını hakkında bilgi almak için bu bilgileri kullanın.

Eşitleme noktası aramaları aşağıdaki gibi kullanılabilir:

## **IBM MQ for z/OS çağrılar**



IBM MQ for z/OS , MQCMIT ve MQBACK çağrılarını sağlar.

Son eşitleme noktasından bu yana tüm MQGET ve MQPUT işlemlerinin kalıcı (kesinleştirilmiş) kılınacağı ya da yedekleneceğini kuyruk yöneticisine anlatmak için z/OS toplu iş programlarında bu çağrılarını kullanın. Diğer ortamlardaki değişiklikleri kesinleştirmek ve geri göndermek için:

### **CICS**

EXEC CICS SYNCPOINT ve EXEC CICS SYNCPOINT ROLLBACK gibi komutları kullanın.

### **IMS**

IOPCB, CHKP (checkpoint) ve ROLB (rollback) çağrılarında GU (benzersiz alma) gibi IMS eşitleme noktası olanaklarını kullanın.

### **RS**

MQCMIT ve MQBACK ya da SRRCMIT ve SRRBACK ' ları uygun şekilde kullanın. (Bkz. "[Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri](#)" sayfa 892.)

**Not:** SRRCMIT ve SRRBACK, yerel RRS komutlarıdır, bu komutlar MQI çağrılarını değildir.

## **IBM i çağrılar**



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 çağrılarını (örneğin, EXEC CICS SYNCPOINT gibi) de kullanabilirsiniz.

## **Windows, UNIX and Linux platformlarındaIBM MQ çağrılarını**



Aşağıdaki ürünler MQCMIT ve MQBACK çağrılarını sağlar:

- IBM MQ for Windows
- UNIX and Linux sistemlerindeIBM MQ

Son eşitleme noktasından bu yana tüm MQGET ve MQPUT işlemlerinin kalıcı (kesinleştirilmiş) kılınacağı ya da yedekleneceğini kuyruk yöneticisine anlatmak için, programlardaki eşitleme noktası çağrılarını

kullanın. CICS ortamında yapılan deęişiklikleri kesinleřtirmek ve yedeklemek için, EXEC CICS SYNCPOINT ve EXEC CICS SYNCPOINT ROLLBACK gibi komutları kullanın.

## **Veri dönüřtürme, veri tipleri, veri tanımları ve yapılar**

İleti Kuyruęu Arabirimi 'ni kullanırken veri dönüřtürmeleri, temel veri tipleri, IBM MQ veri tanımlamaları ve yapılar hakkında bilgi edinmek için bu bilgileri kullanın.

### **Veri dönüřtürme**

MQXCNCV (dönüřtürme karakterleri) çağırısı, ileti karakter verilerini bir karakter kümesinden dięerine dönüřtürür. IBM MQ for z/OS'da, bu çağrı yalnızca bir veri dönüřtürme çıkıřtan kullanılır.

MQXCNCV çağırısıyla kullanılan sözdizimine iliřkin [MQXCNCV-Karakterlerin dönüřtürülmesi](#) başlıklı konuya ve veri dönüřtürme çıkıřlarının yazılması ve çağrılmasına iliřkin yönergeler için [“Veri dönüřtürme çıkıřları yazılıyor” sayfa 1014](#) başlıklı konuya bakın.

### **Temel veri tipleri**

Desteklenen programlama dilleri için, MQI temel veri tiplerini ya da yapılandırılmamıř alanları saęlar.

Bu veri tipleri [Temel veri tipleri'](#) ta tam olarak açıklanmıřtır.

### **IBM MQ veri tanımları**

**z/OS** IBM MQ for z/OS , veri tanımlarını COBOL kopya dosyaları, birleřtirme dili makroları, tek bir PL/I içerme dosyası, tek bir C dili dosya, C++ dili ise dosyaları içerir.

**IBM i** IBM MQ for IBM i , COBOL kopya dosyaları, RPG kopya dosyaları, C dili dosyaları ve C++ dili dosyaları dahil olmak üzere veri tanımlarını saęlar.

IBM MQ ile verilen veri tanımlama dosyaları řunları içerir:

- Tüm IBM MQ deęiřmezlerinin ve dönüř kodlarının tanımları
- IBM MQ yapılarına ve veri tiplerine iliřkin tanımlar
- Yapıları kullanıma hazırlamak için kullanılan deęiřmez tanımlamaları
- Çaęrıların her biri için iřlev prototipleri (yalnızca PL/I ve C dili için)

IBM MQ veri tanımlama kütüklerinin tam tanımı için bkz. [“IBM MQ veri tanımlama dosyaları” sayfa 756.](#)

### **Yapılar**

[“MQI çağrıları” sayfa 761](#) içinde listelenen MQI çağrıları ile kullanılan yapılar, desteklenen

programlama dillerinin her biri için veri tanımlama dosyalarında saęlanır. **IBM i** **z/OS** IBM MQ for z/OS ve IBM MQ for IBM i , bu yapıların bazı alanlarını tamamlarken kullanmak üzere sabit deęerler içeren dosyaları saęlar. Bunlar hakkında daha fazla bilgi için bkz. [IBM MQ veri tanımları.](#)

Yapıların bir özeti için [Yapı veri tipleri özeti](#) başlıklı konuya bakın.

### **IBM MQ kod parçası programları ve kitaplık dosyaları**

Saęlanan sınırlı kod öbeęi programları ve kitaplık dosyaları, her platform için burada listelenir.

Yürütülebilir bir uygulama oluřtururken kod parçası programlarının ve kitaplık dosyalarının nasıl kullanılmasıyla ilgili daha fazla bilgi için bkz. [“Yordamsal uygulama oluřturulması” sayfa 1030.](#) C++ kitaplık dosyalarına baęlantı oluřturma hakkında bilgi için bkz. [Using C++ IBM MQ C++ kullanılması.](#)

**AIX** IBM MQ for AIX kitaplık dosyaları

IBM MQ for AIX' ta, programınızı, uygulamanızı çalıřtırdığınız ortam için saęlanan MQI kitaplık dosyalarına, iřletim sistemi tarafından saęlanan ortamlara baęlamanız gerekir.

İř parçacıklı olmayan bir uygulamada, ařaęıdaki kitaplıklardan birine baęlayın:

Çizelge 109. İş parçacıklı olmayan AIX uygulamaları için kitaplık dosyaları

Kitaplık dosyası	Ortam
libmqm.a	C sunucusu için sunucu
libmqic.a ve libmqm.a	C İçin İstemci
libmqmzf.a	C için kurulabilir hizmet çıkışları
libmqmxa.a	Sunucu XA arabirimi
libmqmxa64.a	Sunucu alternatifi XA arabirimi
libmqcxa.a	İstemci XA arabirimi
libmqcxa64.a	İstemci alternatif XA arabirimi
libmqmcbt.o	Micro Focus COBOL desteği için IBM MQ Runtime kitaplığı
libmqmcb.a	COBOL için sunucu
libmqicb.a	COBOL için İstemci
libimqc23ia.a	C++ için İstemci
libimqs23ia.a	C++ için sunucu

İş parçacıklı bir uygulamada, aşağıdaki kitaplıklardan birine bağlantı:

Çizelge 110. İş parçacıklı AIX uygulamalarına ilişkin kitaplık dosyaları.

Her bir kitaplık dosyası için kitaplık dosyalarını ve ortamı listeleyen iki sütunlu çizelge.

Kitaplık dosyası	Ortam
libmqm_r.a	C sunucusu için 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 alternatifi XA arabirimi
libmqcxa_r.a	İstemci XA arabirimi
libmqcxa64_r.a	İstemci alternatif XA arabirimi
libimqc23ia_r.a	C++ için İstemci
libimqs23ia_r.a	C++ için sunucu

**Not:** Birden çok kitaplığa bağlanamazsınız. Yani, aynı anda hem yivli hem de iş parçacıklı bir kitaplığa bağlanamazsınız.

#### IBM i IBM MQ for IBM i kitaplık dosyaları

IBM MQ for IBM i' ta programınızı, uygulamanızı çalıştırdığınız ortam için sağlanan MQI kitaplık dosyalarına, işletim sistemi tarafından sağlanan ortamlara bağlayın.

İş parçacıklı olmayan uygulamalar için:

<i>Çizelge 111. İş parçacıklı olmayan IBM i uygulamaları için kitaplık dosyaları</i>	
<b>Kitaplık dosyası</b>	<b>Ortam</b>
LIBMQM	Sunucu ve İstemci hizmet programı
LIBMQIC	İstemci hizmet programı
IMQB23I4	C++ temel hizmet programı
IMQS23I4	C++ sunucu hizmeti programı
LIBMQMZF	C için kurulabilir çıkışlar

Bir iş parçacıklı uygulamada:

<i>Çizelge 112. İş parçacıklı IBM i uygulamalarına ilişkin kitaplık dosyaları</i>	
<b>Kitaplık dosyası</b>	<b>Ortam</b>
<b>LIBMQM_R</b>	Sunucu ve istemci hizmet programı
<b>IMQB23I4_R</b>	C++ temel hizmet programı
<b>IMQS23I4_R</b>	C++ sunucu hizmeti programı
<b>LIBMQMZF_R</b>	C için kurulabilir çıkışlar
<b>LIBMQIC_R</b>	İstemci hizmet programı

IBM MQ for IBM i'ta uygulamalarınızı C + + ' da yazabilirsiniz. C++ uygulamalarınızın nasıl bağlanacağını görmek için ve C + + kullanmanın tüm yönleriyle ilgili tüm ayrıntılar için [Using C++](#) konusuna bakın.

**Linux** *Linux kitaplık dosyaları için IBM MQ*

Linux için IBM MQ işletim sisteminde, programınızı, uygulamanızı çalıştırdığınız ortam için sağlanan MQI kitaplık dosyalarına, işletim sistemi tarafından sağlanan ortamlara bağlamanız gerekir.

İş parçacıklı olmayan bir uygulamada, aşağıdaki kitaplıklardan birine bağlayın:

<i>Çizelge 113. İş parçacıklı olmayan Linux uygulamaları için kitaplık dosyaları</i>	
<b>Kitaplık dosyası</b>	<b>Ortam</b>
libmqm.so	C sunucusu için sunucu
libmqic.so ve libmqm.so	C İçin İstemci
libmqmzf.so	C için kurulabilir hizmet çıkışları
libmqmxa.so	Sunucu XA arabirimi
libmqmxa64.so	Sunucu alternatif 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:

Çizelge 114. İş parçacıklı Linux uygulamalarına ilişkin kitaplık dosyaları

Kitaplık dosyası	Ortam
libmqm_r.so	C sunucusu için 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 alternatifi 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 bağlanamazsınız. Yani, aynı anda hem yivli hem de iş parçacıklı bir kitaplığa bağlanamazsınız.

#### **Solaris** IBM MQ for Solaris kitaplık dosyaları

IBM MQ for Solaris' ta, programınızı, uygulamanızı çalıştırdığınız ortam için sağlanan MQI kitaplık dosyalarına, işletim sistemi tarafından sağlanan ortamlara bağlamanız gerekir.

Çizelge 115. Solaris uygulamaları için kitaplık dosyaları

Kitaplık Dosyası	Ortam
libmqm.so	C için sunucu ve istemci
libmqmzse.so	C İçin
libmqic.so	C İçin İstemci
libmqmcs.so	C için ortak hizmetler
libmqmzf.so	C için kurulabilir hizmet çıkışları
libmqmxa.so	Sunucu XA arabirimi
libmqmxa64.so	Sunucu alternatifi XA arabirimi
libmqcxa.so	İstemci XA arabirimi
libmqcxa64.so	İstemci alternatif XA arabirimi
libimqc23as.a	C++ için İstemci
libimqs23as.a	C++ için sunucu

#### **Windows** IBM MQ for Windows kitaplık dosyaları

IBM MQ for Windows' ta, programınızı, uygulamanızı çalıştırdığınız ortam için sağlanan MQI kitaplık dosyalarına, işletim sistemi tarafından sağlanan ortamlara bağlamanız gerekir:

Çizelge 116. Windows uygulamaları için kitaplık dosyaları

Kitaplık Dosyası	Ortam
MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib	Sunucu C (32 bit) için
MQ_INSTALLATION_PATH\Tools\Lib\mqic.lib	C için İstemci (32 bit)

Çizelge 116. Windows uygulamaları için kitaplık dosyaları (devamı var)

<b>Kitaplık Dosyası</b>	<b>Ortam</b>
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqmxa.lib</code>	C için sunucu XA arabirimi (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqcxa.lib</code>	C için istemci XA arabirimi (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqicxa.lib</code>	İstemci MTS for C (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqmcics4.lib</code> 32	C için sunucu TXSeries CICS desteği (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqccics4.lib</code> 32	C (32 bit) için istemci TXSeries CICS desteği
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqmzf.lib</code>	C (32 bit) için kurulabilir hizmetler çıkışı
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqmcbb.lib</code>	Server for IBM COBOL (32-bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqmcb.lib</code>	Server for Micro Focus COBOL (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqicbb.lib</code>	Client for IBM COBOL (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\mqiccb.lib</code>	Client for Micro Focus COBOL (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\imqs23vn.lib</code>	C++ için sunucu (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\imqc23vn.lib</code>	C++ için İstemci (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\imqb23vn.lib</code>	C++ için temel (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib\imqx23vn.lib</code>	C++ için İstemci MTS (32 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib</code>	Sunucu C (64 bit) için
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqic.lib</code>	C için İstemci (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqmxa.lib</code>	C için sunucu XA arabirimi (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqcxa.lib</code>	C için istemci XA arabirimi (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqicxa.lib</code>	İstemci MTS for C (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqmcbb.lib</code>	Server for IBM COBOL (64-bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqmcb.lib</code>	Server for Micro Focus COBOL (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqicbb.lib</code>	Client for IBM COBOL (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\mqiccb.lib</code>	Client for Micro Focus COBOL (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\imqs23vn.lib</code>	C++ için sunucu (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\imqc23vn.lib</code>	C++ için İstemci (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\imqb23vn.lib</code>	C++ için temel (64 bit)
<code>MQ_INSTALLATION_PATH\Tools\Lib64\imqx23vn.lib</code>	C++ için İstemci MTS (64 bit)

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

.NET programlarını derlemek için `amqmdnet.dll` kullanın. Ek bilgi için [“.NET uygulamalarının geliştirilmesi” sayfa 528](#) kısmındaki [“IBM MQ .NET programlarının derlenmesi” sayfa 578](#) konusuna 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/OS ile yazılmış bir programı çalıştırmadan önce, uygulamayı çalıştırmakta olduğunuz ortam için IBM MQ for z/OS ile birlikte sağlanan sınırlı kod öbeği programına bağlanmanız gerekir.

Sınırlı kod öbeği programı, çağrılarınızın işlenmesinin ilk aşamasını, IBM MQ for z/OS işleminin işleyebileceği isteklere ilişkin ilk aşamanızı sağlar.

IBM MQ for z/OS aşağıdaki sınırlı kod öbeği programlarını sağlar:

#### **CSQBKOD**

z/OS toplu iş programları için sınırlı kod öbeği programı

#### **CSQBRRSI**

MQI yoluyla RRS kullanan z/OS toplu iş programlarına ilişkin sınırlı kod öbeği programı

#### **CSQBRSTB**

Doğrudan RRS kullanan z/OS toplu iş programlarına ilişkin sınırlı kod öbeği programı

#### **CSQCKOPOR**

CICS programları için sınırlı kod öbeği programı

#### **CSQQKOD**

IMS programları için sınırlı kod öbeği programı

#### **CSQXSTUB**

Dağıtım kuyruğa alma dışı CICS dışı çıkışlar için sınırlı kod öbeği programı

#### **CSQASOR**

Veri dönüştürme çıkışları için sınırlı kod öbeği programı



**Uyarı:** Belirli bir ortam için listelenenden başka bir sınırlı kod öbeği programı kullanıyorsanız, bu programın önceden kestirilemeyecek sonuçlar ortaya çıktı.

**Not:** CSQBRSTB sınırlı kod öbeği programını kullanırsanız, SYS1.CSSLIB. (SYS1.CSSLIB , *Callable Services Library* (Callable Services Kitaplığı) olarak da bilinir). RRS ile ilgili daha fazla bilgi için bkz. [“Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri” sayfa 892](#).

Diğer bir seçenek olarak, sınırlı kod öbeğini programınızın içinden dinamik olarak çağırabilirsiniz. Bu teknik [“IBM MQ sınırlı kod öbeğini devingen olarak çağırma” sayfa 1065](#) içinde açıklanmıştır.

IMS' ta, IBM MQ tarafından sağlanan özel bir dil arabirimi modülünü de kullanmanız gerekebilir.

Aynı IMS MPP bölgesinde CSQBCOSS ve CSQQSTUB ile bağlantı düzenlenmiş olan uygulamaları çalıştırmayın. Bu, DFS3607I ya da CSQQ005E iletileri gibi sorunlara neden olabilir. Bir adres alanındaki ilk MQCONN çağrısı hangi arabirimin kullanılacağını belirler, bu nedenle CSQQSTUB ve CSQBKOC işlemlerinin farklı IMS ileti bölgelerinde çalışması gerekir.

### **Tüm çağrılar için ortak olan parametreler**

Tüm çağrılar için ortak iki tip deyişirge vardır: tutamaçlar ve dönüş kodları.

### **Tutamaçları kullanma**

Tüm MQI çağrıları bir ya da daha fazla *çekme noktaları* kullanır. Bu bilgiler, kuyruk yöneticisini, kuyruğu ya da diğer nesneyi, iletiyi ya da aboneliği çağrıya uygun şekilde tanımlar.

Bir programın kuyruk yöneticisiyle iletişim kurması için, programın kuyruk yöneticisini tanıdığı benzersiz bir tanıtıcıya sahip olması gerekir. Bu tanıtıcı, bazen *Hconn* olarak da adlandırılan bir *bağlantı tanıtıcısı* adı olarak adlandırılır. CICS programları için, bağlantı tanıtıcısı her zaman sıfırdır. Diğer tüm altyapılar ya



da program biçimleri için, program kuyruk yöneticisine bağlandığında, MQCONN ya da MQCONNX çağrısıyla bağlantı tanıtıcısı döndürülür. Programlar, bağlantı tanıtıcısını, diğer çağrıları kullandıklarında giriş parametresi olarak geçirir.

Bir programın bir IBM MQ nesnesiyle çalışabilmesi için, programın bu nesneyi tanıdığı benzersiz bir tanıtıcısı olmalıdır. Bu tanıtıcının adı *nesne tanıtıcısı*, bazen de *Hobj* olarak adlandırılır. Program nesneyi, onunla çalışmak için açtığında, MQOPEN çağrısı bu işleme geri döndürülür. Programlar, nesne tanıtıcısını, sonraki MQPUT, MQGET, MQINQ, MQSET ya da MQCLOSE çağrılarını kullanırken giriş değiştirgesi olarak geçirir.

Benzer şekilde, MQSUB çağrısı, sonraki MQGET, MQCB ya da MQSUBRQ çağrılarında aboneliği tanımlamak için kullanılan bir *subscription handle* ya da *Hsubd* döndürür ve bazı çağrı işleme iletisi özellikleri bir *ileti tanıtıcısı* ya da *Hmsg* kullanır.

## Dönüş kodlarının anlaşılması

Her çağrıya göre çıkış değiştirgeleri olarak bir tamamlanma kodu ve neden kodu döndürülür. Bunlar toplu olarak *dönüş kodları* olarak bilinir.


Bir çağrıyı başarılı olup olmadığını göstermek için her çağrı, arama tamamlandığında bir *tamamlama kodu* döndürür. Tamamlanma kodu genellikle, başarılı olan MQCC\_OK ya da başarısızlığı gösteren MQCC\_FAILED olur. Bazı çağrılar bir ara düzey durumu (MQCC\_UYARI) döndürebilir, bu da kısmi başarıyı gösterir.

Ayrıca, her çağrı aynı zamanda aramadaki hatanın ya da kısmi başarısının nedenini gösteren bir *neden kodu* döndürür. Bir kuyruğun dolu olması, bir kuyruk için işlemlere izin verilmemesi ve kuyruk yöneticisi için belirli bir kuyruk tanımlanmaması gibi birçok neden kodu vardır. Programlar, işleme nasıl devam edebileceğinize karar vermek için neden kodunu kullanabilir. Örneğin, kullanıcılar kullanıcıların giriş verilerini değiştirmelerini isteyebilir, daha sonra aramayı yeniden yapabilir ya da kullanıcıya bir hata iletisi döndürebilirler.

Tamamlanma kodu MQCC\_OK ise, neden kodu her zaman MQRC\_NONE olur.

Her çağrıya ilişkin tamamlanma ve neden kodları, bu çağrıya ilişkin açıklamayla listelenir. [Çağrı açıklamaları](#) başlıklı konuya bakın ve listeden uygun aramayı seçin.

Düzeltilici eylemle ilgili fikirler de içinde olmak üzere daha ayrıntılı bilgi için bkz:

-  [IBM MQ for z/OS iletileri, tamamlama, ve neden kodları](#) - IBM MQ for z/OS
- Diğer tüm IBM MQ platformları için [Messages and reason codes](#)

## Arabelleklerin belirtilmesi

Kuyruk yöneticisi, yalnızca gerekli oldukları durumlarda arabellekleri belirtir. Çağrı için bir arabelleğe gerek duymuyorsanız ya da arabelleğin uzunluğu sıfırsa, arabelleğe boş değerli bir gösterge kullanabilirsiniz.

Gereksinim duyduğunuz arabelleğin boyutunu belirlerken her zaman "datallength" değerini kullanın.

Çıkışı bir çağrıdan tutmak için bir arabellek kullandığınızda (örneğin, bir MQGET çağrısına ilişkin ileti verilerini ya da MQINQ çağrısıyla sorgulanan özniteliklerin değerlerini tutmak için), kuyruk yöneticisi belirttiğiniz arabellek geçerli değilse ya da salt okunur bir depoda olduğunda bir neden kodu döndürmeyi dener. Ancak, her zaman bir neden kodu döndüremeyebilir.

## z/OS toplu iş konuları

MQI 'yi çağırınız/OS toplu iş programları, denetmen ya da sorun durumunda olabilir.

Ancak, aşağıdaki koşulları yerine getirmeleri gerekir:

- SRB (service request block; hizmet isteği öbeği) kipinde değil, görev kipinde olmalıdır.
- Bunlar, Birincil adres alanı denetimi (ASC) kipinde olmalıdır (Erişim Kaydı ASC kipi değil).
- Bunlar, çapraz bellek kipinde olmamaları gerekir. Birincil adres alanı numarası (ASN), ikincil ASN 'ye ve ana ASN' ye eşit olmalıdır.

- Bunlar MPF çıkış programları olarak kullanılmamalıdır.
- Hiçbir z/OS kilidi tutulamaz.
- FRR yığınının herhangi bir işlev kurtarma yordamı (FRRs) olamaz.
- Herhangi bir program durumu sözcüğü (PSW) anahtarı, MQCONN ya da MQCONNX çağrısı için zorda olabilir (anahtar, TCB anahtarındaki depolamayı kullanarak uyumlu olur), ancak MQCONN ya da MQCONNX tarafından döndürülen bağlantı tanıtıcısını kullanan sonraki çağrılar:
  - MQCONN ya da MQCONNX çağrısında kullanılan PSW anahtarının aynı olması gerekir
  - Aynı PSW anahtarı altında erişilebilir parametrelere (uygun olduğunda, uygun olduğunda) sahip olmalıdır
  - Aynı görev (TCB) altında verilmeli, ancak görevin alt görevlerinde olmamalıdır
- Bunlar 24 bitlik ya da 31 bit adresleme kipinde olabilir. Ancak, 24 bit adresleme kipi yürürlükte ise, parametre adresleri 31 bitlik geçerli adresler olarak yorumlanmalıdır.

Bu koşullardan herhangi biri karşılanmazsa, program denetimi gerçekleştirilebilir. Bazı durumlarda çağrı başarısız olur ve bir neden kodu döndürülür.

### Linux → UNIX **UNIX and Linux önemli noktalar**

Dikkat etmeniz gereken noktalar.

UNIX and Linux uygulamalarını geliştirirken aşağıdaki noktaları göz önünde bulundurun.

### Linux → UNIX **The fork system call in UNIX and Linux systems**

Note these considerations when using a fork system call in IBM MQ applications.

Uygulamanız fork kullanmak isterse, o uygulamanın üst işlemi, IBM MQ çağrılarını yapmadan önce fork çağrısında olmalıdır; örneğin, MQCONN, ya da **ImqQueueManager** komutunu kullanarak bir IBM MQ nesnesi yaratılmalıdır.

Uygulamanız herhangi bir IBM MQ çağrısını yaptıktan sonra bir alt işlem yaratmak istiyorsa, uygulama kodu, alt ögenin tam kopyası değil, yeni bir yönetim ortamı olduğundan emin olmak için exec () ile bir fork () uygulaması kullanılmalıdır.

Uygulamanız exec () kullanmıyorsa, alt süreç içinde yapılan IBM MQ API çağrısı MQRC\_ENVIRONMENT\_ERROR döndürmesini sağlar.

### Linux → UNIX **UNIX and Linux sinyal işleme**

Bu, IBM MQ for z/OS ya da IBM MQ for Windows için geçerli değildir.

Genel olarak, UNIX, Linux ve IBM i sistemleri iş parçacıklı (süreç) ortamından çok iş parçacıklı bir ortama taşınmış olur. İş parçacıklı ortamda, bazı işlevler yalnızca sinyaller kullanılarak uygulanabilir, ancak çoğu uygulamanın sinyaller ve sinyal işleme konusunda haberdar olması gerekmez. Çok iş parçacıklı ortamda, iş parçacığı tabanlı temel öğeler, sinyaller kullanarak iş parçacıklı ortamlarda uygulanmış olan bazı işlevleri destekler.

Birçok örnekte, sinyaller ve sinyal işleme desteklenir, ancak desteklense de, çok iş parçacıklı ortama ve çeşitli sınırlamalara uygun değildir. Her birinin sinyalleri ele almak için çalıştığı çok iş parçacıklı bir ortamda uygulama kodunu farklı ara katman yazılımı kitaplıklarıyla bütünleştirirken (uygulamanın bir parçası olarak çalışan) bu durum sorunlu olabilir. Bir süreç içinde tek bir yürütme iş parçacığı olduğunda çalışan sinyal işleyicileri (süreç başına tanımlanan) tasarruf ve geri yüklemeye ilişkin geleneksel yaklaşım, çok iş parçacıklı bir ortamda çalışmaz. Bunun nedeni, yürütmenin birçok iş parçacığının, önceden kestirilemeyen sonuçlarla, süreç çapında bir kaynağı saklamaya ve geri yüklemeye çalışabileceği içindir.

### UNIX **İş parçacıklı uygulamalar**

Yalnızca tek bir iş parçacığı kullansalar da, tüm uygulamalar iş parçacıklı olarak kabul edildiğinden, Solaris için geçerli değildir.

Her MQI işlevi, sinyaller için kendi sinyal işleyicisini ayarlar:

SIGALRM  
SIGBUS  
SIGFPE  
SIGSEGV  
SIGILL

Bunlar için kullanıcıların işleyicileri, MQI işlev çağrısının süresine göre değiştirilir. Diğer sinyaller, kullanıcı tarafından yazılan işleyiciler tarafından normal şekilde yakalanabilir. Bir işleyici kurmadıysanız, varsayılan işlemler (örneğin, yoksay, çekirdek dökümü ya da çıkış) yerinde bırakılır.

IBM MQ , zamanuyumlu bir sinyal (SIGSEGV, SIGBUS, SIGFPE, SIGILL) işledikten sonra, MQI işlev çağrısını yapmadan önce, kayıtlı herhangi bir sinyal işleyicisine sinyal iletmeyi dener.

**Linux** **UNIX** İş parçacığı uygulamaları

MQDISC 'ye kadar, MQCONN' dan (ya da MQCONNX) IBM MQ ' e bağlı bir iş parçacığı olarak kabul edilir.

## Zamanuyumlu sinyaller

Zamanuyumlu sinyaller belirli bir iş parçacığında ortaya çıkar.

UNIX and Linux sistemleri, tüm süreç için bu tür sinyaller için bir sinyal işleyicisinin ayarlanmasına güvenli bir şekilde izin verir. Ancak, IBM MQ aşağıdaki sinyaller için kendi işleyicisini, uygulama sürecinde, herhangi bir iş parçacığı IBM MQ' a bağlıyken ayarlar:

SIGBUS  
SIGFPE  
SIGSEGV  
SIGILL

Çok iş parçacıklı uygulamalar yazıyorsanız, her sinyal için yalnızca tek bir işlem geniş sinyal işleyicisi vardır. IBM MQ , kendi zamanuyumlu sinyal işleyicilerini ayarladığında, her sinyal için önceden kaydedilmiş tüm işleyicileri kaydeder. IBM MQ , listelenen sinyallerden birini işledikten sonra, IBM MQ , işlem içindeki ilk IBM MQ bağlantısı sırasında yürürlükte olan sinyal işleyicisini aramayı dener. Önceden kayıtlı işleyiciler, tüm uygulama iş parçacıklarının IBM MQ bağlantısı kesildiğinde geri yüklenir.

Sinyal işleyicileri IBM MQ tarafından kaydedilip geri yüklendiğinden, aynı işlemin başka bir iş parçacığının IBM MQ' a da bağlı olması olasılığı varken, uygulama iş parçacıkları bu sinyaller için sinyal işleyicileri oluşturmamalıdır.

**Not:** Bir uygulama ya da bir ara katman yazılımı kitaplığı (uygulamanın bir parçası olarak çalışıyorsa), iş parçacığı IBM MQ' e bağlıyken bir sinyal işleyici kurar; uygulamanın sinyal işleyicisinin, bu sinyalin işlenmesi sırasında karşılık gelen IBM MQ işleyicisini araması gerekir.

Sinyal işleyicilerini kurarken ve geri yüklerken genel ilke, kaydedilecek son sinyal işleyicinin ilk geri yüklenebilecek ilk kişi olması gerekir:

- Bir uygulama, IBM MQ' a bağlandıktan sonra bir sinyal işleyicisi oluşturduğunda, uygulama IBM MQ' den bağlantı kesilmeden önce önceki sinyal işleyicisine geri yüklenmelidir.
- Bir uygulama, IBM MQ' a bağlanmadan önce bir sinyal işleyicisi oluşturduğunda, uygulamanın sinyal işleyicisini geri yüklemeye başlamadan önce IBM MQ ' in bağlantısını kesmesi gerekir.

**Not:** Kaydedilecek son sinyal işleyicisinin, geri yüklenebilecek ilk kişi olması, uygulamada beklenmeyen sinyal işleme ve potansiyel olarak uygulama tarafından sinyallerin kaybedilmesi ile sonuçlanabileceği genel ilkeyi gözlemlememe.

## Zamanuyumsuz sinyaller

IBM MQ , istemci uygulamaları olmadığı sürece, iş parçacıklı uygulamalarda zamanuyumsuz sinyaller kullanmaz.

## İş parçacıklı istemci uygulamalarına ilişkin ek konular

IBM MQ , bir sunucuya G/Ç sırasında aşağıdaki sinyalleri işler. Bu sinyaller, iletişim yığınının göre tanımlanır. Bir iş parçacığı kuyruk yöneticisine bağlıyken, uygulamanın bu sinyaller için bir işaret işleyici oluşturulması gerekir:

SIGPIPE (TCP/IP için)

**UNIX** MQI 'da UNIX sinyal işleme kullanılırken dikkat edilmesi gereken diğer noktalar UNIX sinyal işleme kullanılırken bu noktalara dikkat edin.

## 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 zamanuymulu 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 çağrısını kullanarak kuyruk yöneticisine bağlanma” sayfa 775.](#)

## İşaret işleyicileri içindeki MQI işlev çağrıları

İşleyici işleyicisindeyken MQI işlevini çağırmayın.

Başka bir MQI işlevi etkinken bir sinyal işleyicisinden MQI işlevini çağırılmaya çalışırsanız, MQRC\_CALL\_IN\_PROGRESS döndürülür. Başka bir MQI işlevi etkin olmadığı halde bir sinyal işleyicisinden MQI işlevini çağırılmaya 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 çağrılacak 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 çağrıları 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 çağrıları kullanmadığını göz önünde bulundurun.

## Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

Bu bağlantının yapıldığı platform ve programın çalıştığı ortama bağlıdır:

### Multi IBM MQ for Multiplatforms

Bu ortamlarda çalışan programlar, bağlanmak için MQCONN MQI çağrısını kullanabilir ve bir kuyruk yöneticisinden bağlantıyı kesmek için MQDISC çağrısına da kullanılabilir. Diğer bir seçenek olarak, programlar MQCONNX çağrısını kullanabilir.

### z/OS IBM MQ for z/OS toplu

Bu ortamda çalışan programlar, bağlanmak için MQCONN MQI çağrısını kullanabilir ve bir kuyruk yöneticisinden bağlantıyı kesmek için MQDISC çağrısına da kullanılabilir. Diğer bir seçenek olarak, programlar MQCONNX çağrısını kullanabilir.

z/OS toplu iş programları, aynı TCB ' de birden çok kuyruk yöneticisine eşzamanlı olarak ya da eşzamanlı olarak bağlanabilir.

### z/OS IMS

IMS denetim bölgesi başlatıldığında bir ya da daha çok kuyruk yöneticisine bağlanır. Bu bağlantı IMS komutları tarafından denetlenir. z/OS üzerindeki IMS bağdaştırıcısının nasıl denetleneceği hakkında bilgi için bkz. [Yönetme IBM MQ for z/OS](#). Ancak, IMS programlarını kuyruğa yollayan ileti yazarları, bağlanmak istedikleri kuyruk yöneticisini belirtmek için MQCONN MQI çağrısını kullanmalıdır. Bu kuyruk yöneticisinden kopmak için MQDISC çağrısını kullanabilirler.

Bir eşitleme noktası oluşturan IMS çağrısının ardından ve başka bir kullanıcı için bir iletiyi işlemeden önce, IMS bağdaştırıcısı, uygulamanın kuyruk yöneticisinden tutamaçları kapatmasını ve bağlantısını kesmesini sağlar. Bkz. ["IMS uygulamalarındaki eşitleme noktaları"](#) sayfa 891.

IMS programları art arda ya da koşut zamanlı olarak aynı TCB ' de kuyruk yöneticilerine bağlanabilir.

### z/OS için CICS Transaction Server

CICS programs do not need to do any work to connect to a queue manager because the CICS system itself is connected. This connection is typically made automatically at initialization, but you can also use the CKQC transaction that is supplied with IBM MQ for z/OS. CKQC ile ilgili daha fazla bilgi için bkz. [Yönetme IBM MQ for z/OS](#).

CICS görevleri, yalnızca CICS bölgesinin bağlı olduğu kuyruk yöneticisine bağlanabilir.

CICS programları ayrıca, MQI bağlantısı ve bağlantı kesme çağrılarını (MQCONN ve MQDISC) de kullanabilir. Bunu yapmak isteyebilirsiniz; böylece, bu uygulamaları CICS dışı ortamlar için en az bir recoding değerine sahip ortamlara kapılabilirsiniz. However, these calls *her zaman* complete successfully in a CICS environment. Diğer bir deyişle, dönüş kodu, kuyruk yöneticisiyle bağlantının gerçek durumunu yansıtmayabilir.

### Windows ve Open Systems için TXSeries

CICS sisteminin kendisi bağlı olduğu için, bu programların kuyruk yöneticisine bağlanmak için herhangi bir çalışma yapması gerekmez. Bu nedenle, aynı anda yalnızca bir bağlantı desteklenir. CICS applications must issue an MQCONN call to obtain a connection handle, and an MQDISC call before they exit.

Bir kuyruk yöneticisine bağlanma ve bağlantı kesme hakkında ek bilgi almak için aşağıdaki bağlantıları kullanın:

- ["MQCONN çağrısını kullanarak kuyruk yöneticisine bağlanma"](#) sayfa 774
- ["MQCONNX çağrısını kullanarak kuyruk yöneticisine bağlanma"](#) sayfa 775
- ["MQDISC kullanan bir kuyruk yöneticisinden programların bağlantısı kesiliyor"](#) sayfa 780

### İlgili kavramlar

["Message Queue Interface-Genel Bakış"](#) sayfa 760

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabılır alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OS üzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

[“IBM MQ for z/OS üzerinde IMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

## ***MQCONN çağrısını kullanarak kuyruk yöneticisine bağlanma***

MQCONN çağrısını kullanarak 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/OS için, toplu iş ortamında, CSQBDEFV modülünde varsayılan kuyruk yöneticisi belirtilir.
- IBM MQ for Windows, IBM i, UNIX ve Linux sistemleri için, varsayılan kuyruk yöneticisi mqcs.ini dosyasında belirtilir.

Diğer bir seçenek olarak, z/OS MVS batch, TSO ve RRS ortamlarında, bir kuyruk paylaşım grubu içindeki herhangi bir kuyruk yöneticisine bağlanabilirsiniz. MQCONN ya da MQCONNX isteği, grubun etkin üyelerinden herhangi birini seçer.

Bir kuyruk yöneticisine bağlandığında, görev için yerel bir kuyruk yöneticisine varmalıdır. Bu, IBM MQ uygulaması ile aynı sisteme ait olmalıdır.

In the IMS environment, the queue manager must be connected to the IMS control region and to the dependent region that the program uses. Varsayılan kuyruk yöneticisi, IBM MQ for z/OS kurulu olduğunda CSQQDEFV modülünde belirtilir.

TXSeries CICS ortamı ve Windows ve AIX için TXSeries ile, kuyruk yöneticisi CICS olarak bir XA kaynağı olarak tanımlanmalıdır.

Varsayılan kuyruk yöneticisine bağlanmak için, tümüyle boşluklardan oluşan ya da boş değerli (X'00 ') karakterlerle başlayan bir ad belirterek MQCONN' yi çağırın.

Bir uygulamanın kuyruk yöneticisine başarıyla bağlanması için bir uygulama yetkilendirilmelidir. Daha fazla bilgi için bkz. [Securing](#).

MQCONN ' den çıkış:

- Bir bağlantı tanıtıcısı ( **Hconn** )
- Tamamlanma kodu
- Neden kodu

İzleyen MQI çağrılarında bağlantı tanıtıcısını kullanın.

Neden kodu, uygulamanın o kuyruk yöneticisine önceden bağlı olduğunu gösteriyorsa, döndürülen bağlantı tanıtıcısı, uygulama ilk kez bağlandığında döndürülen bağlantı tanıtıcısı ile aynıdır. Çağırın uygulamanın bağlı kalmasını beklediğinden, uygulama bu durumda MQDISC çağrısını yayınlamamalıdır.

Bağlantı tanıtıcısı kapsamı, nesne tutamacının kapsamı ile aynıdır (bkz. "[MQOPEN çağrısını kullanarak nesnelere açılması](#)" sayfa 782 ).

Parametrelere ilişkin açıklamalar, [MQCONN](#) içindeki MQCONN çağrısının tanımında verilmiştir.

Kuyruk yöneticisi çağrısı yayınlarken ya da kuyruk yöneticisi kapatılıyorsa, MQCONN çağrısı başarısız olur ya da kuyruk yöneticisi durdurulmuş durumdaysa başarısız olur.

## **MQCONN ya da MQCONNX kapsamı**


MQCONN ya da MQCONNX çağrısının kapsamı genellikle onu yayınlayan iş parçacığıdır. Yani, çağrıdan döndürülen bağlantı tanıtıcısı yalnızca çağrısı yayınlayan iş parçacığının içinde geçerlidir. Tutamacı kullanarak herhangi bir zamanda yalnızca bir arama yapılabilir. Farklı bir iş parçacığından kullanılırsa, geçersiz olarak reddedilir. Uygulamanızda birden çok iş parçacığınız varsa ve her biri IBM MQ çağrısını kullanmak isterse, her biri MQCONN ya da MQCONNX yayınlamalıdır.

Bir işlem birden çok MQCONN çağrısını yaptığında, aynı kuyruk yöneticisine her çağrı için bu gerekli değildir. However, only one IBM MQ connection can be made from a thread at a time. Diğer bir seçenek olarak, "[Shared \(thread independent\) connections with MQCONNX](#)" sayfa 778 ' un tek bir iş parçacığından birden çok IBM MQ bağlantısına ve herhangi bir iş parçacığından IBM MQ bağlantısının kullanılmasına izin vermesi için de göz önünde bulundurun.<sup>7</sup>

Uygulamanız istemci olarak çalışıyorsa, iş parçacığı içinde birden çok kuyruk yöneticisine bağlanabiliyor.

## **MQCONNX çağrısını kullanarak kuyruk yöneticisine bağlanma**

MQCONNX çağrısı MQCONN çağrısına benzer, ancak arama çalışmalarının yolunu denetleme seçeneklerini içerir.

As input to MQCONNX, you can supply a queue manager name  Ya da z/OS paylaşılan kuyruk sistemlerinde kuyruk paylaşım grubu adı.

MQCONNX 'in çıkışı şöyledir:

- Bir bağlantı tanıtıcısı (Hconn)
- Tamamlanma kodu
- Neden kodu

İzleyen MQI çağrılarında bağlantı tanıtıcısını kullanıyorsunuz.

MQCONNX 'in tüm deęiřtirgelerinin tanımı MQCONNX içinde verilmiştir. *Options* alanı, MQCNO ' nun herhangi bir sürümü için STANDARD\_BINDING, FASTPATH\_BINDING, SHARED\_BINDING ya da ISOLATED\_BINDING olarak ayarlamınızı sağlar. Ayrıca, bir MQCONNX çağrısı kullanarak paylaşılan (iş parçacığı bağımsız) bağlantıları da yapabilirsiniz. Bunlarla ilgili daha fazla bilgi için bkz. "[Shared \(thread independent\) connections with MQCONNX](#)" sayfa 778 .

## **MQCNO\_STANDARD\_BINDING**

Varsayılan olarak, MQCONNX (MQCONN gibi), IBM MQ uygulamasının ve yerel kuyruk yöneticisi aracısının ayrı süreçlerde çalıştırıldığı iki mantıksal iş parçacığını belirtir. The IBM MQ application requests the IBM MQ operation and the local queue manager agent services the request. Bu, MQCONNX çağrısında MQCNO\_STANDARD\_BINDING seçeneęi tarafından tanımlanır.

<sup>7</sup> When using multithreaded applications with IBM MQ on UNIX and Linux systems you need to ensure that the applications have a sufficient stack size for the threads. Çok iş parçacıklı uygulamalar kendi başına ya da dięer sinyal işleyicilerle (örneğin, CICS gibi) MQI çağrıları yaparken 256 KB ya da daha büyük bir yığın boyutu kullanmayı düşünün.

MQCNO\_STANDARD\_BINDING deęerini belirlerseniz, MQCONNX çağırısı, qm.iniiçinde tanımlı olan kuyruk yöneticisinin **DefaultBindType** özneliğinin deęerine baęlı olarak MQCNO\_SHARED\_BINDING ya da MQCNO\_ISOLATED\_BINDING kullanır.

Bu varsayılan deęerdir.

mqm kitaplığına baęlantı oluřturuyorsanız, önce varsayılan baę tanımlama tipini kullanan bir standart sunucu baęlantısı giriřiminde bulunulması gerekir. Temeldeki sunucu kitaplığı yüklenemediyse, bunun yerine bir istemci baęlantısı giriřiminde bulunulması gerekir.

- MQ\_CONNECT\_TYPE ortam deęiřkeni belirtilirse, MQCB\_STANDARD\_BINDING belirtilirse, MQCONN ya da MQCONNX 'in davranıřını deęiřtirmek için ařaęıdaki seęeneklerden biri saęlanabilir. (Bu kural dıřı durum, MQCNO\_FASTPATH\_BINDING MQ\_CONNECT\_TYPE ile LOCAL ya da STANDARD deęerine ayarlandıysa, uygulama için ilgili bir deęiřiklik yapılmaksızın, fastpath baęlantılarının yönetici tarafından dūřürölmesine izin verir:

Deęer	Anlamı
CLIENT	Yalnızca istemci baęlantısı deniyor.
FastPath	Bu deęer önceki yayınlarda desteklendi, ancak belirtilirse yoksayılr.
LOCAL	Yalnızca sunucu baęlantısı denendi. Fastpath baęlantıları, standart bir sunucu baęlantısına dūřürülebilir.
Standart	Önceki yayın düzeyleriyle uyumluluk için desteklenir. Bu deęer řimdi LOCAL olarak ele alınır.

- MQCONN çağırıldığında MQ\_CONNECT\_TYPE ortam deęiřkeni ayarlanmazsa, varsayılan baę tanımlama tipini kullanan bir standart sunucu baęlantısı giriřiminde bulunulması denir. Sunucu kitaplığı yüklenemezse, bir istemci baęlantısı giriřiminde bulunmaya çalıřılır.

## MQCNO\_FASTPATH\_BINDING

*Güvenilen uygulamalar* , IBM MQ uygulamasının ve yerel kuyruk yöneticisi aracısının aynı iřlem haline geldiğini belirtir. Aracı sürecinin kuyruk yöneticisine eriřmek için artık bir arabirimi kullanması gerekmedięi için, bu uygulamalar kuyruk yöneticisinin bir uzantısı haline gelir. Bu, MQCONNX çağırısında MQCNO\_FASTPATH\_BINDING seęeneęi tarafından tanımlanır.

Güvenilir uygulamaları iř parçacıklı IBM MQ kitaplıklarına baęlamaya gerek duyarsınız. IBM MQ uygulamasının güvenilir olarak çalıřacak řekilde ayarlamaya iliřkin yönergeler için bkz. [MQCNO Seęenekleri](#).

Bu seęenek en yüksek başarımlı saęlar.

**Not: Bu seęenek, kuyruk yöneticisinin bütünlüğünü saęlar: depolamanın üzerine yazıldığında koruma yoktur. Bu, uygulama iletileri ve kuyruk yöneticisinde bulunan dięer verileri de gösterebilecek hatalar içeriyorsa geçerlidir. Bu seęeneęi kullanmadan önce bu sorunları göz önünde bulundurun.**

## MQCNO\_SHARED\_BINDING

Uygulamayı ve yerel kuyruk yöneticisi aracısını ayrı süreçlerde çalıřtırabilmek için bu seęeneęi belirleyin. Bu, kuyruk yöneticisinin bütünlüğünü korur; yani, kuyruk yöneticisini errant programlarından korur. Ancak, uygulama ve yerel kuyruk yöneticisi araçları bazı kaynakları paylařır.

Bu seęenek, MQCNO\_FASTPATH\_BINDING ve MQCNO\_ISOLATED\_BINDING arasında, hem kuyruk yöneticisinin bütünlüğünü korumak aęısından, hem de MQI çağrılarının başarımlı aęısından ara ara saęlar.

Kuyruk yöneticisi bu baę tanımlı tipini desteklemiyorsa, MQCNO\_SHARED\_BINDING yoksayılr. Bu seęenek belirlenmemiř gibi iřleme devam eder.



Bir uygulama, MQCNO\_SHARED\_BINDING MQCNO\_SHARED\_BINDING komutunu kullanarak yerel kuyruk yöneticisine bağlıysa, uygulama çalışırken kuyruk yöneticisi durdurulabilir. Uygulama çalışmaya devam ederken kuyruk yöneticisini yeniden başlatırken, kuyruk yöneticisi tarafından gerekli olan kaynaklar üzerinde çalışmaya devam ettikçe, kuyruk yöneticisini başlatma girişimi AMQ7018 hatasıyla başarısız olur.

Kuyruk yöneticisini başlatmak için uygulamayı durdurmanız gerekir.

### MQCNO\_ISOLATED\_BINDING

Uygulamayı ve yerel kuyruk yöneticisi aracısını MQCNO\_SHARED\_BINDING MQCNO\_SHARED\_BINDING için ayrı işlemlerde çalıştırırken yapmak için bu seçeneği belirleyin. Ancak bu durumda, uygulama işlemi ve yerel kuyruk yöneticisi aracısı, kaynakları paylaşmadıkları için birbirlerinden yalıtılır.

Bu, kuyruk yöneticisinin bütünlüğünü korumak için en güvenli seçenektir, ancak bu, MQI çağrılarının en yavaş başarımını sağlar.

Kuyruk yöneticisi bu bağ tanımlama tipini desteklemiyorsa, MQCNO\_ISOLATED\_BINDING yoksayılır. Bu seçenek belirlenmemiş gibi işleme devam eder.

### MQCNO\_CLIENT\_BINDING

Uygulamanın yalnızca istemci bağlantısı denemesini sağlamak için bu seçeneği belirleyin. Bu seçenek aşağıdaki sınırlamalara sahiptir:

- ▶ **z/OS** MQCNO\_CLIENT\_BINDING, z/OS üzerinde yoksayıldı.
- MQCNO\_CLIENT\_BINDING, MQCNO\_STANDARD\_BINDING dışında herhangi bir MQCNO bağ tanımlama seçeneğiyle belirtildiyse, MQRC\_OPTIONS\_ERROR ile reddedildi.
- MQCNO\_CLIENT\_BINDING, bağ tanımlama tipini seçmek için kendi yöntemlerine sahip olduğu için Java için kullanılamaz.
- MQCONN çağrıldığında MQ\_CONNECT\_TYPE ortam değişkeni ayarlanmazsa, varsayılan bağ tanımlama tipini kullanan bir standart sunucu bağlantısı girişiminde bulunulması denir. Sunucu kitaplığı yüklenemezse, bir istemci bağlantısı girişiminde bulunmaya çalışılır.

### MQCNO\_LOCAL\_BINDING

Uygulamanın bir sunucu bağlantısını denemesini sağlamak için bu seçeneği belirleyin. MQCNO\_FASTPATH\_BINDING, MQCNO\_ISOLATED\_BINDING ya da MQCNO\_SHARED\_BINDING değeri de belirtilirse, bağlantı bu tipte olur ve bu bölümde belgelenir. Ters durumda, varsayılan bağ tanımlama tipi kullanılarak standart bir sunucu bağlantısı girişiminde bulunmaya çalışılır. MQCNO\_LOCAL\_BINDING, aşağıdaki sınırlamalara sahiptir:

- ▶ **z/OS** MQCNO\_LOCAL\_BINDING, z/OS üzerinde yoksayıldı.
- MQCNO\_RECONNECT\_AS\_DEF dışında herhangi bir MQCNO yeniden bağlanma seçeneğiyle belirtilirse, MQCNO\_LOCAL\_BINDING, MQRC\_OPTIONS\_ERROR ile reddedilir.
- MQCNO\_LOCAL\_BINDING, bağ tanımlama tipini seçmek için kendi mekanizmalarına sahip olduğu için, Java için kullanılamaz.
- MQCONN çağrıldığında MQ\_CONNECT\_TYPE ortam değişkeni ayarlanmazsa, varsayılan bağ tanımlama tipini kullanan bir standart sunucu bağlantısı girişiminde bulunulması denir. Sunucu kitaplığı yüklenemezse, bir istemci bağlantısı girişiminde bulunmaya çalışılır.

▶ **z/OS** z/OS ' ta bu seçenekler tolere edilir, ancak yalnızca standart bir bağ tanımlı bağlantı gerçekleştirilir.

▶ **z/OS** MQCNO Version 3 for z/OS allows four different options:

### MQCNO\_SERIALIZE\_CONN\_TAG\_QSG

Bu, bir uygulamanın, bir uygulamanın yalnızca bir eşgörünümünün, bir kuyruk paylaşım grubunda herhangi bir zamanda çalıştırılmasını istemesine olanak sağlar. Bu, uygulama tarafından belirtilen ya

da türetilen bir değere sahip bir bağlantı etiketinin kullanımını kayda geçirilerek elde edilir. Etiket, Sürüm 3 MQCNO ' da belirtilen 128 baytlık karakter dizilimidir.

### MQCNO\_RESTRICT\_CONN\_TAG\_QSG

Bu, her biri bir kuyruk yöneticisine bağlanabilen birden çok süreçten (ya da bir TCB) oluşan bir uygulamanın bulunduğu bir yerde kullanılır. Yalnızca, etiketin geçerli bir kullanımı yoksa ya da istekte bulunan uygulama aynı işlem kapsamı içindeyse bağlantıya izin verilir. Bu, etiket sahibiyle aynı kuyruk paylaşım grubunda bulunan MVS adres alanıdır.

### MQCNO\_SERIALIZE\_CONN\_TAG\_Q\_MGR







Bu, MQCNO\_SERIALIZE\_CONN\_TAG\_QSG komutuna benzer, ancak istenen etiketin zaten kullanımda olup olmadığını görmek için yalnızca yerel kuyruk yöneticisi sorgulanır.

### MQCNO\_RESTRICT\_CONN\_TAG\_Q\_MGR

Bu, MQCNO\_RESTRICT\_CONN\_TAG\_QSG komutuna benzer; ancak, istenen etiketin kullanımda olup olmadığını görmek için yalnızca yerel kuyruk yöneticisi sorgulanır.

### Güvenilir uygulamalar için kısıtlamalar

Güvenilir uygulamalar için aşağıdaki kısıtlamalar geçerlidir:

- Güvenilir uygulamaların kuyruk yöneticisiyle bağlantısını belirttik 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.
-   UNIX 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'ta:
  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 bir yaklaşım, MQ programları çalışırken işin yürürlükteki kullanıcılarını geçici olarak QMQM olarak değiştirmek için IBM i tanıtım değiş tokuş API işlevlerini, QSYGETPH, QWTSETP ve QSYRLSPH ' yi kullanmaktır. Bu işlevlerin ayrıntıları, kullanımlarına ilişkin bir örnekle birlikte, *IBM i System API Reference* adlı yayının Security API ' leri bölümünde sağlanır.
  2. Güvenilen uygulamaları System-Request Option 2 seçeneğini kullanarak ya da ENDJOB komutunu kullanarak çalıştırdıkları işleri sona erdirerek iptal etmeyin.
-    UNIX, Linux, and Windows sistemlerinde güvenilir 32 bit uygulamalar desteklenmez. Güvenilir bir 32 bit uygulamayı çalıştırmayı denerseniz, bu uygulama standart bağlı bir bağlantıya indirgenir.

### Shared (thread independent) connections with MQCONN

MQCONN ile paylaşılan bağlantılar ve dikkate alınacak bazı kullanım notları hakkında bilgi edinmek için bu bilgileri kullanın.

**Not:** IBM MQ for z/OS üzerinde desteklenmez.

IBM MQ for z/OS dışında IBM MQ altyapılarında, MQCONN ile yapılan bir bağlantı yalnızca bağlantıyı yapan iş parçacığıyla kullanılabilir. MQCONN çağrısına ilişkin seçenekler, bir işlemdeki tüm iş parçacıkları tarafından paylaşılabilen bir bağlantı yaratmanızı sağlar. Uygulamanız, aynı iş parçacığıda MQI çağrısını gerektiren bir işlemsel ortamda çalışıyorsa, aşağıdaki varsayılan seçeneği kullanmanız gerekir:

## **MQCNO\_HANDLE\_SHARE\_NONE**

Paylaşılmayan bir bağlantı oluşturur.

Diğer birçok ortamda, aşağıdaki iş parçacığı bağımsız, paylaşılan bağlantı seçeneklerinden birini kullanabilirsiniz:

## **MQCNO\_HANDLE\_SHARE\_BLOCK**

Paylaşılan bir bağlantı oluşturur. Bir MQCNO\_HANDLE\_SHARE\_BLOCK bağlantısında, bağlantı şu anda başka bir iş parçacığıdaki bir MQI çağrısı tarafından kullanıldıysa, MQI çağrısı, yürürlükteki MQI çağrısı tamamlanıncaya kadar bekler.

## **MQCNO\_HANDLE\_SHARE\_NO\_BLOCK**

Paylaşılan bir bağlantı oluşturur. On a MQCNO\_HANDLE\_SHARE\_NO\_BLOCK connection, if the connection is currently in use by an MQI call on another thread, the MQI call fails immediately with a reason of MQRC\_CALL\_IN\_PROGRESS.

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, süreçteki herhangi bir iş parçacığıdaki sonraki MQI çağrılarını tarafından kullanılabilir ve bu çağrılar MQCONNX' den döndürülen tanıtıcı ile ilişkilendirilir. Tek bir paylaşılan tanıtıcı kullanılarak yapılan MQI çağrılarını iş parçacıkları arasında diziselleştirilir.

Örneğin, paylaşılan bir tanıtıcı ile aşağıdaki etkinlik sırası mümkündür:

1. İş parçacığı 1 sorunları MQCONNX ve paylaşılan bir tanıtıcı alır *h1*
2. İş parçacığı 1, bir kuyruğu açar ve *h1* kullanarak bir alma isteği yayınlar
3. İş parçacığı 2, *h1* kullanarak bir put isteği yayınlar.
4. İş parçacığı 3, *h1* kullanarak bir put isteği yayınlar.
5. İş parçacığı 2, *h1* komutunu kullanarak MQDISC

Tanıtıcı herhangi bir iş parçacığının kullanımında olmakla birlikte, bağlantıya erişim diğer iş parçacıklarına kullanılamaz. In circumstances where it is acceptable that a thread waits for any previous call from another thread to complete, use MQCONNX with the option MQCNO\_HANDLE\_SHARE\_BLOCK.

Ancak engelleme, zorluklara neden olabilir. "2" sayfa 779 adımı, 1 numaralı iş parçacığıda henüz gelmemiş olabilecek iletiler için bekleyen bir alma isteği yayınlar (bekleme ile alma). Bu durumda, 2. ve 3 numaralı iş parçacıkları, iş parçacığının 1 numaralı iş parçacığının alma isteği kadar uzun süre bekletilmeye (engellendi) bırakılır. Tanıtıcıda başka bir MQI çağrısı zaten çalışıyorsa, bir MQI çağrısının bir hata ile döndürülmesini tercih ederseniz, MQCONNX seçeneğini MQCNO\_HANDLE\_SHARE\_NO\_BLOCK seçeneğiyle kullanın.

## **Paylaşılan bağlantı kullanım notları**

1. Bir nesne açılarak yaratılan herhangi bir nesne tanıtıcısı (Hobj) bir Hconn ile ilişkilendirilir; bu nedenle, paylaşılan bir Hconn için, Hobjs, Hconn kullanan herhangi bir iş parçacığı tarafından da paylaşılır ve kullanılabilir. Benzer şekilde, bir Hconn altında başlatılan herhangi bir iş birimi Hconn ile ilişkilendirilir; dolayısıyla, bu işlem, paylaşılan Hconn ile iş parçacıkları arasında paylaşılır.
2. *Herhangi biri* iş parçacığı, paylaşılan bir Hconn 'un bağlantısını kesmek için MQDISC ' yi çağırabilir; yalnızca ilgili MQCONNX 'i arayan iş parçacığından değil. MQDISC, tüm iş parçacıklarının kullanılmaması için Hconn 'ı sona erdirir.
3. Tek bir iş parçacığı birden çok paylaşılan Hconn özelliğini kullanabilir; örneğin, bir iletiyi paylaşılan bir Hconn altına koymak için MQPUT kullanarak, başka bir paylaşılan Hconn kullanarak başka bir ileti daha koyun; her işlem farklı bir yerel iş birimi altında olur.
4. Paylaşılan Hconns, genel bir iş birimi içinde kullanılamaz.

*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 yalnızca STANDARD bağ tanımları için herhangi bir etkiye sahiptir. Diğer bağ tanımları için MQ\_CONNECT\_TYPE yoksayıdır.

UNIX and Linux sistemlerinde IBM MQ for IBM i, IBM MQ for Windowsve IBM MQ sistemlerinde, MQCONNX çağrısında kullanılan MQCNO yapısının *Options* alanında belirtilen bağ tanımı tipiyle birlikte MQ\_CONNECT\_TYPE ortam değişkenini kullanabilirsiniz.

<i>Çizelge 117. MQCONNX çağrı seçeneklerinin MQ_CONNECT_TYPE ortam değişkeniyle nasıl kullanıldığı</i>		
<b>MQCONNX çağrı seçeneği</b>	<b>MQ_CONNECT_TYPE ortam değişkeni</b>	<b>Sonuç</b>
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.

### **MQDISC kullanan bir kuyruk yöneticisinden programların bağlantısı kesiliyor**

MQDISC kullanarak bir kuyruk yöneticisinden programların bağlantısını kesme hakkında bilgi edinmek için bu bilgileri kullanın.

MQCONN ya da MQCONNX çağrısını kullanan bir kuyruk yöneticisine bağlı bir program, kuyruk yöneticisiyle tüm etkileşimi bitirdiğinde, MQDISC çağrısını kullanarak bağlantıyı keser.

- CICS Transaction Server for z/OS uygulamaları üzerinde, MQCONNX kullanılmadığı ve uygulama sona erdirilmeden önce bağlantı etiketini atmak istiyorsanız, bu çağrıların isteğe bağlı olduğuyheryu uygulamaları.
- On IBM MQ for IBM i where, when you sign off from the operating system, an implicit MQDISC call is made.

MQDISC çağrısına giriş olarak, kuyruk yöneticisine bağlandığınızda MQCONN ya da MQCONNX tarafından döndürülen bağlantı tanıtıcısını (Hconn) sağlamanız gerekir.

Except on CICS on z/OS, after MQDISC is called the connection handle (Hconn) is no longer valid, and you cannot issue any further MQI calls until you call MQCONN or MQCONNX again. MQDISC, bu tanıtıcıyı kullanarak hala açık olan nesnelere için örtük bir MQCLOSE yapar.

**z/OS** z/OS' e bağlı bir istemci için, bir MQDISC çağrısı yayınlandığında örtük bir kesinleştirme gerçekleştirilir; ancak, kanal gerçekte sona erinceye kadar açık olan kuyruk tutamaçları kapatılmaz.

IBM MQ for z/OS üzerinde bağlanmak için MQCONNX kullanırsanız, MQDISC, MQCONNX tarafından kurulan bağlantı etiketinin kapsamını da sona erdirir. Ancak, bir CICS, IMS ya da RRS uygulamasında, bir bağlantı etiketiyle ilişkilendirilmiş etkin bir kurtarma birimi varsa, MQDISC, MQRC\_CONN\_TAG\_NOT\_SAILD bir neden koduyla reddedilir.

Değiştirgelere ilişkin açıklamalar, [MQDISC](#) içindeki MQDISC çağrısına ilişkin açıklamayla verilir.

### **MQDISC komutu verilmediği zaman**

Bir standart, paylaşılmayan bağlantı (Hconn), oluşturma iş parçacığı sona erdiğinde temizlenir. Paylaşılan bir bağlantı, tüm işlem sona erdiğinde örtük olarak geriletilir ve bağlantısı kesilir. Hconn hala varsa, paylaşılan Hconn 'ı yaratan iş parçacığı sonlandırılırsa, Hconn hala kullanılabilir durumda olur.

## Yetki denetimi

MQCLOSE ve MQDISC çağruları genellikle hiçbir yetki denetimi gerçekleştirmez.

Bir IBM MQ nesnesine açma ya da bağlantı kurma yetkisi olan bir işin olağan bir şekilde kapatılır ya da bu nesneden bağlantı kesilsin. Bir IBM MQ nesnesine bağlı ya da açılmış bir işin yetkisi iptal edilmiş olsa bile, MQCLOSE ve MQDISC çağruları kabul edilir.

## Nesnelerin açılması ve kapatılması

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

Aşağıdaki işlemlerden herhangi birini gerçekleştirmek için, ilgili IBM MQ nesnesini ilk kez *açmalısınız* :

- İletileri kuyruğa koy
- Kuyruktan ileti alma (gözetme ya da alma)
- Bir nesnenin özniteliklerini ayarlama
- Herhangi bir nesnenin özniteliklerine ilişkin olarak sorgulama

Nesneyi açmak için, çağrıya ilişkin seçenekleri kullanarak nesneyle ne yapmak istediğinizi belirtmek için MQOPEN çağrısını kullanın. Tek kural dışı durum, kuyruğa tek bir ileti koymak istiyorsanız, kuyruğu hemen kapatmanız gerekir. Bu durumda, MQPUT1 çağrısını kullanarak *açma* aşasını atlayabilirsiniz (bkz. [“MQPUT1 çağrısını kullanarak bir ileti kuyruğa konması” sayfa 799](#) ).

MQopen çağrısını kullanarak bir nesneyi açmadan önce, programınızı bir kuyruk yöneticisine bağlamanız gerekir. Bu, [“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)' ta tüm ortamlar için ayrıntılı olarak açıklanmıştır.

Açabileceğiniz dört tip IBM MQ nesnesi vardır:

- Kuyruk
- Ad Listesi
- Süreç tanımlaması
- Kuyruk yöneticisi

Tüm bu nesnelere, MQOPEN çağrısını kullanarak benzer şekilde açmanızı sağlar. IBM MQ nesnelere ilişkin ek bilgi için [Nesne tipleri](#) başlıklı konuya bakın.

Aynı nesneyi bir kereden fazla açabilirsiniz ve her defasında yeni bir nesne tanıtıcısı elde edebilirsiniz. Bir kuyruktaki iletilere bir tanıtıcı kullanarak göz atmak ve aynı kuyruktan iletileri başka bir tanıtıcı kullanarak kaldırmak isteyebilirsiniz. Bu işlem, aynı nesneyi kapatmak ve yeniden açmak için kaynakları kullanarak kurtarır. Ayrıca, aynı anda iletileri kaldırmak için ve ' e göz atmak için bir kuyruk da açabilirsiniz.

Ayrıca, tek bir MQXX\_ENCODE\_CASE\_ONE open ile birden çok nesne açabilir ve bu nesnelere MQCLOSE kullanarak kapatabilirsiniz. Bunun nasıl yapacağını ilişkin bilgi için bkz. [“Dağıtım listeleri” sayfa 801](#) .

Bir nesneyi açmaya çalıştığınızda, kuyruk yöneticisi, o nesneyi MQOPEN çağrısında belirlediğiniz seçenekler için açma yetkisine sahip olduğunuzu denetler.

Bir program kuyruk yöneticisinden bağlantıyı kestiğinde nesnelere otomatik olarak kapatılır. IMS ortamında, bir GU (benzersiz alma) IMS çağrısından sonra yeni bir kullanıcı için bir program işlenmeye başladığında bağlantı kesilmeye zorlanır. IBM i platformunda, bir iş sona erdiğinde nesnelere otomatik olarak kapatılır.

Açtığınız nesnelere kapatmak için iyi bir programlama uygulamasıdır. Bunu yapmak için MQCLOSE çağrısını kullanın.

Nesneleri açma ve kapatma hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- [“MQOPEN çağrısını kullanarak nesnelere açılması” sayfa 782](#)
- [“Dinamik kuyruklar oluşturma” sayfa 789](#)
- [“Uzak Kuyrukların Açılması” sayfa 790](#)
- [“MQCLOSE çağrısını kullanarak nesnelere kapatılıyor” sayfa 790](#)

## İlgili kavramlar

[“Message Queue Interface-Genel Bakış” sayfa 760](#)

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabılır alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OSüzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

[“IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

## ***MQOPEN çağrısını kullanarak nesnelerin açılması***

MQOPEN çağrısını kullanarak nesneleri açma hakkında bilgi edinmek için bu bilgileri kullanın.

MQOPER çağrısına giriş olarak, aşağıdaki bilgileri sağlamanız gerekir:

- Bağlantı tanıtıcısı. z/OSüzerindeki CICS uygulamaları için, değişmez MQHC\_DEF\_HCONN (sıfır değerine sahip) değerini belirtebilir ya da MQCONN ya da MQCONNX çağrısının döndürdüğü bağlantı tanıtıcısını kullanabilirsiniz. Diğer programlar için, her zaman MQCONN ya da MQCONNX çağrısının döndürdüğü bağlantı tanıtıcısını kullanın.
- Nesne tanımlayıcı yapısını (MQOD) kullanarak açmak istediğiniz nesneye ilişkin bir açıklama.
- Çağrıyı denetleyen bir ya da daha fazla seçenek.

MQXX\_ENCODE\_CASE\_ONE open komutunun çıkışı şöyledir:

- Nesneye erişiminizi gösteren bir nesne tanıtıcısı. Bunu izleyen herhangi bir MQI çağrısına giriş sırasında kullanın.
- Dinamik bir kuyruk yaratıyorsanız, değiştirilmiş bir nesne tanımlayıcı yapısı (ve platformunuzda destekleniyorsa).
- Bir tamamlanma kodu.
- Bir neden kodu.

## **Bir nesne tutamacının kapsamı**

Bir nesne tanıtıcısı (Hobj) kapsamı, bağlantı tanıtıcısı (Hconn) kapsamı ile aynıdır.

Bu, [“MQCONN ya da MQCONNX kapsamı” sayfa 775](#) ve [“Shared \(thread independent\) connections with MQCONNX” sayfa 778](#) ile kaplıdır. Ancak, bazı ortamlarda dikkate alınması gereken noktalar vardır:

## CICS

Bir CICS programında, tanıtıcıyı yalnızca, MQOPEN çağrısını yaptığınız aynı CICS görevi içinde kullanabilirsiniz.

## IMS ve z/OS toplu işi

IMS ve toplu iş ortamlarında, bu tanıtıcıyı aynı görev içinde kullanabilirsiniz, ancak herhangi bir alt görev içinde kullanamazsınız.

Descriptions of the parameters of the MQOPEN call are given in [MQOPEN](#).

Aşağıdaki kısımlarda, MQOPEN ' a 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 parametresidir. (Devingen kuyruk yaratmak için MQOPEN çağrısını kullandığınızda, yapı kuyruk yöneticisiyle değiştirilir.)

MQOD yapısıyla ilgili tüm ayrıntılar için [MQOD](#) başlıklı konuya bakın.

For information about using the MQOD structure for distribution lists, see “MQOD yapısının kullanılması” sayfa 802 under “Dağıtım listeleri” sayfa 801.

### Ad çözünürlüğü

MQOPER çağrısı, kuyruğu ve kuyruk yöneticisi adlarını nasıl çözer.

**Not:** Kuyruk yöneticisi diğer adı, RNAME alanı olmayan bir uzak kuyruk tanımlamasıdır.

Bir IBM MQ kuyruğu açtığınızda, MQOPER çağrısı belirttiğiniz kuyruk adında bir ad çözme işlevi gerçekleştirir. Bu, kuyruk yöneticisinin sonraki işlemleri hangi kuyruğa gerçekleştireceğini belirler. Diğer bir deyişle, bir diğer ad kuyruğunun adını ya da nesne tanımlayıcınızda (MQOD) uzak bir kuyruk belirlediğinizde, çağrı adı yerel bir kuyrukta ya da bir iletim kuyruğuna çözer. Herhangi bir giriş, göz atma ya da küme için bir kuyruk açılırsa, bu bir kuyruk varsa, yerel bir kuyruğa çözülür ve bir hata varsa, bu bir yerel kuyruğa çözülür. Yalnızca çıkış için açılmışsa, yalnızca çıkış için açılmışsa, yalnızca yalnızca sorguların ya da çıkışın ve sorgu için açılırsa yerel olmayan bir kuyruğa çözülür. Ad çözme işlemine ilişkin genel bilgiler için bkz. [Çizelge 118 sayfa 783](#) . *ObjectQMgrName* içinde sağladığınız ad, *ObjectName* ' ta önce çözümlenir.

[Çizelge 118 sayfa 783](#) ayrıca, kuyruk yöneticisi adı için bir diğer ad tanımlamak üzere uzak bir kuyruğun yerel tanımlamasını nasıl kullanabileceğinin de gösterilir. Bu, iletileri uzak bir kuyruğa koyduğunuzda hangi iletim kuyruğunun kullanılacağını seçmenizi sağlar. Böylece, uzak kuyruk yöneticilerine gönderilen iletiler için tek bir iletim kuyruğu kullanabilirsiniz.

Aşağıdaki çizelgeyi kullanmak için, önce soldaki iki kolonu ( **MQOD ' ye giriş** başlığı altında) okuyun ve uygun vakayı seçin. Ardından, yönergeleri izleyerek karşılık gelen satırı okuyun. **Çözümlen adlar** kolonlarındaki yönergeleri izleyerek, **MQOD ' ye giriş** kolonlarına geri dönebilir ve belirtildiği gibi değerleri ekleyebilirsiniz; tersi durumda, belirtilen sonuçlarla çizelgeden çıkabilirsiniz. Örneğin, *ObjectName* girişini yapmak zorunda kalabilirsiniz.

<b>MQOD ' ye giriş</b>	<b>MQOD ' ye giriş</b>	<b>Çözümlenen adlar</b>	<b>Çözümlenen adlar</b>	<b>Çözümlenen adlar</b>
<i>ObjectQMgrName</i>	<i>ObjectName</i>	<i>ObjectQMgrName</i>	<i>ObjectName</i>	Transmission queue
Boş ya da yerel kuyruk yöneticisi	CLUSTER özniteliği olmayan yerel kuyruk	Yerel kuyruk yöneticisi	Giriş <i>ObjectName</i>	Geçerli değil (yerel kuyruk kullanıldı)

Çizelge 118. MÇOPEN kullanılırken kuyruk adlarının çözülmesi (devamı var)

MÇOD ' ye giriş	MÇOD ' ye giriş	Çözömlenen adlar	Çözömlenen adlar	Çözömlenen adlar
Boş kuyruk yöneticisi	CLUSTER özniteliđi olan yerel kuyruk	İş yükü yönetimi seçilen küme kuyruk yöneticisi ya da PUT üzerinde seçilen belirli bir küme kuyruk yöneticisi	Giriş <i>ObjectName</i>	SYSTEM.CLUSTER.TRANSMIT.QUEUE ve yerel kuyruk kullanıldı SYSTEM.ÇSG.TRANSMIT.QUEUE (bkz. not)
Yerel kuyruk yöneticisi	CLUSTER özniteliđi olan yerel kuyruk	Yerel kuyruk yöneticisi	Giriş <i>ObjectName</i>	Geçerli deđil (yerel kuyruk kullanıldı)
Boş ya da yerel kuyruk yöneticisi	Model kuyruđu	Yerel kuyruk yöneticisi	Oluşturulan ad	Geçerli deđil (yerel kuyruk kullanıldı)
Boş ya da yerel kuyruk yöneticisi	CLUSTER özniteliđi olan ya da olmayan diđer ad kuyruđu	Perform name resolution again with <i>ObjectQMgrAd</i> unchanged, and input <i>ObjectName</i> set to the <i>BaseQName</i> in the alias queue definition object.  <i>ObjectQMgrAd</i> belirtildiđinde yerel olarak tanımlanmış bir diđer ada çözümlenmemelidir, ancak <i>ObjectQMgrName</i> ' in boş olduđu, kümelenmiş diđer ada (diđer kuyruk yöneticilerine ev sahipliđi yaptı) çözümlenmelidir.		
Yerel kuyruk yöneticisi	CLUSTER özniteliđi olan diđer ad kuyruđu	Diđer ad, yerel olarak tanımlanmış olmayan bir küme kuyruđuna ya da diđer ad ile aynı <i>ObjectName</i> olan bir küme kuyruđuna çözümlenmemelidir.		
Boş kuyruk yöneticisi	CLUSTER özniteliđi olan diđer ad kuyruđu	Diđer ad, diđer ad olarak aynı <i>ObjectName</i> olan bir küme kuyruđuna çözümlenmelidir.		



Çizelge 118. MÇOPEN kullanılırken kuyruk adlarının çözülmesi (devamı var)

MÇOD ' ye giriş	MÇOD ' ye giriş	Çözömlenen adlar	Çözömlenen adlar	Çözömlenen adlar
Boş ya da yerel kuyruk yöneticisi	Uzak kuyruğun yerel tanımlaması	<i>ObjectQMGrAd</i> kümesi <i>RemoteQMGrAd</i> olarak ve <i>ObjectName</i> <i>RemoteQName</i> olarak ayarlanmış şekilde ad çözömlemesini yeniden gerçekleştirin. Uzak kuyruklar çözölmemelidir		<i>XmitQName</i> özniteliğinin adı, blank; değilse, uzak kuyruk tanımlaması nesnesindeki <i>RemoteQMGrName</i> ( <i>RemoteQMGr</i> ) adı. SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)
Boş kuyruk yöneticisi	Eşleşen yerel nesne yok; küme kuyruğu bulundu	İş yükü yönetimi seçilen küme kuyruk yöneticisi ya da PUT üzerinde seçilen belirli bir küme kuyruk yöneticisi	Giriş <i>ObjectName</i>	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	Burada geçerli değil
Kuyruk yöneticisinin yerel kuyruk yöneticisi olarak aynı kuyruk paylaşım grubunda yer alan adı	Yerel paylaşılan kuyruk	Yerel kuyruk yöneticisi	Giriş <i>ObjectName</i>	Burada geçerli değil
Yerel iletim kuyruğunun adı	(Çözömlenmedi)	Giriş <i>ObjectQMGrAd</i>	Giriş <i>ObjectName</i>	Giriş <i>ObjectQMGrAd</i> SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)
Kuyruk yöneticisi diğer adı tanımlaması (yerel kuyruk yöneticisi <i>RemoteQMGrAd</i> olabilir)	(Çözömlenmedi, uzak kuyruk)	Perform name resolution again with <i>ObjectQMGrAd</i> set to <i>RemoteQMGrAd</i> . Uzak kuyruklara çözömlenmemelidir	Giriş <i>ObjectName</i>	<i>XmitQName</i> özniteliğinin adı, blank; değilse, uzak kuyruk tanımlaması nesnesindeki <i>RemoteQMGrName</i> ( <i>RemoteQMGr</i> ) adı. SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)
Kuyruk yöneticisi herhangi bir yerel nesnenin adı değil; küme kuyruğu yöneticileri ya da kuyruk yöneticisi diğer adı bulundu	(Çözömlenmedi)	<i>ObjectQMGrAd</i> or specific cluster queue manager selected on PUT	Giriş <i>ObjectName</i>	SYSTEM.CLUSTER.TRANSMIT.QUEUE SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)

Çizelge 118. MÇOPEN kullanılırken kuyruk adlarının çözülmesi (devamı var)				
MÇOD ' ye giriş	MÇOD ' ye giriş	Çözömlenen adlar	Çözömlenen adlar	Çözömlenen adlar
Kuyruk yöneticisi herhangi bir yerel nesnenin adı değil; herhangi bir küme nesnesi bulunamadı	(Çözömlenmedi)	Giriş <i>ObjectQMGrAdı</i>	Giriş <i>ObjectName</i>	<i>DefXmitQName</i> attribute of the queue manager where <i>DefXmitQName</i> is supported.  SYSTEM.QSG.TRANSMIT.QUEUE (bkz. not)

#### Notlar:

1. *BaseQName* , diğer ad kuyruğunun tanımlamasından temel kuyruğun adıdır.
2. *RemoteQName* , uzak kuyruğun yerel tanımlamasından uzak kuyruğun adıdır.
3. *RemoteQMGrName* , uzak kuyruğun yerel tanımlamasından uzak kuyruk yöneticisinin adıdır.
4. *XmitQName* , uzak kuyruğun yerel tanımlamasından iletim kuyruğunun adıdır.
5. Kuyruk paylaşım grubunun (QSG) bir parçası olan IBM MQ for z/OS kuyruk yöneticilerini kullanırken, [Çizelge 118 sayfa 783](#) içindeki yerel kuyruk yöneticisi adı yerine kuyruk paylaşım grubunun adı kullanılabilir.  
  
Yerel kuyruk yöneticisi hedef kuyruğu açamıyorsa ya da kuyruğa bir ileti koyarsa, ileti belirtilen *ObjectQMGrAdına*, grup içi kuyruğa alma ya da bir IBM MQ kanalına aktarılır.
6. Çizelgenin *ObjectName* kolonunda CLUSTER, kuyruğun hem CLUSTER, hem de CLUSNL özniteliklerine gönderme yapar.
7. SYSTEM.QSG.TRANSMIT.QUEUE değeri, yerel ve uzak kuyruk yöneticileri aynı kuyruk paylaşım grubunda yer alıyorsa, grup içi kuyruklama özelliği geçerli kılındığında kullanılır.
8. Her bir küme gönderici 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 how to configure cluster transmissing queues](#) başlıklı konuya bakın.
9. Kuyruk yöneticisinin herhangi bir yerel nesnenin adı olmadığı durumlarda; küme kuyruğu yöneticileri ya da bulunan kuyruk yöneticisi diğer adı bulunur.

**ObjectQMGrName** komutunu kullanarak bir kuyruk yöneticisi adı sağladığınızda ve yerel kuyruk yöneticisi tarafından bu hedefe ulaşabilecek farklı küme adları olan birden çok küme kanalı varsa, hedef kuyruğun küme adıyla bağımsız olarak, iletiyi taşımak için bu kanallardan herhangi biri kullanılabilir.

Yalnızca kuyruğun aynı küme adına sahip bir kanaldan gönderilmek üzere o kuyruğa ilişkin iletileri bekliyorsanız, bu beklenmeyen bir durum olabilir.

Ancak, **ObjectQMGrName** bu durumda öncelikli olur ve küme iş yükü dengelemesi, içinde oldukları küme adı ne olursa olsun, o kuyruk yöneticisine ulaşabilecek tüm kanalları dikkate alır.

Bir diğer ad kuyruğunun açılması, diğer adın çözdüğü temel kuyruğu da açar ve uzak bir kuyruk açılırsa iletim kuyruğunu da açar. Bu nedenle, belirttiğiniz kuyruğu ya da diğerinin açık olduğu kuyrukları silemez.

Bir diğer ad kuyruğu yerel olarak tanımlanmış başka bir diğer ad kuyruğuna (bir kümede paylaşılan ya da değil) çözümlenemediğinde, uzaktan tanımlanan bir diğer ad kuyruğunun çözülmesi izin verilir ve bu nedenle temel kuyruk olarak belirlenebilir.

Çözömlenen kuyruk adı ve çözülen kuyruk yöneticisi adı, MÇOD ' daki *ResolvedQName* ve *ResolvedQMGrName* alanlarında saklanır.

Dağıtılmış bir kuyruğa alma ortamındaki ad çözümlemesi hakkında daha fazla bilgi için [Kuyruk adı çözümlemesi nedir?](#) başlıklı konuya bakın.

### *MQOPER çağrısının seçeneklerinin kullanılması*

MQOPER çağrısının **Options** değiştirilmesinde, açtığınız nesneye verdiğiniz erişimi denetlemek için bir ya da daha çok seçenek seçmeniz gerekir. Bu seçeneklerle şunları yapabilirsiniz:

- Bir kuyruğu açın ve o kuyruğa koyulan tüm iletilerin, aynı yönetim ortamına yönlendirilmesi gerektiğini belirtin
- İleti koymanıza izin vermek için bir kuyruk açın.
- İletilere göz atmanızı sağlamak için bir kuyruk açın
- Kuyruktan ileti kaldırmanıza izin vermek için bir kuyruk açın
- Özniteliklerini sorgulamanıza ve ayarlamasına izin vermek için bir nesneyi açın (ancak yalnızca kuyrukların özniteliklerini ayarlayabilirsiniz)
- İletileri yayınlamak için bir konuyu ya da konu dizesini açın
- Bağlam bilgilerini bir iletiyle ilişkilendirin
- Güvenlik denetimleri için kullanılacak alternatif bir kullanıcı kimliği gösterir.
- Kuyruk yöneticisi susturulmuş durumdaysa, çağrıyı denetleyebilirsiniz.

### *Küme kuyruğu için MQOPEN seçeneği*

Kuyruk tanıtıcısı için kullanılan bağ tanımı, MQBND\_BIND\_ON\_OPEN, MQBND\_BIND\_NOT\_FIXED ya da MQBND\_BIND\_ON\_GROUP değerini alabilen **DefBind** kuyruk özneliğinden 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\_DEF varsayılan 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 919](#).

Bir küme kuyruğunu QALIAS tanımı kullanarak açarsanız, bazı kuyruk öznelikleri temel kuyruk değil, diğer ad kuyruğu tarafından tanımlanır. Küme öznelikleri, diğer ad kuyruğu tarafından geçersiz kılınan temel kuyruk tanımlamasının öznelikleri arasındadır. Örneğin, aşağıdaki parçacıkta, küme kuyruğu MQOO\_BIND\_ON\_OPEN ile 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 kuyruk ya da konu açmak için, MQOO\_OUTPUT seçeneğini kullanın.

### *İletilere göz atmak için MQOPEN seçeneği*

Bir kuyruğu açmak için, ilgili iletileri göz atabilmeniz için, MQOO\_BROWSE seçeneğiyle MQOPEN çağrısını kullanın.

Bu, kuyruk yöneticisinin kuyruğun sonraki iletisini tanımlamak için kullandığı bir göz atma imleçini yaratır. Daha fazla bilgi için [“Kuyruklardaki İletilere Göz Atma” sayfa 839](#) başlıklı konuya bakın.

### **Not:**

1. Uzak kuyruklardaki iletilere göz atamazsınız; MQOO\_BROWSE seçeneğini kullanarak uzak kuyruk açmayın.
2. Bir dağıtım listesi açılırken bu seçeneği belirleyemezsiniz. Dağıtım listeleriyle ilgili daha fazla bilgi için bkz. “Dağıtım listeleri” sayfa 801.
3. İşbirliğine göz atma kullanıyorsanız, MQOO\_CO\_OP ' ı MQOO\_BROWSE ile birlikte kullanın; bkz. [Seçenekler](#)

#### *İletileri kaldırmak için MQOPEN seçenekleri*

Üç seçenek, iletileri bu kuyruktan kaldırmak için bir kuyruğun açılmasını denetler.

Bu çizelgelerden yalnızca birini herhangi bir MQOPER çağrısında kullanabilirsiniz. Bu seçenekler, programınızın kuyruğa özel ya da paylaşılan erişim olup olmadığını tanımlar. *Dışlayıcı erişim* , kuyruğu kapatıncaya kadar, yalnızca siz iletileri bu iletiden kaldırabilirsiniz. İletileri kaldırmak için başka bir program kuyruğu açma girişiminde bulunursa, MQOPEN çağrısı başarısız olur. *Paylaşılan erişim* , birden çok programın kaldırılabilirdiği anlamına gelir kuyruktan iletilere bakın.

En çok önerilebilen yaklaşım, kuyruk tanımlandığında kuyruk için tasarlanan erişim tipini kabul etmesidir. Kuyruk tanımı, **Shareability** ve/ **DefInputOpenOption** öznitelikleri. Bu erişimi kabul etmek için, MQOO\_INPUT\_AS\_Q\_DEF seçeneğini kullanın. Bu özniteliklerin ayarının, bu seçeneği kullanırken size verilecek erişim tipini nasıl etkilediğini görmek için [Çizelge 119 sayfa 788](#) dosyasına bakın.

Kuyruk öznitelikleri		MQREOL seçeneklerine ilişkin erişim tipi		
Shareability	DefInputOpenOption	AS_Q_DEF	Paylaşılan	dışlayıcı
PAYLAŞIM	Paylaşılan	paylaşılan	paylaşılan	dışlayıcı
PAYLAŞIM	dışlayıcı	dışlayıcı	paylaşılan	dışlayıcı
NOT_SHAREABLE*	SHARED*	dışlayıcı	dışlayıcı	dışlayıcı
NOT_SHAREABLE	dışlayıcı	dışlayıcı	dışlayıcı	dışlayıcı

**Not:** \* Bu özniteliklerin birleşimine sahip olmak için bir kuyruk tanımlayabilmenize rağmen, varsayılan giriş açma seçeneği, paylaşılabilirlik öznetemesiyle geçersiz kılınır.

Alternatif olarak:

- Başka programlar kuyruktan aynı anda iletileri kaldırabilse bile uygulamanızın başarılı bir şekilde çalışabileceğini biliyorsanız, MQOO\_INPUT\_SHARED seçeneğini kullanın. [Çizelge 119 sayfa 788](#) , bu seçenekle birlikte, bazı durumlarda kuyruğa dışlayıcı erişim verileceğini gösterir.
- Uygulamanızın, iletileri aynı anda kuyruktan kaldırmak için başka programlar önleniyorsa başarılı bir şekilde çalışabileceğini biliyorsanız, MQOO\_INPUT\_EXCLUSIVE seçeneğini kullanın.

#### **Not:**

1. İletileri uzak bir kuyruktan kaldıramazsınız. Bu nedenle, MQOO\_INPUT\_ \* seçeneklerinden herhangi birini kullanarak uzak bir kuyruk açamazsınız.
2. Bir dağıtım listesi açılırken bu seçeneği belirleyemezsiniz. Daha fazla bilgi için bkz. “Dağıtım listeleri” sayfa 801.

#### *Özniteliklerin ayarlanması ve araştırılmasına ilişkin MQOPEN seçenekleri*

Bir kuyruğu açmak için, bu kuyruğun özniteliklerini ayarlayabilmeniz için MQOO\_SET seçeneğini kullanın.

Başka bir nesne türünün özniteliklerini ayarlayamazsınız (bkz. “Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885 ).

Özniteliklerini sorgulayabilmeniz için bir nesneyi açmak için MQOO\_INCOVER seçeneğini kullanın.

**Not:** Bir dağıtım listesi açılırken bu seçeneği belirleyemezsiniz.

### *İleti bağlamına ilişkin MQAÇ seçenekleri*

Bir iletiyi bir kuyruğa koyduğunuzda bağlam bilgilerini bir iletiyle ilişkilendirebilmek istiyorsanız, kuyruğu açtığınızda ileti bağlamı seçeneklerinden birini kullanmanız gerekir.

Seçenekler, iletiyi oluşturan *kullanıcıyla* ilişkili bağlam bilgilerini ve iletiyi oluşturan *uygulama* ile ilgili olan bağlam bilgilerini birbirinden ayırt etmenizi sağlar. Ayrıca, iletiyi kuyruğa koyduğunuzda bağlam bilgilerini ayarlamayı tercih edebilir ya da bağlamın başka bir kuyruk saptından otomatik olarak alınmasını tercih edebilirsiniz.

### **İlgili kavramlar**

“İleti bağlamı” sayfa 44

*İleti bağlamı* bilgileri, iletiyi alan uygulamanın, iletiyi oluşturan iletiyi bulmasını sağlayan bir uygulamaya olanak tanır.

“İleti bağlamı bilgilerinin denetlenmesi” sayfa 797

Bir iletiyi kuyruğa koymak için MQPUT ya da MQPUT1 çağrısını kullandığınızda, kuyruk yöneticisinin ileti tanımlayıcısına varsayılan bağlam bilgileri eklemesini belirleyebilirsiniz. Uygun yetki düzeyine sahip uygulamalar ek bağlam bilgileri ekleyebilir. Bağlam bilgilerini denetlemek için, MQPMO yapısındaki seçenekler alanını kullanabilirsiniz.

*Diğer kullanıcı yetkisi için MQOPEN seçeneği*

MQOPER çağrısını kullanarak bir nesneyi açmaya çalıştığınızda, kuyruk yöneticisi o nesneyi açma yetkisine sahip olup olmadığını denetler. Yetkiniz yoksa, arama başarısız olur.

Ancak, sunucu programları, kuyruk yöneticisinin, sunucunun kendi yetkisi yerine, üzerinde çalışmakta oldukları kullanıcının yetkilendirmesini denetmesini isteyebilirler. Bunu yapmak için, bu kullanıcılar, MQOPED çağrısının MQOO\_ALTERNATE\_USER\_AUTHORITY seçeneğini kullanmalı ve MQOD yapısının *AlternateUserId* alanında diğer kullanıcı kimliğini belirtmelidir. Genellikle, sunucu, işlem yaptığı iletteki bağlam bilgilerinden kullanıcı kimliğini alır.

**z/OS** *Kuyruk yöneticisi için MQOPEN seçeneği susturulmuş*

Kuyruk yöneticisi susturma durumundaysa, MQOPER çağrısını kullanırsanız, kullandığınız ortama bağlı olarak arama başarısız olabilir.

z/OS üzerindeki CICS ortamında, kuyruk yöneticisi susturma durumundaysa, MQOPER çağrısını kullanırsanız, çağrı her zaman başarısız olur.

Diğer z/OS ortamlarında, IBM i, Windows sistemlerinde ve UNIX and Linux sistem ortamlarında, kuyruk yöneticisi yalnızca MQOPEN çağrısının MQOO\_FAIL\_IF\_QUIESCING seçeneğini kullanıyorsanız, susturma işlemi başarısız olur.

*Yerel kuyruk adlarını çözmek için MQOPEN seçeneği*

Yerel, diğer ad ya da model kuyruğunu açtığınızda, yerel kuyruk döndürülür.

Ancak, uzak bir kuyruğu ya da küme kuyruğunu açtığınızda, MQOD yapısının *ResolvedQName* ve *ResolvedQMgrName* alanları, uzak kuyruk tanımında bulunan uzak kuyruk ya da uzak kuyruk yöneticisi adlarıyla ya da seçilen uzak küme kuyruğunda doldurulur.

MQOP çağrısının MQOO\_RESOLVE\_LOCAL\_Q seçeneğini kullanarak, açıldığı yerel kuyruğun adıyla MQOD yapısındaki *ResolvedQName* ögesini doldurmanız gerekir. *ResolvedQMgrName* , aynı şekilde yerel kuyruğu barındıran yerel kuyruk yöneticisinin adıyla doldurulur. Bu alan yalnızca MQOD yapısındaki Sürüm 3 ile kullanılabilir; yapı Sürüm 3 'ten küçükse, bir hata döndürülmeden MQOO\_RESOLVE\_LOCAL\_Q yoksayıdır.

If you specify MQOO\_RESOLVE\_LOCAL\_Q when opening, for example, a remote queue, *ResolvedQName* is the name of the transmission queue to which messages will be put. *ResolvedQMgrName* , iletim kuyruğunu bulunduran yerel kuyruk yöneticisinin adıdır.

### **Dinamik kuyruklar oluşturma**

Uygulamanızın sona ermesinden sonra kuyruğa gerek kalmadığında dinamik bir kuyruk kullanın.

Örneğin, yanıtla kuyruğunuz için dinamik bir kuyruk kullanabilirsiniz. Bir kuyruğa ileti koyduğunuzda, MQMD yapısının *ReplyToQ* alanında yanıtlanacak yanıtın adını belirtiyorsunuz (bkz. "MQMD yapısıyla iletilerin tanımlanması" sayfa 793 ).

Dinamik bir kuyruk yaratmak için, MQOPEN çağrısıyla birlikte, model kuyruğu olarak bilinen bir şablonu kullanıyorsunuz. You create a model queue using the IBM MQ commands or the operations and control panels. Yarattığınız dinamik kuyruk, model kuyruğun özniteliklerini alır.

MQOL ' u çağırdığınızda, MQOD yapısının *ObjectName* alanında model kuyruğunun adını belirtin. Arama tamamlandığında, *ObjectName* alanı oluşturulan dinamik kuyruğun adına ayarlanır. Ayrıca, *ObjectQMGrName* alanı yerel kuyruk yöneticisinin adına ayarlıdır.

Yarattığınız dinamik kuyruğun adını üç yolla belirtebilirsiniz:

- MQOD yapısındaki *DynamicQName* alanında istediğiniz tam adı verin.
- Ad için bir önek (en az 33 karakter) belirtin ve kuyruk yöneticisinin adın geri kalanını oluşturmasını sağlayın. Bu, kuyruk yöneticisinin benzersiz bir ad oluşturduğu, ancak yine de bazı denetime sahip olduğunuz anlamına gelir (örneğin, her kullanıcının belirli bir öneki kullanmasını ya da adlarında belirli bir öneke sahip kuyruklar için özel bir güvenlik sınıflandırması vermek isteyebilirsiniz). Bu yöntemi kullanmak için, *DynamicQName* alanının en son olmayan karakteri için bir yıldız işareti (\*) belirtin. Dinamik kuyruk adı için tek bir yıldız imi (\*) belirlemeyin.
- Kuyruk yöneticisinin tam adı oluşturmasına izin verin. Bu yöntemi kullanmak için, *DynamicQName* alanının ilk karakter konumunda bir yıldız işareti (\*) belirtin.

Bu yöntemlere ilişkin ek bilgi için DynamicQName (Dinamik QName) alanının tanımına bakın.

Dinamik ve Model kuyruklar alanında dinamik kuyruklara ilişkin daha fazla bilgi vardır.

### **Uzak Kuyrukların Açılması**

Uzak kuyruk, uygulamanın bağlı olduğu kuyruk yöneticisinin sahip olduğu bir kuyruktır.

Uzak bir kuyruğu açmak için, MQOPEN çağrısını yerel bir kuyruk olarak kullanın. Kuyruğun adını aşağıdaki gibi belirtebilirsiniz:

1. MQOD yapısının *ObjectName* alanında, uzak kuyruğun adını *yerel* kuyruk yöneticisi tarafından bilindiği şekilde belirtin.

**Not:** Bu durumda *ObjectQMGrName* alanını boş bırakın.

2. MQOD yapısının *ObjectName* alanında, *uzak* kuyruk yöneticisinde bilindiği gibi uzak kuyruğun adını belirtin. *ObjectQMGrName* alanında aşağıdakilerden birini belirtin:

- Uzak kuyruk yöneticisiyle aynı adı taşıyan iletim kuyruğunun adı. Ad ve büyük/küçük harf (büyük harf, küçük harf ya da bir karışım) *tam olarak* eşleşmelidir.
- Hedef kuyruk yöneticisine ya da iletim kuyruğuna çözülen, kuyruk yöneticisi diğer adı nesnesinin adı.

Bu, kuyruğun yöneticisine iletinin yerini ve oraya ulaşmak için gereken iletim kuyruğunu belirtir.

3. *DefXmitQname* destekleniyorsa, MQOD yapısının *ObjectName* alanında, *remote* kuyruk yöneticisi tarafından bilinen uzak kuyruğun adını belirtin.

**Not:** *ObjectQMGrName* alanını uzak kuyruk yöneticisinin adına ayarlayın (bu durumda boş bırakılamaz).

MQOL ' u çağırdığınızda yalnızca yerel adların geçerliliği denetlenir; son denetim, kullanılacak iletim kuyruğunun varolması içindir.

Bu yöntemler Çizelge 118 sayfa 783 içinde özetlenmiştir.

### **MQCLOSE çağrısını kullanarak nesnelere kapatılıyor**

Bir nesneyi kapatmak için, MQCLOSE çağrısını kullanın.

Nesne kuyruksa, aşağıdakine dikkat edin:

- Kapatmadan önce geçici bir dinamik kuyruğu boşaltmanıza gerek yoktur.

Geçici bir dinamik kuyruğu kapattığınızda, kuyruğun üzerinde hala üzerinde olabilecek iletilerle birlikte kuyruk silinir. Kuyruğa karşı işlenmemiş MQGET, MQPUT ya da MQPUT1 çağrılarını varsa bu doğru olur.

- IBM MQ for z/OS' ta, o kuyruk için MQGMO\_SET\_SIGNAL seçeneği bekleyen bir MQGET isteğiniz varsa, bunlar iptal edilir.
- Kuyruğu MQOO\_BROOK seçeneğini kullanarak açdıysanız, göz atma imleciniz yok edilir.

Kapanış, eşitleme noktalarıyla ilgisiz olduğundan, eşitleme noktasından önce ya da sonra kuyrukları kapatabilirsiniz.

MQCLOSE çağrısına giriş olarak, aşağıdaki bilgileri sağlamanız gerekir:

- Bağlantı tanıtıcısı. Bunu açmak için kullanılan bağlantı tanıtıcısını kullanın ya da diğer bir seçenek olarak, z/OS üzerindeki CICS uygulamaları için, değişmez MQHC\_DEF\_HCONN (sıfır değerine sahip) değerini belirtebilirsiniz.
- Kapatmak istediğiniz nesnenin saptanıdır. Bu işlemi, MQOPEN çağrısının çıkışından alın.
- *Options* alanında MQCO\_NONE (kalıcı bir dinamik kuyruğu kapatmadığınız sürece).
- Kuyruk yöneticisinin, üzerinde hala ileti olup olmadığını (kalıcı bir dinamik kuyruğu kapattığında) bile silmesi gerekip gerekmediğini belirlemek için denetim seçeneği.

MQCLOSE komutunun çıkışı şöyledir:

- Tamamlanma kodu
- Neden kodu
- Nesne tanıtıcısı, MQHO\_UNUSABLE\_HOBJ değerini ilk durumuna getirir

Descriptions of the parameters of the MQCLOSE call are given in [MQCLOSE](#).

## İletileri Kuyruğa Koyma

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

İletileri kuyruğa koymak için MQPUT çağrısını kullanın. MQPUT ' u, ilk MQOPEN çağrısını izleyerek, aynı kuyruğa birden çok ileti koymak için sık olarak kullanabilirsiniz. Tüm iletilerinizi kuyruğa koyduğunuzda MQCLOSE ' yi çağırın.

Bir kuyruğa tek bir ileti koymak ve kuyruğu hemen sonra kapatmak isterseniz, MQPUT1 çağrısını kullanabilirsiniz. MQPUT1 , aşağıdaki çağrılar sırasıyla aynı işlevleri gerçekleştirir:

- MQOPEN
- MQPUT
- MQCLOSE

Genellikle, kuyruğa koymak için birden çok iletiniz varsa, MQPUT çağrısını kullanmak daha verimli olur. Bu, iletinin boyutuna ve üzerinde çalışmakta olduğunuz altyapıya bağlıdır.

Bir kuyruğa ileti yerleştirilmesiyle ilgili daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- [“MQPUT çağrısını kullanarak iletileri yerel bir kuyruğa koyma” sayfa 792](#)
- [“İletileri Uzak Kuyruğa Koyma” sayfa 797](#)
- [“İletiyeye ilişkin özelliklerin ayarlanması” sayfa 797](#)
- [“İleti bağlamı bilgilerinin denetlenmesi” sayfa 797](#)
- [“MQPUT1 çağrısını kullanarak bir ileti kuyruğa konması” sayfa 799](#)
- [“Dağıtım listeleri” sayfa 801](#)
- [“Put çağrılarının başarısız olduğu bazı durumlar” sayfa 805](#)

### İlgili kavramlar

[“Message Queue Interface-Genel Bakış” sayfa 760](#)

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OS üzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

[“IBM MQ for z/OS üzerinde IMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

## ***MQPUT çağrısını kullanarak iletileri yerel bir kuyruğa koyma***

MQPUT çağrısını kullanarak iletileri yerel bir kuyruğa koyma hakkında bilgi edinmek için bu bilgileri kullanın.

MQPUT çağrısına giriş olarak, aşağıdaki bilgileri sağlamanız gerekir:

- Bir bağlantı tanıtıcısı (Hconn).
- Bir kuyruk tanıtıcısı (Hobj).
- Kuyruğa koymak istediğiniz iletinin açıklaması. Bu, ileti tanımlayıcı yapısı (MQMD) biçimidir.
- Denetim bilgileri, bir put-message options yapısı (MQPMO) biçiminde olur.
- İleti içinde yer alan verilerin uzunluğu (MQlong).
- İleti verilerinin kendisi.

MQPUT çağrısından çıkış aşağıdaki gibidir:

- Neden kodu (MQUBE)
- Tamamlanma kodu (MQUBE)

Arama başarıyla tamamlanırsa, seçenek yapılarınızı ve ileti tanımlayıcı yapınızı da döndürür. Çağrı, seçenek yapınızı, iletinin adını ve iletinin gönderildiği kuyruk yöneticisini gösterecek şekilde değiştirir. Kuyruk yöneticisinin koymakta olduğunuz iletinin tanıtıcısı için benzersiz bir değer oluşturduğunu (MQMD yapısının *MsgId* alanında ikili sıfır belirterek) istediyseniz, çağrı size bu yapıyı döndürmeden önce *MsgId* alanına değeri ekler. Başka bir MQPUT yayınlamadan önce bu değeri ilk durumuna getirin.

[MQPUT](#) içinde MQPUT çağrısının bir açıklaması var.

MQPUT çağrısına giriş olarak gereken bilgilere ilişkin ek bilgi için aşağıdaki bağlantılara bakın:

- [“Tanıtıcıların belirtilmesi” sayfa 793](#)
- [“MQMD yapısıyla iletilerin tanımlanması” sayfa 793](#)
- [“MQPMO yapısını kullanarak seçenekleri belirtme” sayfa 793](#)
- [“İletinizdeki veriler” sayfa 796](#)



- [“İleti koyma: İleti tanıtıcılarını kullanma” sayfa 797](#)

## Tanıtıcıların belirtilmesi

z/OS uygulamalarında CICS içindeki bağlantı tanıtıcısı (*Hconn*) için, değişmez MQHC\_DEF\_HCONN değerini (sıfır değerine sahip) belirleyebilir ya da MQCONN ya da MQCONNX çağrısının döndürdüğü bağlantı tanıtıcısını kullanabilirsiniz. Diğer uygulamalar için, her zaman MQCONN ya da MQCONNX çağrısının döndürdüğü bağlantı tanıtıcısını kullanın.

Çalışmakta olduğunuz ortam ne olursa olsun, MQOPEN çağrısının döndürdüğü kuyruk tanıtıcısını (*Hobj*) kullanın.

## MQMD yapısıyla iletilerin tanımlanması

İleti tanımlayıcı yapısı (MQMD), MQPUT ve MQPUT1 çağrılarında ilişkin bir giriş/çıkış değiştirgeci. Kuyruğa koymakta olduğunuz iletiyi tanımlamak için bu programı kullanın.

İleti için MQPRI\_PRIORITY\_AS\_Q\_DEF ya da MQPER\_PERSISTENCE\_AS\_Q\_DEF değeri belirtilirse ve kuyruk bir küme kuyruğalıysa, kullanılan değerler MQPUT ' un çözdüğü kuyruklardır. Bu kuyruk MQPUT için geçersiz kılındıysa, çağrı başarısız olur. Ek bilgi için [Kuyruk yöneticisi kümesinin yapılandırılması](#) başlıklı konuya bakın.

**Not:** *MsgId* ve *CorrelId* 'in benzersiz olduğundan emin olmak için yeni bir ileti koymadan önce MQPMO\_NEW\_MSG\_ID ve MQPMO\_NEW\_CORREL\_ID' yi kullanın. Bu alanlardaki değerler başarılı bir MQPUT ' un (MQPUT) içinde döndürülür.

MQMD 'nin [“IBM MQ ileti” sayfa 16](#) ' de tanımladığı ve [MQMD](#) içinde yapının kendisiyle ilgili bir açıklaması olan ileti özelliklerine giriş var.

## MQPMO yapısını kullanarak seçenekleri belirtme

MQPUT ve MQPUT1 çağrılarında seçenekler geçirmek için MQPMO (İleti Koyma Seçeneği Ekle) yapısını kullanın.

Aşağıdaki kısımlar, bu yapının alanlarını doldurmada size yardımcı olur. There is a description of the structure in [MQPMO](#).

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 put-message options yapısı olarak tanımlar. Bu, 4 karakterlik bir alandır. Her zaman MQPMO\_STRUC\_ID değerini belirtin.

## S\u00fcr\u00fcm

Bu, yapının sürüm numarasını açıklar. Varsayılan değer MQPMO\_VERSION\_1' dir. MQPMO\_VERSION\_2 yazarsanız, dağıtım listelerini kullanabilirsiniz (bkz. "Dağıtım listeleri" sayfa 801 ). MQPMO\_VERSION\_3 değerini girerseniz, ileti çekme noktaları ve ileti özelliklerini kullanabilirsiniz. MQPMO\_CURRENT\_VERSION değerini girerseniz, uygulamanız her zaman en son düzeyi kullanacak şekilde ayarlanır.

## Seçenekler

Bu denetim, aşağıdakileri denetler:

- Put işleminin bir iş birimine dahil edilip edilip etmeyeceğini
- Bir iletiyle ne kadar bağlam bilgisi ilişkilendirilir?
- Bağlam bilgilerinin alındığı yer
- Kuyruk yöneticisi susturulmuş durumda olduğunda çağrılarının başarısız olup olmadığını
- Gruplamaya ya da bölümlenmeye izin verilip
- Yeni ileti tanıtıcısı ve ilinti tanıtıcısı oluşturma
- İletilerin ve bölümlerin bir kuyruğa konacağı sıralama düzeni
- Yerel kuyruk adlarının çözümlenip çözümlenmeyeceği

*Options* alanını varsayılan değere (MQPMO\_NONE) bırakırsanız, koyduğunuz ileti, onunla ilişkili varsayılan bağlam bilgilerini içerir.

Ayrıca, arama, eşitleme noktalarıyla çalışma biçiminin altyapıya göre belirlenmesine yol gösterir. Eşitleme noktası denetimi varsayılan değeri, z/OS ' ta yes (evet); diğer platformlar için ise hayır.

## Bağlam

Bu durum, ( *Options* alanında istenirse), bağlam bilgilerinin kopyalanacağı kuyruk tanıtıcısı adını belirtir.

İleti bağlamına giriş için bkz. "İleti bağlamı" sayfa 44. Bir iletide bağlam bilgilerini denetlemek için MQPMO yapısının kullanılmasıyla ilgili bilgi edinmek için bkz. "İleti bağlamı bilgilerinin denetlenmesi" sayfa 797.

## ResolvedQName

Bu, iletiyi almak için açılan kuyruğun adını (diğer ad adını çözümledikten sonra) içerir. Bu bir çıkış alanıdır.

## ResolvedQMgrAdı

Bu, *ResolvedQName* içinde kuyruğa sahip olan kuyruk yöneticisinin adını (diğer ad adını çözümledikten sonra) içerir. Bu bir çıkış alanıdır.

MQPMO, dağıtım listeleri için gereken alanları da barındırabilir (bkz. "Dağıtım listeleri" sayfa 801 ). Bu olanağı kullanmak istiyorsanız, MQPMO yapısının Sürüm 2 kullanılır. Bu, aşağıdaki alanları içerir:

## RecsPresent

Bu alan, dağıtım listesindeki kuyrukların sayısını, yani MQPMR (Put Message Records) ve karşılık gelen Yanıt Kayıtların (MQRR) sayısını içerir.

Girdiğiniz değer, MQOPEN ' da sağlanan Nesne Kayıtları sayısı ile aynı olabilir. Ancak, değer MQOPEN çağrısında sağlanan Nesne Kaydı sayısından azsa ya da herhangi bir İleti Kaydı Koyma değeri sağlıyorsa, tanımlanmamış olan kuyrukların değerleri, ileti tanımlayıcısı tarafından sağlanan varsayılan değerlerden alınır. Ayrıca, değer, sağlanan Nesne Kayıtları sayısından büyükse, Fazla Olan İleti Kayıtları dikkate alınmaz.

Aşağıdakilerden birini yapmanız için önerildiniz:

- Her hedeften bir rapor ya da yanıt almak istiyorsanız, MQOR yapısındaki gibi, aynı değeri girin ve *MsgId* alanlarını içeren MQPR ' leri kullanın. Bu *MsgId* alanlarını sıfırlarla başlatın ya da MQPMO\_NEW\_MSG\_ID değerini belirtin.

İletiyi kuyruğa koyduğunuzda, kuyruk yöneticisinin oluşturduğu *MsgId* değerleri, MQPMR; ' da kullanılabilir duruma gelir; bu değerleri, her bir raporla ya da yanıtla ilişkin hedefi tanımlamak için kullanabilirsiniz.

- Rapor ya da yanıt almak istemezseniz, aşağıdakilerden birini seçin:

1. If you want to identify destinations that fail immediately, you might still want to enter the same value in the *RecsPresent* field as appears in the MQOR structure and provide MQRRs to identify these destinations. Herhangi bir MQPMR belirtmeyin.
2. Başarısız olan hedefleri tanımlamak istemiyorsanız, *RecsPresent* alanına sıfır girin ve MQPR 'leri ya da MQRR' leri sağlamayın.

**Not:** MQPUT1 kullanıyorsanız, Yanıt Kaydı Göstergelerinin sayısı ve Yanıt Kaydı Offsets sayısı sıfır olmalıdır.

İleti Kayıtlarının (MQPMR) ve Yanıt Kayıtlarının (MQRR) tam açıklaması için [MQPMR](#) ve [MQRR](#) başlıklı konuya bakın.

### **PutMsgRecFields**

Bu, her Put Message Record (MQPMR) ' da hangi alanların bulunduğunu belirtir. Bu alanların bir listesi için bkz. [“MQPMR yapısının kullanılması”](#) sayfa 805.

### **PutMsgRecOffset ve PutMsgRecPtr**

Göstergeler (genellikle C içinde) ve görelî konumlar (genellikle COBOL ' de) İleti Kayıtlarını Koyma (MQPMR yapısına genel bakış için [“MQPMR yapısının kullanılması”](#) sayfa 805 başlıklı konuya bakın).

İlk Put Message Record (İleti Kaydı) ya da ilk put iletisi kaydının görelî konumunu belirtmek için *PutMsgRecOffset* alanını kullanarak *PutMsgRecPtr* alanını kullanın. Bu, MQPMO ' nun başlangıcından görelî konudur. *PutMsgRecFields* alanına bağılı olarak, *PutMsgRecOffset* ya da *PutMsgRecPtr* için boş olmayan bir deęer girin.

### **ResponseRecGörelî Konumu ve ResponseRecPtr**

Ayrıca, Yanıt Kayıtlarına yanıt vermek için işaretçiler ve görelî konumlar da kullanıyorsunuz (Yanıt Kayıtlarıyla ilgili daha fazla bilgi için bkz. [“MQRR yapısının kullanılması”](#) sayfa 804 ).

İlk Yanıt Kaydına ilişkin bir gösterge belirlemek için *ResponseRecPtr* alanını ya da ilk Yanıt Kaydının görelî konumunu belirtmek için *ResponseRecOffset* alanını kullanın. Bu, MQPMO yapısının başlangıcından görelî konudur. *ResponseRecOffset* ya da *ResponseRecPtr* için boş olmayan bir deęer girin.

**Not:** İletileri bir dağıtım listesine koymak için MQPUT1 kullanıyorsanız, *ResponseRecPtr* boş deęerli ya da sıfır olmalı ve *ResponseRecOffset* deęeri sıfır olmalıdır.

MQPMO yapısının 3. sürümü ek olarak aşağıdaki alanları da içerir:

### **OriginalMsgTanıtıcısı**

Bu alanda yapabildiğiniz kullanım, *Eylem* alanının deęerine bağılıdır. İlişkili ileti özellikleriyle yeni bir ileti yerleřtiriyorsanız, bu alanı önceden yaratmış olduęunuz ileti tanıtıcısı olarak ayarlayın ve özellikleri bu alana ayarlayın. Önceden alınan bir iletiye yanıt olarak bir rapor iletiyorsanız, yanıtlıyorsanız ya da bir rapor oluřturuyorsanız, bu alan o iletinin ileti tanıtıcısını içerir.

### **NewMsgTanıtıcısı**

If you specify a *NewMsgİřlemesi*, any properties associated with the handle override properties associated with the *OriginalMsgİřlemesi*. Daha fazla bilgi için bkz. [Action \(MQUZE\)](#).

### **İřlem**

Gerçekleřtirilmekte olan kont tipinin tipini belirtmek için bu alanı kullanın. Olası deęerler ve anlamları aşağıdaki gibidir:

#### **MQACTP\_YENİ**

Bu başka bir mesajla alakasız yeni bir mesaj.

#### **MQAKP\_ILERI**

Bu ileti önceden alındı ve řimdi iletiliyor.

#### **MQACTP\_CEVAPLA**

Bu ileti, önceden alınan bir iletinin yanıtına neden olur.

#### **MQACP\_REPORT**

Bu ileti, önceden alınan bir iletinin sonucu olarak oluřturulan bir rapordur.

Daha fazla bilgi için bkz. [Action \(MQUZE\)](#).

### PubLevel

Bu ileti bir yayınsa, hangi aboneliklerin hangi abonelikleri alacağını belirlemek için bu alanı ayarlayabilirsiniz. Yalnızca *SubLevel* değeri bu değerden küçük ya da bu değere eşit olan abonelikler bu yayını alır. Varsayılan değer, en yüksek düzey olan 9 'tır ve herhangi bir *SubLevel* ' e sahip aboneliklerin bu yayını alabileceği anlamına gelir.

### İletinizdeki veriler

MQPUT çağrısının **Buffer** parametresindeki verinizi içeren arabelleğin adresini verin. İletilerinizdeki verilere herhangi bir şey dahil edebilirsiniz. Ancak iletilerde bulunan veri miktarı, bunları işleyen uygulamanın performansını etkiler.

Veri büyüklüğü üst sınırı aşağıdaki tarafından belirlenir:

- Kuyruk yöneticisinin **MaxMsgLength** özniteliği
- İletiyi koymakta olduğunuz kuyruğun **MaxMsgLength** özniteliği
- IBM MQ tarafından eklenen ileti üstbilgisinin (ölü-harf üstbilgisi, MQDLH ve dağıtım listesi üstbilgisi, MQDH gibi) boyutu

Kuyruk yöneticisinin **MaxMsgLength** özniteliği, kuyruk yöneticisinin işleyebileceği ileti büyüklüğünü tutar. Bu, V6 ya da daha sonraki bir sürüm olan tüm IBM MQ ürünleri için 100 MB ' lik bir varsayılan değere sahiptir.

Bu özniteliğin değerini saptamak için, kuyruk yöneticisi nesnesindeki MQINQ çağrısını kullanın. Büyük iletiler için bu değeri değiştirebilirsiniz.

Kuyruğun **MaxMsgLength** özniteliği, kuyruğa koyabileceğiniz ileti boyutu üst sınırını belirler. Bu özniteliğin değerinden büyük bir boyuta sahip bir ileti yerleştirmeye çalışırsanız, MQPUT çağrılarınız başarısız olur. Bir iletiyi uzak bir kuyruğa yerleştiriyorsanız, başarıyla yerleştirebileceğiniz ileti boyutu üst sınırı, uzak kuyruğun **MaxMsgLength** özniteliği tarafından belirlenir ve bu ileti, iletinin hedef rotasına ve kullanılan kanallara ilişkin ara iletim kuyruklarının herhangi bir ara iletim kuyruğundan saptanır.

Bir MQPUT işlemi için, iletinin büyüklüğü hem kuyruğun, hem de kuyruk yöneticisinin **MaxMsgLength** özniteisinden küçük ya da ona eşit olmalıdır. Bu özniteliklerin değerleri bağımsızdır, ancak kuyruğun *MaxMsgLength* ' unu kuyruk yöneticisinden küçük ya da ona eşit bir değere ayarlamanız önerilir.

IBM MQ , aşağıdaki durumlarda iletilere üstbilgi bilgileri ekler:


- Uzak bir kuyruğa ileti yerleştirdiğinizde, IBM MQ iletiye bir iletim üstbilgisi yapısı (MQXQH) ekler. Bu yapı, hedef kuyruğun adını ve sahip olan kuyruk yöneticisini içerir.
- IBM MQ bir iletiyi uzak bir kuyruğa teslim edemezse, iletiyi ölü-mektup (teslim edilemeyen ileti) kuyruğuna yerleştirmeye çalışır. İletiyeye MQDLH yapısı ekler. Bu yapı, hedef kuyruğun adını ve iletinin ölü-mektup kuyruğuna konmasının nedenini içerir.
- Birden çok hedef kuyruğa ileti göndermek istiyorsanız, IBM MQ iletiye bir MQDH üstbilgisi ekler. Bu, bir iletim kuyruğunda yer alan bir dağıtım listesine ait olan bir iletide bulunan verileri açıklar. İleti uzunluğu üst sınırı için en iyi değer seçilirken bunu dikkate alın.
- İleti bir bölümse ya da bir gruptaki bir iletiyse, IBM MQ bir MQMDE ekleyebilir.

Bu yapılar [MQDH](#) ve [MQMDE](#) içinde açıklanmıştır.

İletilerinizde bu kuyruklar için izin verilen büyüklük üst sınırı varsa, bu üstbilgilerin eklenmesi, iletilerin artık çok büyük olduğu için, koyma işlemlerinin başarısız olduğu anlamına gelir. Koyma işlemlerinin başarısız olma olasılığını azaltmak için:

- İletilerinizin boyutunu, iletim ve ölü harf kuyruklarının **MaxMsgLength** öznitelikten daha küçük bir hale getiriniz. En azından MQ\_MSG\_HEADER\_LENGTH değişiminin (büyük dağıtım listeleri için daha fazla) değerine izin verin.
- Ölü-harfli kuyruğun **MaxMsgLength** özniteliğinin, ölü harf kuyruğunun sahibi olan kuyruk yöneticisinin *MaxMsgLength* ile aynı olarak ayarlandığından emin olun.

Kuyruk yöneticisine ilişkin öznitelikler ve kuyruğa alma değişmezleri, [kuyruk yöneticisine ilişkin öznitelikler](#) altında açıklanmıştır.

 Dağıtılmış bir kuyruklama ortamında teslim edilemeyen iletilerin nasıl işlendiği hakkında bilgi için bkz. [Teslim edilemeyen/işlenmemiş iletiler](#).

## İleti koyma: İleti tanıtıcılarını kullanma

MQPPO yapısında iki ileti tanıtıcısı bulunur, *OriginalMsgHandle* ve *NewMsgHandle* (Yeni Msg) tanıtıcısı. Bu ileti tanıtıcıları arasındaki ilişki, MQPPO *Action* alanının değeriyle tanımlanır.

Tüm ayrıntılar için bkz. [Eylem \(MQUZE\)](#). Bir iletiyi koymak için bir ileti tanıtıcısı gerekli değildir. Bunun amacı, özellikleri bir iletiyle ilişkilendirmeniz, bu nedenle yalnızca ileti özelliklerini kullanıyorsanız gereklidir.

### İletileri Uzak Kuyruğa Koyma

Bir iletiyi uzak bir kuyruğa (örneğin, uygulamanızın bağlı olduğu bir kuyruk yöneticisine ait bir kuyruk) yerel bir kuyruk yerine koymak istediğinizde, daha fazla dikkat edilmesi gereken tek dikkat, kuyruğun adını açtığınızda adın nasıl belirtildiğini belirtir. Bu, "[Uzak Kuyrukların Açılması](#)" sayfa 790 içinde açıklanmaktadır. Yerel bir kuyruk için MQPUT ya da MQPUT1 çağrısını kullanma şekliniz bir değişiklik yok.

Uzak ve iletim kuyruklarının kullanılmasına ilişkin ek bilgi için [IBM MQ dağıtılmış kuyruklama teknikleri](#) başlıklı konuya bakın.

### İletiyeye ilişkin özelliklerin ayarlanması

Ayarlamak istediğiniz her özellik için MQSETMP ' yi çağırın. İletiyeye yerleştirdiğinizde, MQPPO yapısının ileti tanıtıcısı ve işlem alanları ayarlanır.

Özellikleri bir iletiyle ilişkilendirmek için, iletinin bir ileti tanıtıcısı olması gerekir. MQCRTMH işlev çağrısını kullanarak bir ileti tanıtıcısı yaratın. Ayarlamak istediğiniz her özellik için bu ileti tanıtıcısını belirterek MQSETMP ' yi çağırın. MQSETMP kullanımını göstermek için bir örnek program ( amqsstma.c) sağlanmıştır.

Bu yeni bir iletiyse, bu iletiyi bir kuyruğa koyduğunuzda, MQPUT ya da MQPUT1 kullanarak, MQPPO ' daki OriginalMsgHandle alanını bu ileti tanıtıcısı değerine ayarlayın ve MQPPO Action alanını MQACTP\_NEW olarak ayarlayın (varsayılan değer budur).

Bu, önceden aldığınız bir iletiyse ve şimdi bu iletiyi iletiyor ya da yanıtladınız ya da buna yanıt olarak bir rapor gönderiyorsanız, özgün ileti tanıtıcısını MQPPO ' nun OriginalMsgHandle alanına ve NewMsgHandle alanındaki yeni ileti tanıtıcısını yerleştirdiniz. İşlem alanını, uygun olduğu şekilde MQACTP\_FORWARD, MQACTP\_REPLY ya da MQACP\_REPORT olarak ayarlayın.

Bir MQRFH2 üstbilgisinde önceden aldığınız bir iletiden özellikler varsa, bu iletiyi MQBUFMH çağrısını kullanarak ileti işleyici özelliklerine dönüştürebilirsiniz.

İletinizi, ileti özelliklerini işleyemeyen IBM WebSphere MQ 7.0 düzeyinden önceki bir düzeydeki kuyruk yöneticisinde bir kuyruğa yerleştiriyorsanız, özelliklerin nasıl işleneceğini belirtmek için kanal tanımındaki PropertyControl parametresini ayarlayabilirsiniz.

### İleti bağlamı bilgilerinin denetlenmesi

Bir iletiyi kuyruğa koymak için MQPUT ya da MQPUT1 çağrısını kullandığınızda, kuyruk yöneticisinin ileti tanımlayıcısına varsayılan bağlam bilgileri eklemesini belirleyebilirsiniz. Uygun yetki düzeyine sahip uygulamalar ek bağlam bilgileri ekleyebilir. Bağlam bilgilerini denetlemek için, MQPPO yapısındaki seçenekler alanını kullanabilirsiniz.

İleti bağlamı bilgileri, iletiyi alan uygulamanın, iletiyi oluşturan iletiyi bulmasını sağlar. Tüm bağlam bilgileri, ileti tanımlayıcısının bağlam alanlarında saklanır. Bilgi tipi, kimlik, kaynak ve kullanıcı bağlamı bilgilerine rastlar.

Bağlam bilgilerini denetlemek için, MQPPO yapısındaki *Options* alanını kullanın.

Bağlam bilgileri için herhangi bir seçenek belirlemezseniz, kuyruk yöneticisi, iletiniz için oluşturduğu kimlik ve bağlam bilgileriyle önceden ileti tanımlayıcısında olabilecek bağlam bilgilerinin üzerine yazar. Bu,

MQPMO\_DEFAULT\_CONTEXT seçeneğinin belirlendiği şekliyle aynıdır.Yeni bir ileti yarattığınızda (örneğin, bir sorgu ekranından kullanıcı girişi işlerken) bu varsayılan bağlam bilgilerinin olmasını isteyebilirsiniz.

İletinizle ilişkilendirilmiş bir bağlam bilgisi istemiyorsanız, MQPMO\_NO\_CONTEXT seçeneğini kullanın. Bağlam olmadan bir ileti yerleştirilirken, IBM MQ tarafından yapılan tüm yetki denetimleri boş bir kullanıcı kimliği kullanılarak yapılır. Boş bir kullanıcı kimliği, IBM MQ kaynaklarına belirtik yetki atanamaz; ancak, 'kimse' grubunun bir üyesi olarak işlem görür. nobody özel grubuna ilişkin ayrıntılar için [Kurulabilir hizmetler arabirimi başvuru bilgiler](#) başlıklı konuya bakın.

İzleyen kısımlarda belirtilen MQOO\_ seçeneğini ve MQPMO\_ seçeneğini kullanarak MQPUT ' u izleyen bağlam ayarını kullanarak bağlam ayarını yapabilirsiniz. Bağlam ayarını yalnızca bir MQPUT1 kullanarak da yapabilirsiniz; bu durumda, aşağıdaki kısımlarda belirtilen MQPMO\_ seçeneğini belirlemeniz yeterlidir.

Bu konunun aşağıdaki bölümleri, kimlik bağlamı, kullanıcı bağlamı ve tüm bağlamın kullanımını açıklar.

- [“Kimlik bağlamı geçirme” sayfa 798](#)
- [“Kullanıcı bağlamı geçirme” sayfa 798](#)
- [“Tüm bağlamın geçirilmesi” sayfa 799](#)
- [“Kimlik bağlamının ayarlanması” sayfa 799](#)
- [“Kullanıcı bağlamını ayarlama” sayfa 799](#)
- [“Tüm bağlamın ayarlanması” sayfa 799](#)

## Kimlik bağlamı geçirme

Genel olarak, programlar, bir uygulamanın son hedefine ulaşıncaya kadar, bir uygulama çevresinde kimlik bağlamı bilgilerini iletiden iletiye iletmelidir.

Programlar, verileri her değiştirilerinde kaynak bağlamı bilgisini değiştirmelidir. Ancak, herhangi bir bağlam bilgisini değiştirmek ya da ayarlamak isteyen uygulamaların uygun yetki düzeyine sahip olması gerekir. Kuyruk yöneticisi, uygulamalar kuyrukları alarken bu yetkiyi denetler; MQOPEN çağrısına ilişkin uygun bağlam seçeneklerini kullanma yetkisi olmalıdır.

Uygulamanız bir ileti alırsa, iletiden verileri işler, daha sonra değiştirilen verileri başka bir iletiye koyar (olasılıkla başka bir uygulama tarafından işlenebilir), uygulamanın kimlik bağlamı bilgilerini özgün iletiden yeni iletiye geçirmesi gerekir. Kuyruk yöneticisinin kaynak bağlamı bilgilerini yaratmasına izin verebilirsiniz.

Bağlam bilgilerini özgün iletiden kaydetmek için, iletiyi almak için kuyruğu açtığınızda MQOO\_SAVE\_ALL\_CONTEXT seçeneğini kullanın. Bu, MQOPEN çağrısıyla birlikte kullandığınız diğer seçeneklerin yanı sıra. Ancak, yalnızca iletiye göz attığınızda bağlam bilgilerini kaydedemezsiniz.

İkinci iletiyi oluşturduğunuzda:

- MQOO\_PASS\_IDENTITY\_CONTEXT seçeneğini kullanarak kuyruğu açın (MQOO\_OUTPUT seçeneğinin yanı sıra).
- İletiyeye ilişkin seçenekler yapısının *Context* alanında, bağlam bilgilerini sakladığınız kuyruğun tanıtıcısını verin.
- Put-message options yapısının *Options* alanında, MQPMO\_PASS\_IDENTITY\_CONTEXT seçeneğini belirtin.

## Kullanıcı bağlamı geçirme

Yalnızca kullanıcı bağlamını geçmeyi seçemezsiniz. Bir ileti yerleştirilirken kullanıcı bağlamını geçirmek için, MQPMO\_PASS\_ALL\_CONTEXT belirtin. Kullanıcı bağlamındaki özellikler, kaynak bağlamla aynı şekilde geçirilir.

Bir MQPUT ya da MQPUT1 gerçekleştirildiğinde ve bağlam geçiriliyorsa, kullanıcı bağlamındaki tüm özellikler, alınan iletiden alınan iletiye iletir. Koyma uygulamasının değiştirilmiş olduğu kullanıcı bağlamı özellikleri özgün değerleriyle konmuştur. Koyma uygulamasının silmiş olduğu kullanıcı bağlamı özellikleri, koyma iletisinde geri yüklenir. Ekleme uygulamasının iletiye eklediği kullanıcı bağlamı özellikleri korunur.

## Tüm bağlamın geçirilmesi

Uygulamanız bir ileti alıyorsa ve ileti verilerini (değiştirilmemiş) başka bir iletiye koyarsa, uygulamanın tümünü (kimlik, kaynak ve kullanıcı) bağlam bilgilerini özgün iletiden yeni iletiye iletmesi gerekir. Bunu yapabilen bir uygulama örneği, iletileri bir kuyruktan diğerine taşıyan bir ileti modemdir.

MQOO\_PASS\_PASS\_ALL\_CONTEXT ve MQPMO\_PASS\_ALL\_CONTEXT koyma iletisi seçeneğini kullanmanız dışında, kimlik bağlamını geçirmek için aynı yordamı izleyin.

## Kimlik bağlamının ayarlanması

Bir ileti için kimlik bağlamı bilgilerini ayarlamak istiyorsanız:

- MQOO\_SET\_IDENTITY\_CONTEXT seçeneğini kullanarak kuyruğu açın.
- MQPMO\_SET\_IDENTITY\_CONTEXT seçeneğini belirterek, iletiyi kuyruğa koyun. İleti tanımlayıcısında, gereksinim duyduğunuz kimlik bağlamı bilgilerini belirtin.

**Not:** MQOO\_SET\_IDENTITY\_CONTEXT ve MQPMO\_SET\_IDENTITY\_CONTEXT seçeneklerini kullanarak, kimlik bağlamı alanlarının bazılarını (ancak tümü değil) ayarladığınızda, kuyruk yöneticisinin diğer alanların hiçbirini ayarlamadığını fark etmek önemlidir.

İleti bağlamı seçeneklerinden herhangi birini değiştirmek için, aramayı yayınlamak için gereken yetkilerin olması 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 gerçekleştirdiğinizde, ileti özelliği tanımlayıcısının (MQPD) Bağlam alanını MQPD\_USER\_CONTEXT olarak ayarlayın.

Kullanıcı bağlamında bir özellik ayarlamak için özel bir yetkiye sahip olmamanız gerekir. Kullanıcı bağlamında MQOO\_SET\_\* ya da MQPMO\_SET\_\* bağlam seçenekleri yok.

## Tüm bağlamın ayarlanması

Bir ileti için hem kimlik, hem de kaynak bağlamı bilgilerini ayarlamak istiyorsanız:

1. MQOO\_SET\_ALL\_CONTEXT seçeneğini kullanarak kuyruğu açın.
2. MQPMO\_SET\_ALL\_CONTEXT seçeneğini belirterek, iletiyi kuyruğa koyun. İleti tanımlayıcısında, gereksinim duyduğunuz kimlik ve kaynak bağlamı bilgilerini belirtin.

Her bir bağlam ayarı tipi için uygun yetki gereklidir.

### İlgili kavramlar

[“İleti bağlamı” sayfa 44](#)

*İleti bağlamı* bilgileri, iletiyi alan uygulamanın, iletiyi oluşturan iletiyi bulmasını sağlayan bir uygulamaya olanak tanır.

### İlgili başvurular

[“İleti bağlamına ilişkin MQAÇ seçenekleri” sayfa 789](#)

Bir iletiyi bir kuyruğa koyduğunuzda bağlam bilgilerini bir iletiyle ilişkilendirebilmek istiyorsanız, kuyruğu açtığınızda ileti bağlamı seçeneklerinden birini kullanmanız gerekir.

## MQPUT1 çağrısını kullanarak bir ileti kuyruğa konması

Kuyruğa tek bir ileti koyduktan hemen sonra kuyruğu kapatmak istediğinizde MQPUT1 çağrısını kullanın. Örneğin, bir sunucu uygulaması, farklı kuyrukların her birine yanıt gönderirken MQPUT1 çağrısını kullanacaktır.

MQPUT1 , MQPUT ve onu izleyen MQCLOSE ' yi çağırarak işlevsel olarak eşdeğerdir. MQPUT ve MQPUT1 çağrılarında ilişkin sözdizimindeki tek fark, MQPUT için bir nesne tanımlayıcı belirtmenizi sağlar; MQPUT1 için, MQPAN ' da tanımlandığı gibi bir nesne tanımlayıcı yapısı (MQOD) belirtin (bkz. [“Nesnelerin tanımlanması](#)

(MQOD yapısı)” sayfa 783 ). Bunun nedeni, MQPUT1 'in açılması gerektiği kuyruğunuz hakkında bilgi vermeniz gerektiğinden, MQPUT' u çağırduğunuzda, kuyruk zaten açık olmalıdır.

MQPUT1 çağırısına giriş olarak, aşağıdaki bilgileri sağlamanız gerekir:

- Bağlantı tanıtıcısı.
- Açmak istediğiniz nesneye ilişkin bir açıklama. Bu, bir nesne tanımlayıcı yapısı biçimidir (MQOD).
- Kuyruğa koymak istediğiniz iletinin açıklaması. Bu, ileti tanımlayıcı yapısı (MQMD) biçimidir.
- Put-message options structure (MQPMO) yapısı biçiminde bilgileri denetleyin.
- İleti içinde yer alan verilerin uzunluğu (MQlong).
- İleti verilerinin adresi.

The output from MQPUT1 is:

- Tamamlanma kodu
- Neden kodu

Arama başarıyla tamamlanırsa, seçenek yapılarınızı ve ileti tanımlayıcı yapınızı da döndürür. Çağrı, seçenek yapınızı, iletinin adını ve iletinin gönderildiği kuyruk yöneticisini gösterecek şekilde değiştirir. Kuyruk yöneticisinin koymakta olduğunuz iletinin tanıtıcısı için benzersiz bir değer oluşturmasını isterseniz (MQMD yapısının *MsgId* alanında ikili sıfır belirleyerek), arama bu yapıyı size döndürmeden önce *MsgId* alanına değeri ekler.

**Not:** Model kuyruğu adıyla MQPUT1 kullanamazsınız; ancak, bir model kuyruğu açıldıktan sonra, dinamik kuyruk için bir MQPUT1 komutu verebilirsiniz.

MQPUT1 için altı giriş parametresi şunlardır:

#### **Hconn**

Bu bir bağlantı tanıtıcısı. CICS uygulamaları için, MQHC\_DEF\_HCONN değişmezini (sıfır değerine sahip) belirtebilir ya da MQCONN ya da MQCONNX çağırısının döndürdüğü bağlantı tanıtıcısını kullanabilirsiniz. Diğer programlar için, her zaman MQCONN ya da MQCONNX çağırısının döndürdüğü bağlantı tanıtıcısını kullanın.

#### **ObjDesc**

Bu bir nesne tanımlayıcısı yapısıdır (MQOD).

*ObjectName* ve *ObjectQMgrName* alanlarında, ileti koymak istediğiniz kuyruğun adını ve bu kuyruğun sahibi olan kuyruk yöneticisinin adını verin.

Model kuyruklarını kullanmadığı için, MQPUT1 çağırısı için *DynamicQName* alanı yoksayılr.

Kuyruğu açmak için yetki vermek üzere kullanılacak diğer bir kullanıcı kimliğini aday göstermek istiyorsanız, *AlternateUserId* alanını kullanın.

#### **MsgDesc**

Bu bir ileti tanımlayıcı yapısıdır (MQMD). MQPUT çağırısıyla olduğu gibi, kuyruğa koymakta olduğunuz iletiyi tanımlamak için bu yapıyı kullanın.

#### **PutMsgOpts**

Bu bir put-message options yapısıdır (MQPMO). Bunu, MQPUT çağırısı için kullanırken kullanın (bkz. [“MQPMO yapısını kullanarak seçenekleri belirtme” sayfa 793](#) ).

*Options* alanı sıfır olarak ayarlandığında, kuyruk yöneticisi, kuyruğa erişim yetkisi için sınamalar gerçekleştirirken kendi kullanıcı kimliğinizi kullanır. Ayrıca, kuyruk yöneticisi, MQOD yapısının *AlternateUserId* alanında verilen diğer kullanıcı kimliğini yoksayar.

#### **BufferLength**

Bu, iletinizin uzunluğudur.

#### **Buffer**

Bu, iletinizin metnini içeren arabellek alanıdır.



Kümeleri kullandığınızda, MQPUT1 , MQOO\_BIND\_NOT\_FIXED etkin olduğu gibi çalışır. İletinin nereye gönderileceğini belirlemek için, uygulamaların MQPMO yapısındaki çözülmüş alanları MQPO yapısında kullanması gerekir. Ek bilgi için [Kuyruk yöneticisi kümesinin yapılandırılması](#) başlıklı konuya bakın.

MQPUT1 içinde MQPUT1 çağrısının bir açıklaması vardır.

### **Dağıtım listeleri**

**IBM MQ for z/OS üzerinde desteklenmez.** Dağıtım listeleri, tek bir MQPUT ya da MQPUT1 çağrısında birden çok hedefe bir ileti koymanızı sağlar. Tek bir MQOPER çağrısı birden çok kuyruk açabilir ve sonra tek bir MQPUT çağrısı bu kuyrukların her birine bir ileti yerleştirebilir. Bu süreç için kullanılan MQI yapılarından alınan bazı genel bilgiler, dağıtım listesinde yer alan tek tek hedeflerle ilgili belirli bilgiler tarafından geçersiz kılınabilir.



**Uyarı:** Dağıtım listeleri, konu nesnelere 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.

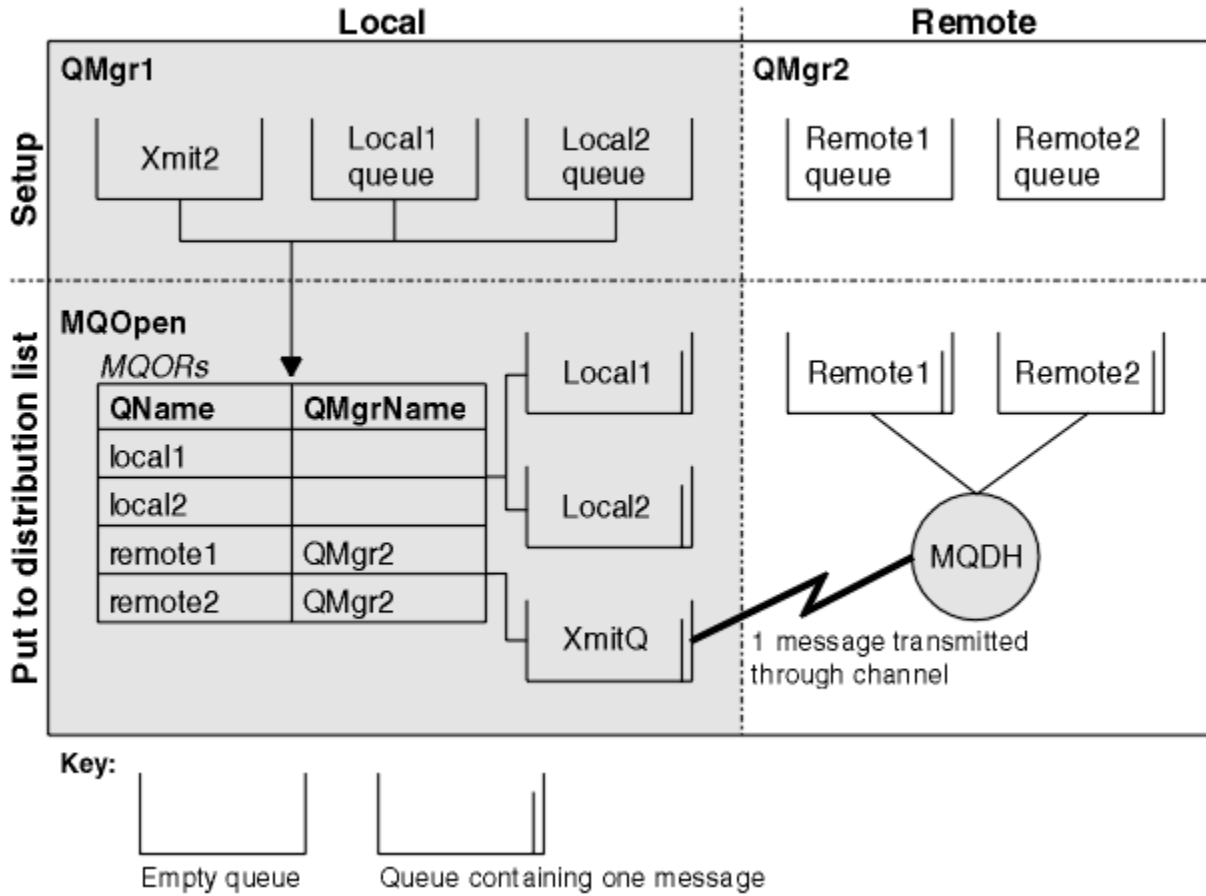
Bir MQOPED çağrısı yayınlandığında, Nesne Tanımlayıcısından (MQOD) soysal bilgiler alınır. *Version* alanında MQOD\_VERSION\_2 değerini ve *RecsPresent* alanında sıfırdan büyük bir değer belirtirseniz, *Hobj* , kuyruk yerine bir listenin (bir ya da daha çok kuyruğun) tanıtıcı değeri olarak tanımlanabilir. Bu durumda, hedef ayrıntılarını (yani, *ObjectName* ve *ObjectQMgrName*) veren nesne kayıtları (MQORS) aracılığıyla belirli bilgiler verilir.

Nesne tanıtıcısı (*Hobj*) MQPUT çağrısına iletilir ve tek bir kuyruk yerine bir listeye girmenize olanak tanır.

Kuyruklara bir ileti konduğunda (MQPUT) soysal bilgiler, Put Message Option structure (MQPMO) ve Message Descriptor (MQMD)) olanağından alınır. Belirli bilgiler, Put Message Records (MQPMR ' ler) biçiminde verilir.

Yanıt Kayıtları (MQRR), her hedef kuyruğa özgü bir tamamlanma kodu ve neden kodu alabilir.

[Şekil 61 sayfa 802](#) , dağıtım listelerinin nasıl çalıştığını gösterir.



Şekil 61. Dağıtım listeleri nasıl çalışır

#### Dağıtım listelerini açma

Bir dağıtım listesi açmak için MQOPEN çağrısını kullanın ve arama seçeneklerini kullanarak, listeye ne yapmak istediğinizi belirtin.

MQOPER 'a giriş olarak, aşağıdaki bilgileri sağlamanız gerekir:

- Bağlantı tanıtıcısı (açıklama için “İletileri Kuyruğa Koyma” sayfa 791 konusuna bakın)
- Nesne Tanımlayıcı yapısındaki sosyal bilgiler (MQOD)
- Nesne Kaydı yapısını (MQOR) kullanarak açmak istediğiniz her kuyruğun adı.

MQXX\_ENCODE\_CASE\_ONE open komutunun çıkışı şöyledir:

- Dağıtım listesine erişiminizi temsil eden bir nesne tanıtıcısı
- Sosyal bir tamamlanma kodu
- Sosyal neden kodu
- Yanıt Kayıtları (isteğe bağlı), her hedef için bir tamamlanma kodu ve neden içeren

#### MQOD yapısının kullanılması

Açmak istediğiniz kuyrukları tanımlamak için MQOD yapısını kullanın.

Bir dağıtım listesi tanımlamak için, *Version* alanında MQOD\_VERSION\_2 belirtmeli, *RecsPresent* alanında sıfırdan büyük bir değer ve *ObjectType* alanında MQOT\_Q belirtilmelidir. MQOD yapısının tüm alanlarının bir açıklaması için bkz. [MQOD](#).

#### MQOR yapısının kullanılması

Her hedef için bir MQOR yapısı sağlayın.

Yapı, hedef kuyruğu ve kuyruk yöneticisi adlarını içerir. MQOD 'daki *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 ek bilgi için bkz. [ObjectName](#) ve [ObjectQMgrAd](#).

Hedef kuyrukları iki şekilde belirtebilirsiniz:

- *ObjectRecOffset* görelili konum alanını kullanarak.

In this case, the application must declare its own structure containing an MQOD structure, followed by the array of MQOR records (with as many array elements as are needed), and set *ObjectRecOffset* to the offset of the first element in the array from the start of the MQOD. Bu görelili konumun doğru olduğundan emin olun.

Bu programlar, uygulamanın çalıştığı tüm ortamlarda kullanılabilir, programlama dili tarafından sağlanan yerleşik tesislerin kullanılması önerilir. Aşağıdaki kod, COBOL programlama diline ilişkin bu tekniği göstermektedir:

```
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 ortamlardaki gerekli yerleşik olanakları desteklemiyorsa, MQOD\_CURRENT\_LENGTH değişimini 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 işlev yalnızca MQOD yapısı ve MQOR kayıtları dizisi bitişik olduğunda doğru çalışır; derleyici, MQOD ile MQOR dizisi arasına atlamalı byte ekler eklediye, bunlar *ObjectRecOffset* içinde saklanan değere eklenmelidir.

İşaretçi veri tipini desteklemeyen programlama dilleri için *ObjectRecOffset* kullanılması önerilir ya da gösterge verileri tipi, farklı ortamlara (örneğin, COBOL programlama dili) portatif olmayan bir biçimde uygulanır.

- *ObjectRecPtr* işaretçi alanını kullanarak.

Bu durumda uygulama, MQOR yapılarının dizisini, MQOD yapısından ayrı olarak bildirebilir ve *ObjectRecPtr* 'yi dizinin adresine ayarlayabilir. Aşağıdaki kod C programlama dili için bu tekniği göstermektedir:

```
MQOD MyMqod;  
MQOR MyMqor[100];  
MyMqod.ObjectRecPtr = MyMqor;
```

İşaretçi veri tipini farklı ortamlara (örneğin, C programlama dili gibi) destekleyen programlama dilleri için *ObjectRecPtr* 'nin kullanılması önerilir.

Seçtiğiniz teknik hangisinden olursa olsun, *ObjectRecOffset* ve *ObjectRecPtr* 'den birini kullanmanız gerekir. Her ikisi de sıfır ya da her ikisi de sıfır değilse, çağrı neden kodu MQRC\_OBJECT\_RECORDS\_ERROR ile başarısız olur.

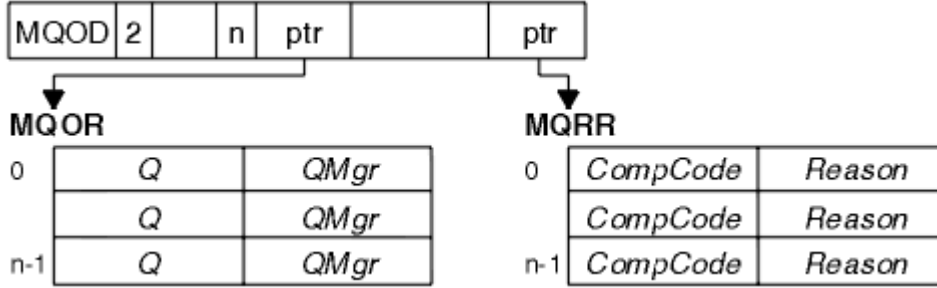
## MQRR yapısının kullanılması

Bu yapılar hedef belirtimidir; her Yanıt Kaydı, dağıtım listesinin her kuyruğu için bir *CompCode* ve *Reason* alanı içerir. Herhangi bir sorunun yalanı ayırt edebilmeyi sağlamak için bu yapıyı kullanmanız gerekir.

Örneğin, MQRC\_MULTIPLE\_REASONS bir neden kodu alırsanız ve dağıtım listeniz beş hedef kuyruk içeriyorsa, bu yapıyı kullanmamanız durumunda sorunların hangi kuyruklara uygulanacağını bilmeyeceksiniz. Ancak, her hedef için bir tamamlama kodunuz ve neden kodunuz varsa, hataları daha kolay bulabilirsiniz.

MQRR yapısıyla ilgili ek bilgi için [MQRR](#) konusuna bakın.

Şekil 62 sayfa 804 , C içinde bir dağıtım listesinin nasıl açılacağını gösterir.



Şekil 62. C içinde bir dağıtım listesinin açılması

Şekil 63 sayfa 804 , COBOL ' da bir dağıtım listesinin nasıl açılacağını gösterir.



Şekil 63. COBOL ' da bir dağıtım listesinin açılması

## MQOPER seçeneklerinin kullanılması

Bir dağıtım listesi açarken aşağıdaki seçenekleri belirleyebilirsiniz:

- MQOO\_OUTPUT
- 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 781](#) .

### İletileri Dağıtım Listesine Koyma

Bir dağıtım listesine ileti koymak için, MQPUT ya da MQPUT1' i kullanabilirsiniz.

Giriş olarak şu bilgileri sağlamanız gerekir:

- Bağlantı tanıtıcısı (açıklama için [“İletileri Kuyruğa Koyma” sayfa 791](#) konusuna bakın).
- Bir nesne tanıtıcısı. Bir dağıtım listesi MQOPEN kullanılarak açılırsa, *Hobj* yalnızca listeye koymanıza izin verir.
- Bir ileti tanımlayıcı yapısı (MQMD). Bu yapının bir açıklaması için bkz. [MQMD](#) .
- Put-message option structure (MQPMO) biçiminde bilgileri denetleyin. MQPMO yapısının alanlarının tamamlanmasına ilişkin bilgi için bkz. [“MQPMO yapısını kullanarak seçenekleri belirtme” sayfa 793](#) .
- İleti Koyma Kayıtları (MQPMR) biçimindeki denetim bilgileri.
- İleti içinde yer alan 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 bir şekilde tanımlamak isteyebileceğiniz bazı alanlar için hedefe özgü bilgileri verir.

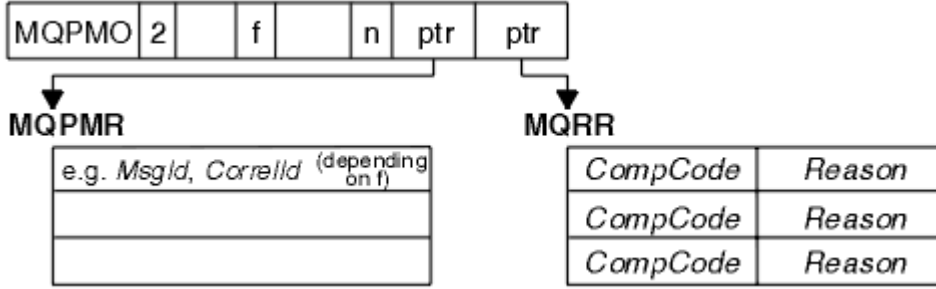
Bu alanlara ilişkin açıklamalar için [MQPMR](#) başlıklı konuya bakın.

Her kaydın içeriği, MQPMO ' nun *PutMsgRecFields* alanında verilen bilgilere bağlıdır. Örneğin, dağıtım listelerinin kullanımını gösteren AMQSPTLO.C örnek programında (bir açıklama için “Dağıtım Listesi örnek programı” sayfa 1122 başlıklı konuya bakın), örnek MQPMR ' de *MsgId* ve *CorrelId* için değer sağlamayı seçer. Örnek programın bu bölümü şu şekilde görünür:

```
typedef struct
{
  MQBYTE24 MsgId;
  MQBYTE24 CorrelId;
} PutMsgRec;
...
/*****
MQLONG PutMsgRecFields=MQPMRF_MSG_ID | MQPMRF_CORREL_ID;
```

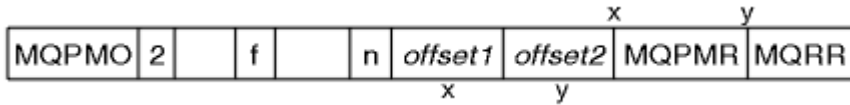
Bu, dağıtım listesinin her hedefi için *MsgId* ve *CorrelId* ' in sağlandığına işaret eder. Put Message Records, bir dizi olarak sağlanır.

Şekil 64 sayfa 805 , C içindeki bir dağıtım listesine nasıl bir ileti yerleştirebileceğinin gösterir.



Şekil 64. C içindeki bir dağıtım listesine ileti konması

Şekil 65 sayfa 805 , COBOL ' da bir dağıtım listesine nasıl bir ileti yerleştirebileceğinin gösterir.



Şekil 65. COBOL ' de bir dağıtım listesine ileti konması

## MQPUT1komutunu kullanma

MQPUT1kullanıyorsanız, aşağıdaki noktaları göz önünde bulundurun:

1. *ResponseRecOffset* ve *ResponseRecPtr* alanlarının değerleri boş değerli ya da sıfır olmalıdır.
2. Gerekirse, Yanıt Kayıtları MQOD ' den adreslenmelidir.

### Put çağrılarının başarısız olduğu bazı durumlar

Bir MQOPEN ve bir MQPUT çağrısıyla arasındaki aralık sırasında bir komutun FORCE seçeneği kullanılarak bir kuyruğun bazı öznelikleri değiştirilirse, MQPUT çağrısı başarısız olur ve MQRC\_OBJECT\_CHANGED neden kodunu döndürür.

Kuyruk yöneticisi, nesne tanıtıcısını artık geçerli değil olarak işaretler. Bu durum, bir MQPUT1 çağrısı işlenirken ya da değişikliklerin kuyruk adının çözümlediği herhangi bir kuyruk üzerinde uygulanırsa, bu durum da oluşur. Bu şekilde, tanıtıcıyı etkileyen öznitelikler, MQOPEN' da MQOPER çağrısının tanımında listelenir. Aramanız MQRC\_OBJECT\_CHANGED neden kodunu döndürürse, kuyruğu kapatın, yeniden açın ve sonra bir ileti yeniden yerleştirmeyi deneyin.

İletileri (ya da kuyruk adının çözümlendiği herhangi bir kuyruğu) yerleştirmeye çalıştığınız bir kuyruk için bir işlem engellenirse, MQPUT ya da MQPUT1 çağrısı başarısız olur ve MQRC\_PUT\_INHIBITED neden kodunu döndürür. Aramayı daha sonra denerseniz, başka programlar kuyrukların özniteliklerini düzenli olarak değiştiriyorsa, iletiyi daha sonra başarılı bir şekilde gerçekleştirebilmeniz de, iletiyi başarıyla yerleştirebilirsiniz.

Furthermore, iletinizi yerleştirmeye çalıştığınız kuyruk dolu olursa, MQPUT ya da MQPUT1 çağrısı başarısız olur ve MQRC\_Q\_FULL değerini döndürür.

Bir dinamik kuyruk (geçici ya da kalıcı) silindiyse, MQPUT çağrıları önceden edinilmiş bir nesne tanıtıcısı kullanılarak başarısız olur ve MQRC\_Q\_DELETED neden kodunu döndürür. Bu durumda, nesne tanıtıcısını kapatmanız artık size herhangi bir faydası olmadığı için iyi bir uygulamadır.

Dağıtım listeleri durumunda, tek bir istekte birden çok tamamlanma kodu ve neden kodu ortaya çıkabilir. Bu işlem, yalnızca MQOUT ve MQPUT üzerindeki *CompCode* ve *Reason* çıkış alanları kullanılarak işlenemez.

Birden çok hedefe ileti koymak için dağıtım listeleri kullandığınızda, Yanıt Kayıtları her hedef için belirli *CompCode* ve *Reason* ' yi içerir. MQCC\_FAILED tamamlanma kodu alırsanız, herhangi bir hedef kuyruğa başarıyla herhangi bir ileti konmaz. Tamamlanma kodu MQCC\_UYARI ise, ileti bir ya da daha çok hedef kuyruktan başarıyla konabiliyor. Dönüş kodu MQRC\_MULTIPLE\_REASONS değerini alırsanız, neden kodları her hedef için aynı değildir. Bu nedenle, bir hataya neden olan kuyruğu ya da kuyrukları saptamanız için MQRR yapısının kullanılması önerilir; böylece, bir hataya neden olan kuyruklar ve nedenler de vardır.

## Kuyruktan İleti Alınması

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.



Bir kuyruktan iki şekilde ileti alabilirsiniz:

1. Bir iletiyi diğer programların artık görebilmesi için kuyruktan kaldırabilirsiniz.
2. Bir iletiyi kopyalayabilir ve özgün iletiyi kuyruğun üzerine bırakabilirsiniz. Bu, *göz atma* olarak bilinir. Bu iletiyi göz attığınızda, iletiyi kaldırabilirsiniz.

Her iki durumda da, MQGET çağrısını kullanıyorsunuz, ancak önce uygulamanızın kuyruk yöneticisine bağlı olması ve kuyruğu açmak için MQOPEN çağrısını kullanmanız (giriş, göz atma ya da her ikisi için) kullanmanız gerekir. Bu işlemler [“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#) ve [“Nesnelerin açılması ve kapatılması” sayfa 781](#) de anlatılır.

Kuyruğu açtığınızda, aynı kuyruktaki iletilere göz atmak ya da iletileri kaldırmak için MQGET çağrısını sık olarak kullanabilirsiniz. Kuyruktan istediğiniz tüm iletileri almayı bitirdiğinizde, MQCLOSE ' yi çağırın.

Bir kuyruktan ileti alma hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- [“MQGET çağrısını kullanarak kuyruktan ileti alma” sayfa 807](#)
- [“İletilerin kuyruktan alınma sırası” sayfa 811](#)
- [“Belirli bir iletiyi alma” sayfa 822](#)
- [“Kalıcı olmayan iletilerin performansını artırma” sayfa 823](#)
-  [“Dizin tipi” sayfa 827](#)
- [“4 MB ' den büyük iletilerin işlenmesi” sayfa 828](#)
- [“İleti bekleniyor” sayfa 833](#)
-  [“Sinyalizasyon” sayfa 834](#)
- [“Geri alma işlemi atlanıyor” sayfa 835](#)

- [“Uygulama verileri dönüştürme” sayfa 838](#)
- [“Kuyruklardaki İletilere Göz Atma” sayfa 839](#)
- [“MQGET çağrısının başarısız olduğu bazı durumlar” sayfa 844](#)

### **İlgili kavramlar**

[“Message Queue Interface-Genel Bakış” sayfa 760](#)

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabılır alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OSüzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

[“IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

### ***MQGET çağrısını kullanarak kuyruktan ileti alma***

MQGET çağrısı, açık yerel kuyruktan bir ileti alır. Başka bir sistemdeki kuyruktan ileti alamıyor.

MQGET çağrısına giriş olarak, aşağıdaki bilgileri sağlamanız gerekir:

- Bağlantı tanıtıcısı.
- Kuyruk tanıtıcısı.
- Kuyruktan almak istediğiniz iletinin açıklaması. Bu, bir ileti tanımlayıcısı (MQMD) yapısıdır.
- İleti Alma Seçenekleri (MQGMO) yapısındaki bilgileri denetleyin.
- İletiyi tutmak için atadığınız arabelleğin büyüklüğü (MQlong).
- İletinin konulması için kullanılan depolama alanının adresi.

MQGET ' tan çıkışı:


- Neden kodu
- Tamamlanma kodu
- Çağrı başarıyla tamamlanırsa, belirttiğiniz arabellek alanındaki ileti.
- Seçenek yapınız, iletinin alındığı kuyruğun adını göstermek için değiştirildi
- Alınan iletiyi açıklamak için değiştirilen alanların içeriğiyle ileti tanımlayıcı yapınız
- İletinin uzunluğu (MQLONG)

[MQGET](#) içinde MQGET çağrısının bir açıklaması var.

Aşağıdaki kısımlarda, MQGET çağrısına giriş olarak sağlamanız gereken bilgiler açıklanmaktadır.

- [“Bağlantı tanıtıcılarının belirtilmesi” sayfa 808](#)
- [“MQMD yapısını ve MQGET çağrısını kullanarak iletileri tanımlama” sayfa 808](#)
- [“MQGMO yapısını kullanarak MQGET seçeneklerini belirtme” sayfa 808](#)
- [“Arabellek alanının büyüklüğünün belirtilmesi” sayfa 810](#)

## Bağlantı tanıtıcılarının belirtilmesi

 z/OS üzerinde CICS uygulamaları için, MQHC\_DEF\_HCONN değişmezini (sıfır değerine sahip) belirtebilir ya da MQCONN ya da MQCONNX çağrısının döndürdüğü bağlantı tanıtıcısını kullanabilirsiniz. Diğer uygulamalar için, her zaman MQCONN ya da MQCONNX çağrısının döndürdüğü bağlantı tanıtıcısını kullanın.

MQASAçık 'ı çağırdığınızda döndürülen kuyruk tanıtıcısını (*Hobj*) kullanın.

## MQMD yapısını ve MQGET çağrısını kullanarak iletileri tanımlama

Bir kuyruktan almak istediğiniz iletiyi tanımlamak için, ileti tanımlayıcı yapısını (MQMD) kullanın.

Bu, MQGET çağrısına ilişkin bir giriş/çıkış parametresidir. MQMD 'nin [“IBM MQ ileti” sayfa 16'](#) de tanımladığı ve MQMD içinde yapının kendisiyle ilgili bir açıklaması olan ileti özelliklerine giriş var.

Kuyruktan hangi iletiyi almak istediğinizi biliyorsanız, bkz. [“Belirli bir iletiyi alma” sayfa 822](#).

Belirli bir ileti belirtmezseniz, MQGET kuyruқта *ilk* iletisini alır. [“İletilerin kuyruktan alınma sırası” sayfa 811](#) , bir iletinin önceliğinin, kuyruğun **MsgDeliverySequence** özniteliğinin ve MQGMO\_LOGICAL\_ORDER seçeneğinin, kuyruklardaki iletilerin sırasını nasıl belirlediğini açıklar.

**Not:** MQGET ' yi bir kereden fazla kullanmak istiyorsanız (örneğin, kuyruқта bulunan iletileri adımlamak için), her çağrıdan sonra bu yapının *MsgId* ve *CorrelId* alanlarını boş değer olarak ayarlamanız gerekir. Bu işlem, alınan iletinin tanıtıcılarının bu alanlarını temizler.

Ancak, iletilerinizi gruplamak istiyorsanız, *GroupId* ' un aynı gruptaki iletiler için aynı olması gerekir; böylece, arama, tüm grubu oluşturan bir iletinin önceki iletiyle aynı tanıtıcılara sahip olmasını sağlar.

## MQGMO yapısını kullanarak MQGET seçeneklerini belirtme

MQGMO yapısı, MQGET çağrısına seçenekleri geçirmek için kullanılan bir giriş/çıkış değişkenidir. Aşağıdaki kısımlar, bu yapının bazı alanlarını tamamlamanıza yardımcı olur.

[MQGMO](#) içinde MQGMO yapısının bir açıklaması vardır.

### **StrucId**

*StrucId* , bir get-message options yapısı olarak yapıyı tanımlamak için kullanılan 4 karakterlik bir alandır. Her zaman MQGMO\_STRUC\_ID değerini belirtin.

### **Version**

*Version* yapının sürüm numarasını açıklar. Varsayılan değer MQGMO\_VERSION\_1 ' dir. Sürüm 2 alanlarını kullanmak ya da iletileri mantıksal sırada almak istiyorsanız, MQGMO\_VERSION\_2 değerini belirtin. Sürüm 3 alanlarını kullanmak ya da iletileri mantıksal sırayla almak istiyorsanız, MQGMO\_VERSION\_3 değerini belirtin. MQGMO\_CURRENT\_VERSION uygulamanızı en son düzeyi kullanacak şekilde ayarlar.

### **Options**

Kodunuz içinde, seçenekleri istediğiniz sırayla seçebilirsiniz; her seçenek *Options* alanında bir bit tarafından temsil edilir.

*Options* alan denetimleri:

- MQGET çağrısının, tamamlanmadan önce kuyruğa gelmesi için bekleyeceği (bkz. [“İleti bekleniyor” sayfa 833](#) )
- Alma işleminin bir iş birimine dahil edilip etmeyeceğini belirleyin.



- Kalıcı olmayan bir iletinin, eşitleme noktası dışında alınıp alınmadığı, hızlı ileti sistemine izin verilip verilmeyeceği
- **z/OS** IBM MQ for z/OS üzerinde, alınan iletinin geri alma işlemi atlanıyor olarak işaretlenip işaretlenmediği (bkz. “Geri alma işlemi atlanıyor” sayfa 835 )
- İletinin kuyruktan kaldırılıp kaldırılmadığını ya da yalnızca göz atmadığını
- Göz atma ya da diğer seçim ölçütlerine göre bir iletinin seçilip seçilmeyeceği
- İleti arabelleğinizden daha uzun olsa bile çağrılarının başarılı olup olmayacağını
- **z/OS** On IBM MQ for z/OS, whether to allow the call to complete. Bu seçenek, bir ileti geldiğinde bildirim almak istediğinizi belirtmek için bir sinyal de ayarlar.
- Kuyruk yöneticisi susturulmuş durumda olduğunda çağrılarının başarısız olup olmadığını
- **z/OS** IBM MQ for z/OS' ta, bağlantı susturulmuş durumdaysa arama başarısız olur.
- Uygulama iletisi veri dönüştürmesinin gerekli olup olmadığı (bkz. “Uygulama verileri dönüştürme” sayfa 838 )
- İletilerin ve bölümlerin bir kuyruktan alındığı sıralama düzeni **z/OS** ( IBM MQ for z/OS dışında)
- Tam olarak, mantıksal iletiler yalnızca yeniden alınabilir **z/OS** ( IBM MQ for z/OS dışında)
- Gruptaki iletilerin yalnızca, gruptaki tüm iletiler kullanılabilir olduğu durumlarda alınıp alınmayacağı
- Whether segments in a logical message can be retrieved only when Tüm segments in the logical message are available **z/OS** ( IBM MQ for z/OS dışında)

*Options* alanını varsayılan değere (MQGMO\_NO\_WAIT) bırakırsanız, MQGET çağrısı şu şekilde çalışır:

- Kuyruktaki seçim ölçütlerinizle eşleşen bir ileti yoksa, çağrı ileti gelmesini beklemez, ancak hemen tamamlanır. **z/OS** Ayrıca, IBM MQ for z/OS içinde arama, böyle bir ileti geldiğinde bildirim isteyen bir sinyal ayarlamaz.
- Arama, eşitleme noktalarıyla çalışma şeklinin altyapıya göre belirlendiği şekilde belirlenir:

Platform	Eşitleme noktası denetimi altında
IBM i	Hayır
UNIX and Linux sistemleri	Hayır
<b>z/OS</b> <b>z/OS</b> z/OS	Evet
Windows sistemleri	Hayır

- **z/OS** IBM MQ for z/OS' da, alınan ileti atlanıyor olarak işaretlenmedi.
- Seçilen ileti kuyruktan kaldırılır (göz atılmaz).
- Herhangi bir uygulama iletisi veri dönüştürme işlemi gerekmiyor.
- İleti arabelleğinizden uzunsu arama başarısız olur.

### **WaitInterval**

*WaitInterval* alanı, MQGMO\_WALEM seçeneğini kullandığınızda, MQGET çağrısının bir iletinin kuyruğa varması için bekleyeceği sürenin üst sınırını (milisaniye olarak) belirtir. *WaitInterval* ' ta belirtilen süre içinde hiçbir ileti gelmezse, arama tamamlanır ve kuyruktaki seçim ölçütlerinizle eşleşen bir ileti olmadığını gösteren bir neden kodu döndürür.

**z/OS** IBM MQ for z/OS' ta, MQGMO\_SET\_SIGNAL seçeneğini kullanıyorsanız, *WaitInterval* alanı, işaretin ayarının ne zaman ayarlandığını belirtir.

Bu seçenekler hakkında daha fazla bilgi için bkz. [“İleti bekleniyor” sayfa 833](#) **z/OS** ve [“Sinyalizasyon” sayfa 834](#).

## **Signal1**

**Signal1 yalnızca **z/OS** IBM MQ for z/OS üzerinde desteklenir.**

Uygun bir ileti geldiğinde uygulamanızın bilgilendirileceğini istemek için MQGMO\_SET\_SIGNAL seçeneğini kullanırsanız, *Signal1* alanında sinyal tipini belirtmiş olur. Diğer tüm platformlarda IBM MQ içinde, *Signal1* alanı ayrılır ve değeri anlamlı değildir.

**z/OS** Daha fazla bilgi için bkz [“Sinyalizasyon” sayfa 834](#).

## **Signal2**

*Signal2* alanı tüm platformlarda ayrılır ve değeri önemli değildir.

**z/OS** Daha fazla bilgi için [“Sinyalizasyon” sayfa 834](#) başlıklı konuya bakın.

## **ResolvedQName**

*ResolvedQName*, kuyruk yöneticisinin iletinin alındığı kuyruğun adını (herhangi bir diğer ad çözüldükten sonra) döndürdüğü bir çıkış alanıdır.

## **MatchOptions**

*MatchOptions*, MQGET için seçim ölçütlerini denetler.

## **GroupStatus**

*GroupStatus*, aldığınız iletinin bir grup içinde olup olmadığını belirtir.

## **SegmentStatus**

*SegmentStatus*, aldığınız ögenin mantıksal bir iletinin bir parçası olup olmadığını belirtir.

## **Segmentation**

*Segmentation*, alınan ileti için bölümlenmeye izin verilip verilmediğini belirtir.

## **MsgToken**

*MsgToken*, bir iletiyi benzersiz şekilde tanımlar.

## **ReturnedLength**

*ReturnedLength*, kuyruk yöneticisinin döndürdüğü ileti verilerinin uzunluğunu (bayt olarak) döndürdüğü bir çıkış alanıdır.

## **MsgHandle**

Kuyruktan alınan iletinin özellikleri ile doldurulacak bir iletinin tanıtıcısı. Tanıtıcı daha önce bir MQCRTMH çağrısı tarafından yaratılmıştır. Bir ileti alınmadan önce, bu tanıtıcı ile ilişkilendirilmiş olan tüm özellikler temizlenir.

## **Arabellek alanının büyüklüğünün belirtilmesi**

MQGET çağrısının **BufferLength** değiştirilmesinde, aladığınız ileti verilerini tutmak için arabellek alanının büyüklüğünü belirtin. Bunun üç şekilde ne kadar büyük olması gerektiğine karar verirsiniz:

1. Bu programdan beklenecek iletilerin uzunluğunu zaten bilebilirsiniz. Bu durumda, bu boyutta bir arabellek belirtin.

Ancak, ileti arabellek için çok büyük olsa da MQGET çağrısının tamamlanmasını istiyorsanız, MQGMO yapısındaki MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneğini kullanabilirsiniz. Bu durumda:

- Arabellek, tutulabildiği kadar iletiyi doldurur.
- Çağrı, bir uyarı tamamlanma kodu döndürür
- İleti kuyruktan kaldırılır (iletinin geri kalan kısmı atılır) ya da göz at imleci ilerletilir (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 boyut tahmin edin (ya da sıfır byte büyüklüğünde bir boyut belirtin) ve *yapma* , MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneğini kullanın. MQGET çağrısının başarısız olması (örneğin, arabellek çok küçük olduğu için), iletinin uzunluğu çağrıya ilişkin **DataLength** parametresine döndürülür. (Arabellek, iletinin tutulabildiği kadarını doldurur, ancak arama işlemi tamamlanmamaktadır.) Bu iletinin *MsgId* değerini saklayın, daha sonra, doğru büyüklükte bir arabellek alanı belirterek, MQGET çağrısını yineleyin ve ilk çağrıdan not ettiğiniz *MsgId* ' i yineleyin.

Programınız başka programlar tarafından da sunulmakta olan bir kuyruğa sunuyorsa, diğer programlardan biri, programınız başka bir MQGET çağrısını yayınlamadan önce istediğiniz iletiyi kaldırabilir. Programınız, artık var olmayan bir iletiyi arayarak zaman kaybedebilir. Bunu önlemek için, istediğiniz iletiyi buluncaya kadar önce kuyruğa göz atın, sıfır *BufferLength* belirterek ve MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneğini kullanın. Bu, göz atma imlecini istediğiniz iletinin altına konumlayın. Bundan sonra MQGET çağrılarak, MQGMO\_MSG\_UNDER\_CURSOR seçeneğini belirterek iletiyi alabilirsiniz. Göz atma ve kaldırma çağrılarınız arasındaki iletiyi başka bir program kaldırırsa, göz atma imleciniz altında ileti olmadığı için, ikinci MQGET işlemi hemen başarısız olur (tüm kuyruğun aranması olmadan).

3. *MaxMsgLength queue* özneliği, o kuyruk için kabul edilen ileti uzunluğu üst sınırını belirler; *MaxMsgLength kuyruk yöneticisi* özneliği, o kuyruk yöneticisi için kabul edilen ileti uzunluğu üst sınırını belirler. Beklenecek iletinin uzunluğunu bilmiyorsanız, **MaxMsgLength** özneliğini sorgulayabilir (MQINQ çağrısını kullanarak), bu büyüklükte bir arabellek belirtebilirsiniz.

Başarımı düşürmemek için arabellek büyüklüğünü, gerçek ileti büyüklüğünün olabildiğince yakın olması için deneyin.

**MaxMsgLength** özneliğe ilişkin daha fazla bilgi için bkz. [“İleti uzunluğu üst sınırını artırma” sayfa 828.](#)

## ***İletilerin kuyruktan alınma sırası***

Kuyruktan ileti almanıza yardımcı olacak sırayı denetleyebilirsiniz. Bu bölüm seçeneklere bakar.

### *Öncelik*

Bir program, iletiyi bir kuyruğa koyduğunda bir ileti için öncelik atayabilir (bkz. [“İleti öncelikleri” sayfa 24](#) ). Eşit önceliğe sahip iletiler, işlendikleri sıraya göre değil, geliş sırasına göre kuyruğa saklanır.


Kuyruk yöneticisi, kuyrukları, sıkı FIFO (ilk giren, ilk çıkış) sırasıyla ya da öncelik sırası içinde FIFO ' da tutar. Bu, kuyruğun **MsgDeliverySequence** özneliğinin ayarına bağlıdır. Bir ileti kuyruğa ulaştığında, aynı önceliğe sahip son iletinin hemen ardından eklenir.

Programlar bir kuyruktan ilk iletiyi alabilir ya da bir kuyruktan belirli bir iletiyi alabilirler, bu iletilerin önceliğini dikkate almayabilir. Örneğin, bir program, yanıtı daha önce gönderdiği belirli bir iletiye işlemek isteyebilirler. Daha fazla bilgi için [“Belirli bir iletiyi alma” sayfa 822](#) başlıklı konuya bakın.

Bir uygulama kuyruқта bir ileti dizisi koyarsa, başka bir uygulama bu iletileri, yerleştirdikleri sırayla alabilir ve bu iletileri aşağıdaki sırayla alabilir:

- İletilerin hepsinin önceliği aynı.
- Mesajların hepsi aynı iş birimi içinde, ya da hepsi bir iş biriminin dışına konulmak üzere.
- Kuyruk, uygulama koymak için yereldir

Bu koşullar karşılanmazsa ve uygulamalar belirli bir sırayla alınmakta olan iletilere bağlı olduğunda, uygulamalar ileti verilerinde sıralama bilgilerini içermeli ya da bir iletinin sonraki gönderilmeden önce bir iletinin alınmasını kabul etmek için bir araç oluşturmalıdır.

 On IBM MQ for z/OS, you can use the queue attribute, *IndexType*, to increase the speed of MQGET operations on the queue. Daha fazla bilgi için [“Dizin tipi” sayfa 827](#) başlıklı konuya bakın.

### *Mantıksal ve fiziksel sıralama*

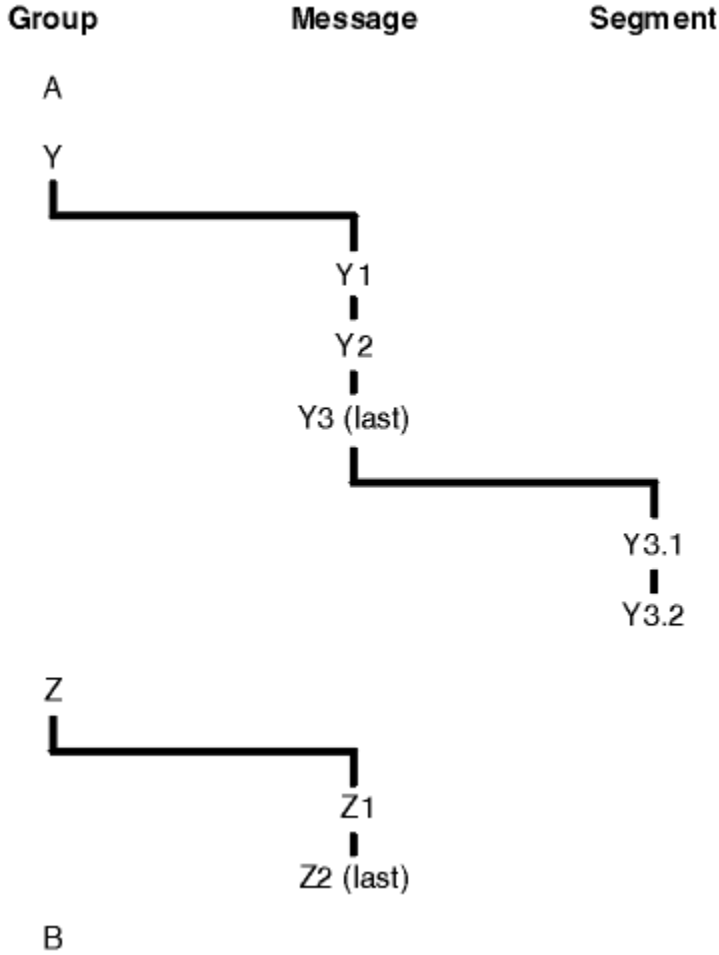
Kuyruklardaki iletiler *fiziksel* ya da *mantıksal* düzende (her bir öncelik düzeyi içinde) gerçekleştirilebilir.

Fiziksel sıralama, iletilerin kuyruқта vardiđı sıradır. Mantıksal sipariř, bir gruptaki tüm ileti ve kesimlerin, gruba ait ilk öđenin fiziksel konumu tarafından belirlenen konumda, birbirinin yanında mantıksal sırada yer aldıklarında yer alan bir sıradır.

Grupların, iletilerin ve bölümlerin bir açıklaması için bkz. “İleti grupları” sayfa 41. Bu fiziksel ve mantıksal sipariřler farklı olabilir:

- Gruplar, farklı uygulamalardan benzer zamanlarda bir hedefe varabilir ve bu nedenle farklı fiziksel düzeni kaybedebilir.
- Tek bir grup içinde bile iletiler, gruptaki iletilerin bir kısmının yeniden yönlendirmesi ya da gecikmesi nedeniyle sıradan çıkabiliyor.

Örneđin, mantıksal sipariř Şekil Şekil 66 sayfa 812 gibi görünebilir:

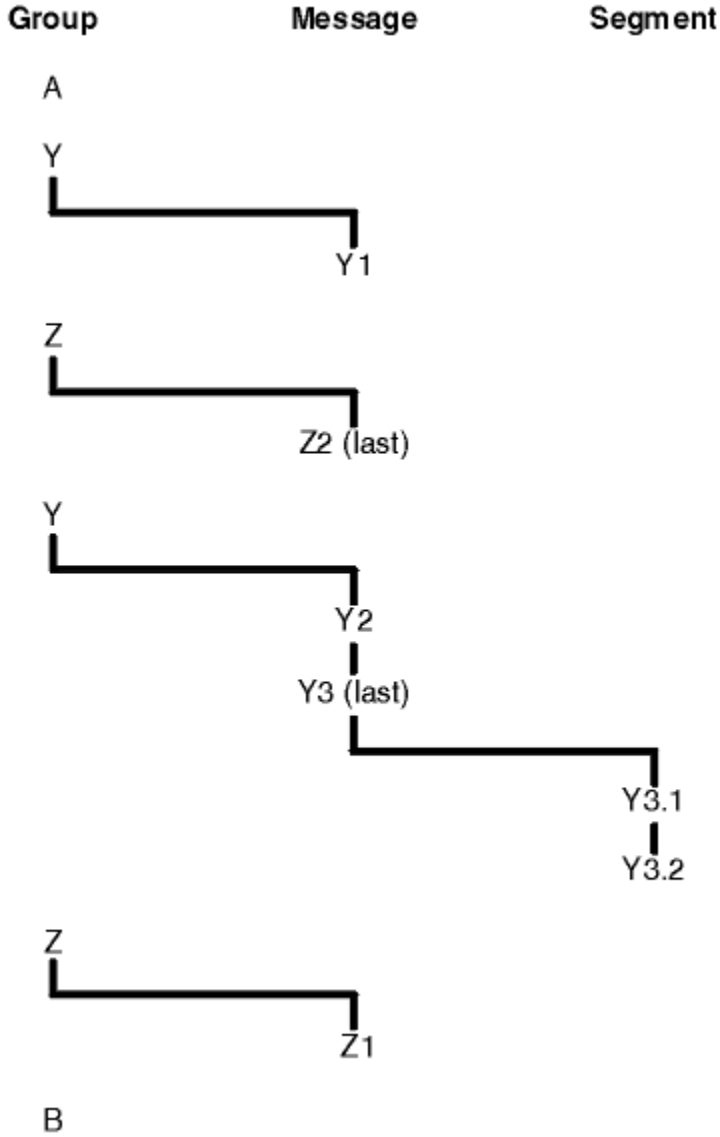


Şekil 66. Kuyruđun mantıksal sırası

Bu iletiler, bir kuyruқта ařađıdaki mantıksal sırada gerçekteřir:

1. İleti A (bir grupta deđil)
2. Grup Y ' nin mantıksal iletisi 1
3. Grup Y ' nin mantıksal iletisi 2
4. Grup Y ' nin (son) mantıksal iletisi 3 ' ünün (son) 1. bölümü
5. (Son) grup Y ' nin 2 numaralı (son) mantıksal iletisi 3
6. Grup Z ' nin mantıksal iletisi 1
7. (Son) grup Z ' nin mantıksal iletisi 2
8. İleti B (bir grupta deđil)

Ancak fiziksel düzen tamamen farklı olabilir. Her bir grup içindeki *ilk* ögenin fiziksel konumu, tüm grubun mantıksal konumunu belirler. Örneğin, Y ve Z grupları benzer zamanlarda geldiyse ve Z grup Z ' nin iletisi aynı grubun 1 numaralı iletisini aşıyorsa, fiziksel sipariş Şekil Şekil 67 sayfa 813 gibi görünür:



Şekil 67. Kuyruklardaki fiziksel sıralama

Bu iletiler, kuyrukta aşağıdaki fiziksel sırada gerçekleşir:

1. İleti A (bir grupta değil)
2. Grup Y ' nin mantıksal iletisi 1
3. Z grubu Z mantıksal iletisi 2
4. Grup Y ' nin mantıksal iletisi 2
5. Grup Y ' nin (son) mantıksal iletisi 3 'ün (son) 1. bölümü
6. (Son) grup Y ' nin 2 numaralı (son) mantıksal iletisi 3
7. Grup Z ' nin mantıksal iletisi 1
8. İleti B (bir grupta değil)

**Not:** IBM MQ for z/OS' ta, kuyruğun GROUPID tarafından dizinlenmesi durumunda, kuyruklardaki iletilerin fiziksel sırası garanti edilmez.

İletileri alırken, iletileri fiziksel sıralama yerine mantıksal sırada almak için MQGMO\_LOGICAL\_ORDER belirtebilirsiniz.

MQGMO\_BROWSE\_FIRST ve MQGMO\_LOGICAL\_ORDER ile MQGET işlemi yayınlarsa, MQGMO\_BROWSE\_NEXT ile sonraki MQGET çağruları da MQGMO\_LOGICAL\_ORDER değerini de belirtmelidir. Bunun tersine, MQGMO\_Browse\_first ile MQGET işlemi MQGMO\_LOGICAL\_ORDER belirtmiyorsa, MQGMO\_BROWSE\_NEXT ile birlikte izleyen MQGES ' ler de geçerli değildir.

Kuyruk yöneticisinin MQGET çağruları için sakladığı grup ve kesim bilgileri, kuyruklardaki iletilere göz atmanın grup ve bölüm bilgilerinden ayrıdır ve kuyruk yöneticisinin iletileri kuyruktan kaldırmak için MQGET çağruları için sakladığı bilgileri içerir. MQGMO\_BROWSE\_FIRST belirttiğinizde, kuyruk yöneticisi, göz atma için grup ve bölüm bilgilerini yoksayar ve yürürlükteki grup ve yürürlükteki mantıksal ileti yok gibi kuyrukları tarar.

**Not:** Do not use an MQGET call to browse *sonun ötesinde* of a message group (or logical message not in a group) without specifying MQGMO\_LOGICAL\_ORDER. For example, if the last message in the group *emsaller* the first message in the group on the queue, using MQGMO\_BROWSE\_NEXT to browse beyond the end of the group, specifying MQMO\_MATCH\_MSG\_SEQ\_NUMBER with *MsgSeqNumber* set to 1 (to find the first message of the next group) returns again the first message in the group already browsed. Bu durum hemen olabilir ya da daha sonra (araya giren gruplar varsa) MQGET çağruları sayısı hemen olabilir.

Göz atma işlemi için *iki kez* kuyruğunu açarak sonsuz döngü olasılığını önlein:

- Her gruptaki ilk iletiye göz atmak için ilk tanıtıcıyı kullanın.
- Yalnızca belirli bir grup içindeki 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 sonuna kadar MQGMO\_BROWSE\_NEXT göz atma olanağını kullanmayın.

Bu konuda ek bilgi için bkz. [MQGET](#), [MQMDve MQI seçeneklerinin geçerliliğini denetlemek için kurallar](#).

Çoğu uygulama için, göz atılırken mantıksal ya da fiziksel sıralamayı seçebilirsiniz. Ancak, bu kipler arasında geçiş yapmak istiyorsanız, ilk olarak MQGMO\_LOGICAL\_ORDER ile bir göz atma işlemi ilk kez yayınlarken, mantıksal sıra içindeki konumunuz oluşturulur.

Gruptaki ilk öge şu anda mevcut değilse, içinde bulunmanız gereken grup, mantıksal sıranın bir parçası olarak kabul edilmez.

Göz at imleci bir grup içindeyse, ilk ileti kaldırılrsa bile, aynı grup içinde devam edebilir. Başlangıçta, ilk ögenin mevcut olmadığı MQGMO\_LOGICAL\_ORDER kullanarak bir gruba hiçbir zaman geçemeyebilirsiniz.

### **MQPMO\_LOGICAL\_ORDER**

MQPMO seçeneği, kuyruk yöneticisine uygulamanın, iletileri gruplar ve mantıksal iletiler segmentlerine nasıl yerleştirdiğini bildirir. Yalnızca MQPUT çağrısında belirtilebilir; MQPUT1 çağrısında geçerli değildir.

MQPMO\_LOGICAL\_ORDER belirtilirse, uygulamanın aşağıdaki öğeler için ardışık MQPUT çağruları kullandığını gösterir:

1. Her mantıksal iletideki kesimleri, 0 'dan başlayarak, aralıksız olarak, kesim görelî konumunun artma sırasına göre koyun.
2. Bölümleri sonraki mantıksal iletiye yerleştirmeden önce tüm bölümleri tek bir mantıksal iletiye koyun.
3. Her ileti grubundaki mantıksal iletileri, 1 'den başlayarak, aralıksız artan ileti sıra numarası sırasına göre 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 gruplar ve mantıksal iletiler bölümlerine nasıl yerleştirdiğini anlattığından, kuyruk yöneticisi bu bilgileri saklayıp güncellediğinden, uygulamanın her bir MQPUT çağrısıyla ilgili grup ve bölüm bilgilerini korumak ve güncellemek zorunda kalmayacağından emin olun. Özellikle, kuyruk yöneticisi bu alanları uygun değerlere ayarlandığından, uygulamanın MQMD

' de *GroupId*, *MsgSeqNumber* ve *Offset* alanlarını ayarlamaya gerek olmadığı anlamına gelir. Uygulamanın yalnızca MQMD ' deki *MsgFlags* alanını ayarlaması gerekir; bu alan, iletilerin ne zaman gruplara ait olduğunu ya da mantıksal iletilerin bölümleri olduğunu göstermek ve bir gruptaki son iletiyi ya da mantıksal iletinin son bölümünü belirtmek için.

After a message group or logical message has been started, subsequent MQPUT calls must specify the appropriate MQMF\_\* flags in *MsgFlags* in MQMD. Uygulama, sonlandırılmamış bir ileti grubu olduğunda ya da sonlandırılmamış bir mantıksal ileti olduğunda, bir ileti grubu olmayan bir iletiyi yerleştirmeye çalışırsa, uygun olduğu şekilde, çağrı neden kodu MQRC\_INCOMPLE\_GROUP ya da MQRC\_INCOMPLE\_MSG neden koduyla başarısız olur. Ancak, kuyruk yöneticisi yürürlükteki ileti grubuyla ya da yürürlükteki mantıksal iletiyle ilgili bilgileri saklar ve MQMF\_LAST\_MSG\_IN\_GROUP ya da MQMF\_LAST\_SEGMENT belirtimini uygun olarak belirterek, MQPUT çağrısını, grupta olmayan ya da bir kesim olmayan iletiyi koymak için yeniden vermeden önce, uygulama bu bilgileri bir ileti (uygulama ileti verisi olmadan) göndererek sonlandırabilir.

Şekil 67 sayfa 813 , geçerli olan seçenek ve işaretlerin birleşimlerini ve kuyruk yöneticisinin her bir 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 değeri anlamına gelir:

#### OTURUM KAPAT

Çağrıda MQPMO\_LOGICAL\_ORDER seçeneğinin belirtilip belirtilmediğini belirleyin.

#### MIG

Çağrıda MQMF\_MSG\_IN\_GROUP ya da MQMF\_LAST\_MSG\_IN\_GROUP seçeneğinin belirtilip belirtilmediğini.

#### GÇ

Çağrıda MQMF\_SEGMENT ya da MQMF\_LAST\_SEGMENT seçeneğinin belirtilip belirtilmediğini belirleyin.

#### SEEG Tamam

Çağrıda MQMF\_SEGMENTATION\_ALLOWLI seçeneğinin belirtilip belirtilmediğini.

#### Cur grp

Çağrıdan önce geçerli bir ileti grubunun var olup olmadığını.

#### Cur günlük ileti

Çağrıdan önce geçerli bir mantıksal iletinin var olup olmadığını.

#### Diğer kolonlar

Kuyruk yöneticisinin kullandığı değerleri gösterir. Önceki ileti, kuyruk tanıtıcısı için önceki iletide alan için kullanılan değeri gösterir.

Çizelge 120. Mantıksal iletilerin gruplarındaki ve kısımlarındaki iletilere ilişkin MQPUT seçenekleri								
Belirttiğiniz seçenekler	Belirttiğiniz seçenekler	Belirttiğiniz seçenekler	Belirttiğiniz seçenekler	Aramadan önce grup ve günlük k-msg durumu	Aramadan önce grup ve günlük k-msg durumu	Kuyruk yöneticisinin kullandığı değerler	Kuyruk yöneticisinin kullandığı değerler	Kuyruk yöneticisinin kullandığı değerler
OTURUM KAPAT	MIG	GÇ	SEEG Tamam	Cur grp	Cur günlük ileti	<b>GroupId</b>	<b>MsgSeqNumber</b>	<b>Offset</b>
Evet	Hayır	Hayır	Hayır	Hayır	Hayır	MQGI_NONE	1	0
Evet	Hayır	Hayır	Evet	Hayır	Hayır	Yeni grup tanıtıcısı	1	0

Çizelge 120. Mantıksal iletilerin gruplarındaki ve kısımlarındaki iletilere ilişkin MQPUT seçenekleri (devamı var)

Belirttiğiniz seçenekler	Belirttiğiniz seçenekler	Belirttiğiniz seçenekler	Belirttiğiniz seçenekler	Aramadan önce grup ve günlük k-msg durumu	Aramadan önce grup ve günlük k-msg durumu	Kuyruk yöneticisinin kullandığı değerler	Kuyruk yöneticisinin kullandığı değerler	Kuyruk yöneticisinin kullandığı değerler
Evet	Hayır	Evet	Herhangi biri	Hayır	Hayır	Yeni grup tanıtıcısı	1	0
Evet	Hayır	Evet	Herhangi biri	Hayır	Evet	Önceki grup tanıtıcısı	1	Önceki görel konum + önceki bölüm 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 görel konum + önceki bölüm uzunluğu
Hayır	Hayır	Hayır	Hayır	Herhangi biri	Herhangi biri	MQGI_NONE	1	0
Hayır	Hayır	Hayır	Evet	Herhangi biri	Herhangi biri	MQGI_NONE ise, alanda başka değer varsa yeni grup tanıtıcısı	1	0
Hayır	Hayır	Evet	Herhangi biri	Herhangi biri	Herhangi biri	MQGI_NONE ise, alanda başka değer varsa yeni grup tanıtıcısı	1	Alandaki değer
Hayır	Evet	Hayır	Herhangi biri	Herhangi biri	Herhangi biri	MQGI_NONE ise, alanda başka değer varsa yeni grup tanıtıcısı	Alandaki değer	0
Hayır	Evet	Evet	Herhangi biri	Herhangi biri	Herhangi biri	MQGI_NONE ise, alanda başka değer varsa yeni grup tanıtıcısı	Alandaki değer	Alandaki değer

**Not:**

- MQPMO\_LOGICAL\_ORDER, MQPUT1 çağrısında geçerli değil.
- *MsgId* alanı için, MQPMO\_NEW\_MSG\_ID ya da MQMI\_NONE belirtilirse, kuyruk yöneticisi yeni bir ileti tanıtıcısı oluşturur ve bu değeri alanda başka bir değer kullanır.
- *CorrelId* alanı için, MQPMO\_NEW\_CORREL\_ID belirtildiyse, kuyruk yöneticisi yeni bir ilinti tanıtıcısı oluşturur ve diğer bir biçimde alanda değer kullanır.

MQPMO\_LOGICAL\_ORDER belirttiğinizde, kuyruk yöneticisi bir gruptaki tüm iletilerin ve mantıksal bir iletide belirtilen tüm iletilerin, MQMD 'deki *Persistence* alanında aynı değere sahip olmasını



gerektirir; yani, tüm bunların kalıcı olması ya da tümünün kalıcı olmaması gerekir. Bu koşul karşılanmazsa, MQPUT çağrısına MQRC\_INCONSISTENT\_PERSISTENCE neden koduyla başarısız olur.

MQPMO\_LOGICAL\_ORDER seçeneği, iş birimlerini aşağıdaki gibi etkiler:

- Bir gruptaki ilk fiziksel ileti ya da mantıksal ileti bir iş birimi içine konursa, aynı kuyruk tanıtıcısı kullanılsa, gruptaki ya da mantıksal iletteki diğer tüm fiziksel iletiler bir iş birimi içine konmalıdır. Ancak, iki ya da daha fazla sayıda fiziksel iletilerden oluşan bir ileti grubunun ya da mantıksal iletinin kuyruk tanıtıcısı için birbirini izleyen iki ya da daha çok iş birimi arasında bölünmesine olanak sağlayan bir ileti grubuna ya da mantıksal iletiye izin verilmesine gerek yoktur.
- Bir gruptaki ilk fiziksel ileti ya da mantıksal ileti bir iş birimi içine konmazsa, aynı kuyruk tanıtıcısı kullanılsa, gruptaki diğer fiziksel iletilerden hiçbiri ya da mantıksal ileti bir iş birimi içine yerleştirilebilir.

Bu koşullar karşılanmazsa, MQPUT çağrısına neden kodu MQRC\_INCONSISTENT\_UOW neden kodu girilir.

MQPMO\_LOGICAL\_ORDER belirtildiğinde, MQPUT çağrısında belirtilen MQMD, MQMD\_VERSION\_2değerinden küçük olmamalıdır. Bu koşul karşılanmazsa, çağrı neden kodu MQRC\_WRONG\_MD\_VERSION ile başarısız olur.

MQPMO\_LOGICAL\_ORDER belirtilmezse, mantıksal iletilerin gruplarındaki ve kısımlarındaki iletiler herhangi bir sıraya yerleştirilebilir ve tam ileti gruplarının ya da tam mantıksal iletilerin tamamlanmaması gerekmez. It is the responsibility of the application to ensure that the *GroupId*, *MsgSeqNumber*, *Offset*, and *MsgFlags* fields have appropriate values.

Bir sistem hatası ortaya çıktıktan sonra, ortadaki 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 daha sonra MQPMO\_SYNCPOINT ya da MQPMO\_NO\_SYNCPOINT ayarlı MQPUT çağrısını zorunlu olarak ayarlayabilir, ancak MQPMO\_LOGICAL\_ORDER belirtilmeden. Bu çağrı başarılı olursa, kuyruk yöneticisi grup ve bölüm bilgilerini saklar ve izleyen MQPUT çağrıları bu kuyruk tanıtıcısını kullanarak MQPMO\_LOGICAL\_ORDER değerini normal olarak belirtebilir.

Kuyruk yöneticisinin MQPUT çağrısı için sakladığı grup ve kesim bilgileri, MQGET çağrısına ilişkin sakladığı grup ve bölüm bilgilerinden ayrıdır.

Herhangi bir kuyruk tanıtıcısı için uygulama, MQPMO\_LOGICAL\_ORDER ile belirtilen MQPUT çağrılarını karıştırabilir; bu çağrılar, aşağıdaki noktaları dikkate almaz:

- MQPMO\_LOGICAL\_ORDER belirtilmediyse, her başarılı MQPUT çağrısı kuyruk yöneticisinin, kuyruk tanıtıcısı için kuyruk yöneticisi tarafından tutulan var olan grup ve kesim bilgilerinin yerine, kuyruk tanıtıcısı için grup ve kesim bilgilerini uygulama tarafından belirlenen değerlere ayarlamasına neden olur.
- MQPMO\_LOGICAL\_ORDER belirtilmediyse, yürürlükteki ileti grubu ya da mantıksal ileti varsa arama başarısız olmaz; çağrıya MQCC\_UYARI tamamlanma kodu ile başarılı olabilir. [Çizelge 121 sayfa 818](#), oluşabilecek çeşitli vakaları gösterir. Bu durumlarda, tamamlanma kodu MQCC\_OK değilse, neden kodu aşağıdakilerden biridir (uygun olduğu gibi):
  - MQRC\_INCOMPLE\_GROUP
  - MQRC\_INCOMPLE\_MSG
  - MQRC\_INTUTARLMENT\_PERSISTENCE
  - MQRC\_INCONSISTENT\_UOW

**Not:** Kuyruk yöneticisi, MQPUT1 çağrısına ilişkin grup ve bölüm bilgilerini denetlemez.

Çizelge 121. MQPUT ya da MQCLOSE çağrısının grup ve bölüm bilgileriyle tutarlı olmadığı bir sonuç		
Yürürlükteki çağrı	Önceki arama MQPMO_LOGICAL_ORDER ile MQPUT oldu	Önceki arama MQPMO_LOGICAL_ORDER olmadan MQPUT oldu
MQPUT ile MQPMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQPMO_LOGICAL_ORDER olmadan MQPUT	MQCC_UYARI	MQCC_OK
Sonlandırılmamış bir grupta ya da mantıksal iletiyle MQCLOSE	MQCC_UYARI	MQCC_OK

İletileri ve bölümleri mantıksal sırada koyan uygulamalar için, kullanılacak en basit seçenek olduğu için MQPMO\_LOGICAL\_ORDER değerini belirtin. Bu seçenek, kuyruk yöneticisi bu bilgileri yönettiği için, grup ve bölüm bilgilerini yönetme gereksiniminin uygulanını giderir. Ancak, özelleştirilmiş uygulamaların MQPMO\_LOGICAL\_ORDER seçeneği tarafından sağlanandan daha fazla denetime gereksinimi olabilir. Bu durumda, bu seçeneğin belirlenmemesi sağlanabilir; bunu gerçekleştirdiğinizde, MQMD 'deki *GroupId*, *MsgSeqNumber*, *Offset* ve *MsgFlags* alanlarının her bir MQPUT ya da MQPUT1 çağrısından önce doğru olarak ayarlandığından emin olmanız gerekir.

Örneğin, bu iletilerin grup halinde mi, yoksa mantıksal ileti bölümleri mi olduğunu dikkate almadan, aldığı fiziksel iletileri iletmek isteyen bir uygulama, iki nedenden dolayı MQPMO\_LOGICAL\_ORDER belirtmemelidir:

- İletiler alındıysa ve sıraya konursa, MQPMO\_LOGICAL\_ORDER belirtilirse, iletilere yeni bir grup tanıtıcısı atar; bu, iletilerin kaynağı için ileti grubunun sonucundaki yanıt ya da rapor iletilerini ilintilendirmek için zorlanabilir ya da imkansız hale gelebilir.
- Kuyruk yöneticileri gönderme ve alma arasında birden çok yol içeren karmaşık bir ağda, fiziksel iletiler sıradan çıkabilirler. MQGET çağrısında MQPMO\_LOGICAL\_ORDER ve MQGMO\_LOGICAL\_ORDER belirtilmeden, iletme uygulaması her fiziksel iletiyi alır almaz ve mantıksal sırada gelecek mantıksal sırada beklemeden alabilir ve iletebilirler.

Grup ya da mantıksal ileti bölümlerindeki iletiler için rapor iletileri oluşturan uygulamaların, rapor iletisi yerleştirilirken MQPMO\_LOGICAL\_ORDER belirtmemesi de gerekir.

MQPMO\_LOGICAL\_ORDER, diğer MQPMO\_\* seçeneklerinin hiçbirisiyle belirtilebilir.

## Mantıksal Olarak Sıralı Grupları Kümelmiş Bir Kuyruğa Koyma (MQOO\_BIND\_ON\_GROUP)

MQOO\_BIND\_ON\_OPEN seçeneği, bu uygulamadaki tüm iletilerin ve dolayısıyla tüm grupların tek bir yönetim ortamına yöneltilmesini sağlar. Bu, uygulama trafiğinin bir küme kuyruğunun birden çok örneğinde dengeli olarak yüklenmemesi için bir dezavantaja sahiptir. İş yükü dengelemeyi etkin bir şekilde tutarken, iş yükü dengelemeyi etkinleştirmek için aşağıdaki seçenekleri ayarlamamız gerekir:

- MQPUT çağrısının MQPMO\_LOGICAL\_ORDER belirtilmeli
- MQOPEN çağrısı aşağıdaki iki seçenekten birini belirtmelidir:
  - MQOO\_BIND\_ON\_GROUP
  - MQOO\_BIND\_AS\_Q\_DEF ve kuyruk tanımlamasının DEFBIND (GROUP) belirtmesi gerekir

İş yükü dengelemesi, kuyruğun MQCLOSE ve MQOPER gerektirmeden, iletilerin *gruplar arasında* yönlendirilmesini sağlar. *Gruplar arasında*, MQMF\_MSG\_IN\_GROUP 'un MQMD (v2) ya da MQMDE olarak ayarının olduğu ve devam etmekte olan bir kısmı tam grubu olmadığı anlamına gelir. Bir grup devam ederken, nesne tutamacındaki 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 değil, PUT çağrısı MQRC\_INCOMPLE\_GROUP ile başarısız olur.

Tek bir MQPUT işlevi MQPMO\_LOGICAL\_ORDER belirtmezse ve yürürlükteki grup etkin değilse, o ileti için iş yükü dengelemesi sürülüyorsa (bu MQOPEN çağrısının MQOO\_BIND\_NOT\_FIXED belirtmiş gibi).

MQOO\_BIND\_ON\_GROUP kullanan bir hedefe yönelik iletiler için gerçek konum yok. Gerçek konum hakkında daha fazla bilgi için bkz. [“İleti grupları” sayfa 41.](#)

### *Mantıksal iletileri gruplama*

Bir gruptaki mantıksal iletilerin kullanılmasının başlıca iki nedeni vardır:

- İletileri belirli bir sırada işlemeniz gerekebilir.
- Bir gruptaki her bir iletiyi ilgili bir şekilde işlemeniz gerekebilir.

Her iki durumda da, aynı uygulama örneğine sahip tüm grubu alın.

Örneğin, grubun dört mantıksal iletimden oluştuğunu varsayın. Uygulama koyma işlemi şu şekilde 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
```

Uygulama alma işlemi, gruptaki ilk ileti için MQGMO\_ALL\_MSGS\_AVAM seçeneğini belirtir. Bu, grup içindeki tüm iletiler ulaşmadan işlemin başlatılmamasını sağlar. Grup içindeki sonraki iletiler için MQGMO\_ALL\_MSGS\_AVAILD seçeneği yoksa kaldırılır.

Grubun ilk mantıksal ileti alındığında, grubun geri kalan mantıksal iletilerinin sırayla alındığından emin olmak için MQGMO\_LOGICAL\_ORDER seçeneğini kullanabilirsiniz.

Yani, uygulama alma şu şekilde 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 gruplamak için daha fazla örnek için bkz. [“Mantıksal iletilerin uygulama bölümlenmesi” sayfa 831](#) ve [“İş birimlerine yayılan bir grubu koymak ve almak” sayfa 819.](#)

Bir uygulamanın, küme kuyukları için aynı hedef somut örneğe ayrılmış bir ileti grubunu istemesine izin verilmesine ilişkin bilgi için [DefBind](#) başlıklı konuya bakın.

### *İş birimlerine yayılan bir grubu koymak ve almak*

Önceki durumda, iletiler ya da kesimler düğümünden ayrılmadan (hedef uzakta) ya da tüm grup çalışmaya başlayınca ve iş birimi kesinleştirilinceye kadar, alınmaya başlayamaz. Bu, tüm grubu koymak uzun sürerse ya da düğüm üzerinde kuyruk alanı sınırlıysa, istediğiniz bu olmayabilir. Bunu aşmak için, grubu birkaç ünite işe sokun.

Grup birden çok iş birimi içine yerleştirilirse, uygulama koyma işlemi başarısız olduğunda da bazı grup tarafından kesinleştirilmesinin mümkün olur. Bu nedenle, uygulama, tamamlanmamış bir grubu sürdürmek için yeniden başlatma işleminden sonra kullanabileceği, her iş birimi ile kesinleştirilen durum bilgilerini kaydetmelidir. Bu bilgileri kaydetmenin en basit yeri STATUS kuyruğunda yer alıyor. Tam bir grup başarıyla ortaya konulduysa, STATUS kuyruğu boş olur.

Bölümlenme ilişkisine dahil olursa, mantık benzerdir. Bu durumda **StatusInfo** , *Offset* içermelidir.

Burada, grubu birkaç iş birimine yerleştirmenin bir örneği yer almaktadır:

```
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 yerleştirilir ve STATUS kuyruğu boş olur. Aksi takdirde, grubun durum bilgisiyle belirtilen noktada sürdürülmesi gerekir. MQPMO\_LOGICAL\_ORDER ilk kez koyma için kullanılamaz, ancak bundan sonra olabilir.

Yeniden başlatma işlemi şu şekilde 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 iyileştirir ve ayrıca, tüm grup için depolamanın gerekli olmadığı anlamına gelir. Avantajların farkına varmak için, her ileti grubu için birkaç iş birimi kullanın. Kurtarma nedenlerinden dolayı, her bir iletiyi bir iş birimi içinde almanız gerekir.

Buna karşılık gelen uygulamada olduğu gibi, durum bilgilerinin her bir iş birimi kesinleştirildiğinde otomatik olarak bir yere kaydedilmesini gerektirir. Yine, bu bilgileri kaydetmenin en basit yeri STATUS kuyruğunda yer alıyor. Tam bir grup başarıyla işlendiyse, STATUS kuyruğu boş olur.

**Not:** Ara düzey iş birimleri için, her bir MQPUT işleminin bir ileti bölümü olduğunu (MQMF\_SEGMENT işaretini ayarlayarak), her iş birimi için eksiksiz bir yeni ileti koymak yerine, STATUS kuyruğundan MQGET çağrılarını önleyebilirsiniz. Son iş biriminde, MQMF\_LAST\_SEGMENT değerini belirten durum kuyruğuna son bir kesim yerleştirilir ve MQGMO\_COMPLETE\_MSG belirtilerek, durum bilgileri bir MQGET ile temizlenir.

Yeniden başlatma işlemi sırasında, olası bir durum iletisi almak için tek bir MQGET kullanmak yerine, son kesime ulaşıncaya kadar (başka bir kesim döndürülünceye kadar) MQGMO\_LOGICAL\_ORDER ile durum kuyruğuna göz atın. Yeniden başlatmadan sonra ilk iş biriminde, durum bölümü yerleştirilirken açık olarak da görelili konum belirtin.

Aşağıdaki örnekte, yalnızca bir grup içindeki iletileri göz önünde bulundurarak, uygulamanın arabelleğinin tüm iletiyi tutmak için her zaman yeterince büyük olduğunu varsayarak, iletinin bölümlenip kesilmediğini dikkate alın. Bu nedenle, her MQGET üzerinde MQGMO\_COMPLE\_MSG belirtildi. Kesimlere ayırma ilişkisi varsa, aynı ilkeler geçerli olur (bu durumda, StatusInfo , *Offset* ' u içermelidir).

Basitlik için, tek bir UOW içinde en fazla 4 ileti alındığını varsayıyoruz:

```

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 alınır ve STATUS kuyruğu boş olur. Aksi takdirde, grubun durum bilgisiyle belirtilen noktada sürdürülmesi gerekir. MQGMO\_LOGICAL\_ORDER ilk alma işlemi için kullanılamıyor, ancak bundan sonra olabilir.

Yeniden başlatma işlemi şu şekilde 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

Bir kuyruktan belirli bir iletiyi almak için bir dizi yol vardır. Bunlar şunlardır: `MsgId` ve `CorrelId` üzerinde seçim yapmak, `GroupId`, `MsgSeqNumber` and `Offset` seçeneğinden ve `MsgToken`' da seçilip seçilmediğiniz. Ayrıca, kuyruğu açtığınızda bir seçim dizisi de kullanabilirsiniz.

Kuyruktan belirli bir ileti almak için, `MQMD` yapısının `MsgId` ve `CorrelId` alanlarını kullanın. Ancak, uygulamalar bu alanları belirttik olarak ayarlayabilirler, böylece belirttiğiniz değerler benzersiz bir iletiyi tanımlayamayabilir. Çizelge 122 sayfa 822 , bu alanların olası ayarları için hangi iletinin alındığını gösterir. `MQGET` çağrısının **GetMsgOpts** değiştirgesinde `MQGMO_MSG_UNDER_CURSOR` değerini belirlerseniz, bu alanlar girişte yoksayılır.

Çizelge 122. İleti ve ilinti tanıtıcılarının kullanılması		
Almak için ...	MsgId	CorrelId
Kuyrukta 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 *ilk* , seçim ölçütlerini karşılayan ilk ileti anlamına gelir (`MQGMO_BROWSE_NEXT` belirtildiyse, bu, seçim ölçütlerine uygun sırayla *sonraki* iletisi anlamına gelir).

Geri dönüşte, `MQGET` çağrısı `MsgId` ve `CorrelId` alanlarını, döndürülen iletinin ileti ve ilinti tanıtıcılarına ayarlar.

`MQMD` yapısının `Version` alanını 2 olarak ayarladıysanız, `GroupId`, `MsgSeqNumber` ve `Offset` alanlarını kullanabilirsiniz. Çizelge 123 sayfa 822 , bu alanların olası ayarları için hangi iletinin alındığını gösterir.

Çizelge 123. Grup tanıtıcısını kullanma	
Almak için ...	Eşleştirme seçenekleri
Kuyrukta ilk ileti	MQMO_NONE
MsgId ile eşleşen ilk ileti	MQMO_MATCH_MSG_ID
CorrelId ile eşleşen ilk ileti	MQMO_MATCH_COREL_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 ayarlanmasını belirtir.

2. MQMO işaretleri birleşimde kullanılabilir. For example, MQMO\_MATCH\_GROUP\_ID, MQMO\_MATCH\_MSG\_SEQ\_NUMBER, and MQMO\_MATCH\_OFFSET can be used together to give the segment identified by the GroupId, MsgSeqNumber, and Offset fields.
3. MQGMO\_LOGICAL\_ORDER belirtilirse, almaya çalıştığınız ileti etkilenir; bu seçenek, kuyruk tanıtıcısı için denetlenen durum bilgilerine bağlıdır. Bu konuya ilişkin bilgi için bkz. [“Mantıksal ve fiziksel sıralama” sayfa 811 ve Seçenekler.](#)

MQGET çağrısı genellikle bir kuyruktan ilk iletiyi alır. MQGET çağrısını kullandığınızda belirli bir ileti belirtirseniz, kuyruk yöneticisi iletiyi buluncaya kadar kuyruğun aranması gerekir. Bu, uygulamanızın performansını etkileyebilir.

If you are using Version 2 or later of the MQGMO structure and do not specify the MQMO\_MATCH\_MSG\_ID or MQMO\_MATCH\_CORREL\_ID flags, you do not need to reset the MsgId or CorrelId fields between MQGETs.

**z/OS** IBM MQ for z/OS işletim sistemi üzerinde, kuyruktaki MQGET işlemlerinin hızını artırmak için IndexType kuyruk özniteliği kullanılabilir. Daha fazla bilgi için bkz [“Dizin tipi” sayfa 827.](#)

MQGMO yapısındaki MsgToken değerini ve MatchOption MQMO\_MATCH\_MSG\_TOKEN değerini belirterek bir kuyruktan belirli bir ileti alabilirsiniz. MsgToken , MQPUT çağrısıyla bu iletiyi kuyruğa ilk olarak koyan ya da önceki MQGET işlemleri tarafından döndürülür ve kuyruk yöneticisi yeniden başlatılmadıkça, sabit kalır.

Kuyruktaki yalnızca bir ileti alt kümesiyle ilgileniyorsanız, MQOL ya da MQSUB çağrısıyla bir seçim dizgisi kullanarak işlenmesini istediğiniz iletileri belirleyebilirsiniz. Sonra MQGET, o seçim dizesini karşılayan bir sonraki iletiyi alır. Seçim dizgileri hakkında daha fazla bilgi için bkz. [“Seçiciler” sayfa 29.](#)

### ***Kalıcı olmayan iletilerin performansını artırma***

İstemci bir sunucudan ileti gerektirdiğinde, sunucuya bir istek gönderir. Tükettiği iletilerin her biri için ayrı bir istek gönderir. Bir istemcinin, bu istek iletilerini göndermek zorunda kalmaktan kaçınarak, kalıcı olmayan iletileri tüketen bir istemcinin performansını artırmak için, bir istemci *devamını oku'* yi kullanacak şekilde yapılandırılabilir. Önden okuma, bir uygulamanın istekte bulunmadan istemciye gönderilmesine izin verir.

When read ahead is enabled, messages are sent to a memory buffer on the client called the *okuma yazma arabelleği*. İstemci, önceden okuma etkinleştirilmiş olarak açık olduğu her kuyruk için okuma arabelleğiyle ilgili bir okuma arabelleği'yecektir. İleriye okuma arabelleğindeki iletiler kalıcı olarak saklanmaz. İstemci düzenli olarak sunucuyu, tükettiği veri miktarına ilişkin bilgilerle güncelleştirir.

MQOPEN ' i MQOO\_READ\_ÖNCEDEDEN ile çağırdığınızda, IBM MQ istemcisi belirli koşullar karşılandığında yalnızca önden okumayı etkinleştirir. Bu koşullar şunlardır:

- Hem istemci, hem de uzak kuyruk yöneticisi IBM WebSphere MQ 7.0 ya da sonraki bir yerde olmalıdır.
- 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.

İleriye okumanın kullanılması, bir istemci uygulamasından kalıcı olmayan iletiler tüketirken performansı yükseltebilirler. Bu performans iyileştirmesi, hem MQI hem de JMS uygulamaları için kullanılabilir. MQGET ya da zamanuyumsuz tüketimi kullanan istemci uygulamaları, kalıcı olmayan iletiler tüketildiğinde performans iyileştirmelerinden yararlanırlar.

İleriye okuma ile birlikte kullanım için tüm seçenekler desteklenmediği için, tüm istemci uygulama tasarımları okuma öncesinde kullanılması için uygun değildir ve önceden okuma etkinleştirildiğinde, MQGET çağrıları arasında tutarlı olması için bazı seçeneklerin kullanılması gerekir. Bir istemci, seçim ölçütlerini MQGET çağrıları arasında değiştirirse, ileriye okuma arabelleğindeki saklanmakta olan iletiler, istemcinin önünde okuma arabelleğindeki iplikçik olarak kalır.

Önceki seçim ölçütlerine sahip bir birikim arka günlüğü artık gerekmiyorsa, istemciden bu iletilerin istemciden otomatik olarak temizlenmesi için istemcide yapılandırılabilir bir temizleme aralığı

ayarlanabilir. Temizleme aralığı, istemci tarafından belirlenen ileriye doğru ayarlama seçenekleri gruplarından biridir. Gereksinimlerinizi karşılamak için bu seçenekleri ayarlamak mümkündür.

Bir istemci uygulaması yeniden başlatılırsa, ileriye okuma arabelleğindeki iletiler kaybedilebilir. Ters durumda, ileriye doğru okuma arabelleğiyle taşınan bir ileti, temeldeki kuyruktan silinebilir; bu, arabellekten kaldırılmamasına neden olmaz, bu nedenle okuma öncesinde kullanılan bir MQGET çağrısı, artık var olmayan bir iletiyi geri döndürebilir.

Önceden okuma, yalnızca istemci bağ tanımları için gerçekleştirilir. Diğer tüm bağ tanımlamalar için öznitelik yoksayıdır.

İleriye okumanın tetikleme konusunda hiçbir etkisi yoktur. İstemci tarafından ileriye doğru bir ileti okunduğunda tetikleme iletileri oluşturulmaz. Önceden okuma, etkinleştirildiğinde, muhasebe ve istatistik bilgileri oluşturmaz.

## **Yayınlama ile birlikte okuma yazma ileti alışverişi**

Abone olunan bir uygulama, yayınların gönderildiği hedef kuyruğunu belirtiyorsa, varsayılan okuma değeri olarak, belirlenen kuyruğun DEFREADA değeri kullanılır.

When a subscribing application requests that IBM MQ manages the destination to which publications are sent, a managed queue is created as a dynamic queue based upon a predefined model queue. Bu, varsayılan okuma değeri değeri olarak kullanılan model kuyruğunun DEFREADA değeridir. Varsayılan model kuyrukları SYSTEM.DURABLE.PUBLICATIONS.MODEL ya da SYSTEM.NONDURABLE.PUBLICATIONS.MODEL , bu ya da bir üst konu için bir model kuyruğu tanımlanmadıkça kullanılır.

### **İlgili kavramlar**

[“AIXüzerinde kalıcı olmayan iletiler için performans ayarlaması” sayfa 826](#)

AIX V5.3 ya da sonraki bir sürümünü kullanıyorsanız, ayarlama parametresini, kalıcı olmayan iletiler için tam performans kullanacak şekilde ayarlamayı düşünün.

### **İlgili görevler**

[“Okuma öncesinde okuma ve devre dışı bırakma” sayfa 825](#)

Varsayılan olarak okuma seçeneği geçersiz kılınmaktadır. Okuma öncesinde okuma işlemini kuyruksuz ya da uygulama düzeyinde etkinleştirebilirsiniz.

### **İlgili başvurular**

[“MQGET seçenekleri ve okuma önerisi” sayfa 824](#)

Önceden okuma etkinleştirildiğinde tüm MQGET seçenekleri desteklenmez; MQGET çağrıları arasında tutarlı olması için bazı seçenekler gereklidir.

#### *MQGET seçenekleri ve okuma önerisi*

Önceden okuma etkinleştirildiğinde tüm MQGET seçenekleri desteklenmez; MQGET çağrıları arasında tutarlı olması için bazı seçenekler gereklidir.

MQOPEN ' i MQOO\_READ\_ÖNCEDEN ile çağırdığınızda, IBM MQ istemcisi belirli koşullar karşılandığında yalnızca önden okumayı etkinleştirir. Bu koşullar şunlardır:

- Hem istemci, hem de uzak kuyruk yöneticisi IBM WebSphere MQ 7.0 ya da sonraki bir yerde olmalıdır.
- 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, ileriye okuma ve MQGET çağrıları arasında değiştirilip değiştirilmedikleri için hangi seçeneklerin desteklendiğini gösterir.



Çizelge 124. MQGET seçenekleri ve okuma önerisi

MQGET değerleri ve seçenekleri	Önceden okuma etkinleştirildiğinde izin verilir ve MQGET çağruları arasında değiştirilebilir <sup>5</sup>	Okuma önden okuma etkinleştirildiğinde izin verilir, ancak MQGET çağruları arasında değiştirilemez <sup>1</sup>	Önceden okuma etkinleştirildiğinde izin verilmeyen MQGET Seçenekleri geçerli kılındığında <sup>2</sup>
MQGET MQMD değerleri	MsgId <sup>3</sup> CorrelId <sup>3</sup>	Kodlama CodedCharSetId	
MQGET MQGMO Seçenekleri	<ul style="list-style-type: none"> <li>MQGMO_NO_BEKLEME</li> <li>MQGMO_BROWSE_MESSAGE_UNDER_CURSOR</li> <li>MQGMO_BROWSE_FIRST</li> <li>MQGMO_BROWSE_NEXT</li> <li>MQGMO_FAIL_IF QUIESCING</li> </ul>	<ul style="list-style-type: none"> <li>MQGMO_SYNCPOINT_IF_PERSISTENSE</li> <li>MQGMO_NO_SYNCPOINT</li> <li>MQGMO_ACCEPT_TICHTED_MSG</li> <li>MQGMO_CONVERT</li> </ul>	<ul style="list-style-type: none"> <li>MQGMO_SET_SIGNAL</li> <li>MQGMO_SYNCPOINT</li> <li>MQGMO_MARK_SKIP_BACKUT</li> <li>MQGMO_MSG_UNDER_CURSOR <sup>4</sup></li> <li>MQGMO_LOCK</li> <li>MQGMO_UNLOCK</li> <li>MQGMO_LOGICAL_ORDER</li> <li>MQGMO_COMPLE_MSG</li> <li>MQGMO_ALL_MSGS_AVALABILIR</li> <li>MQGMO_ALL_SEGMENTS_KULLANILABILIR</li> </ul>

#### Notlar:

1. Bu seçenekler MQGET çağruları arasında değiştirilirse, bir MQRC\_OPTIONS\_CHANGED neden kodu döndürülür.
2. Bu seçenekler ilk MQGET çağrısında belirtilirse, önceden okuma geçersiz kılınır. İzleyen bir MQGET çağrısında bu seçenekler belirtilirse, MQRC\_OPTIONS\_ERROR neden kodunda bir neden kodu döndürülür.
3. Bir istemci uygulaması, MQGET çağruları arasında MsgId ve CorrelId değerlerini değiştirirse, önceki değerlere sahip iletiler istemciye önceden gönderilmiş olabilir ve tüketilmeye kadar (ya da otomatik olarak temizlenen) istemcide önceden okuma arabelleğindeki iletiler kalır.
4. MQGMO\_MSG\_UNDER\_CURSOR önden okuma ile olanaklı değil. Kuyruk açılırken, hem MQOO\_BROWSE hem de MQOO\_INPUT\_SHARED ya da MQOO\_INPUT\_EXCLUSIVE seçeneklerinden biri belirtildiğinde, okuma öncesinde okuma geçersiz kılınır.
5. İleriye okuma etkinleştirildiğinde, iletilerin bir kuyruktan mı göz atılacağını, yoksa kuyruktan mı atılacağını ilk MQGET belirler. İstemci uygulaması değiştirilen seçeneklerle MQGET işlevini kullanıyorsa (örneğin, bir ilk alma işlemini izlemeye ya da ilk göz atma işlemini izlemeye çalışmak gibi), bir MQRC\_OPTIONS\_CHANGED neden kodu döndürülür.

Bir istemci, seçim ölçütlerini MQGET çağruları arasında değiştirirse, ilk seçim ölçütleriyle eşleşen okuma öncesinde saklanan iletiler istemci uygulaması tarafından tüketilmez ve istemci okuma değeri arabelleğindeki iplikçik olarak kalmaya devam eder. İstemcinin önceden okuma arabelleğindeki birçok hata içeren ileti içerdiği durumlarda, önden okuma ile ilişkili yararlar kaybedilir ve her ileti tüketilen her ileti için sunucuya ayrı bir istek gerekir. Önden okuma işleminin etkin bir şekilde kullanılıp kullanılmadığını belirlemek için, READA bağlantı durumu parametresini kullanabilirsiniz.

İlk MQGET çağrısında belirtilen uyumsuz seçenekler nedeniyle, bir uygulama tarafından istendiğinde okuma engelleyici olabilir. Bu durumda, bağlantı durumu engellendikçe öndeki okunurları gösterir.

MQGET ile ilgili bu kısıtlamalardan ötürü, ileride bir istemci uygulaması tasarımının okunmaya uygun olmadığını, MQOO\_READ\_AHEAD\_NO adlı MQOO\_READE 'yi belirtin. Diğer bir seçenek olarak, açılmakta olan kuyruğun varsayılan okuma tamamlama değeri NO ya da DISABLE olarak değiştiriliyor.

#### Okuma öncesinde okuma ve devre dışı bırakma

Varsayılan olarak okuma seçeneği geçersiz kılınmaktadır. Okuma öncesinde okuma işlemini kuyruğa ya da uygulama düzeyinde etkinleştirebilirsiniz.

## Bu görev hakkında

MQOPEN ' i MQOO\_READ\_ÖNCEDEDEN ile çağırdığınızda, IBM MQ istemcisi belirli koşullar karşılandığında yalnızca önden okumayı etkinleştirir. Bu koşullar şunlardır:

- Hem istemci, hem de uzak kuyruk yöneticisi IBM WebSphere MQ 7.0 ya da sonraki bir yerde olmalıdır.
- 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.

İleriye okumayı etkinleştirmek için:

- Okuma öncesinde kuyruk düzeyinde okuma yapılandırmak için, kuyruk özniteliğini, DEFREADA ' yı YES değerine ayarlayın.
- Uygulama düzeyinde okuma öbeklerini yapılandırmak için:
  - MQOPEN işlev çağırısındaki MQOO\_READ\_AHEAD seçeneğini mümkün olan her yerde kullanmak için okuma seçeneğini kullanın. DEFREADA kuyruk özniteliği DISABLE olarak ayarlandıysa, istemci uygulamasının okuma öncesinde okuma kullanması mümkün değildir.
  - Bir kuyruğun üzerinde okuma yazma özelliği etkinleştirildiğinde, önceden okunmayı kullanmak için, MQOPER işlev çağırısında MQOO\_READ\_AHEAD\_AS\_Q\_DEF seçeneğini kullanın.

İleride bir istemci uygulaması tasarımı okumak için uygun değilse, bu tasarımı devre dışı bırakabilirsiniz:

- Kuyruk özniteliğini ayarlayarak, bir istemci uygulaması tarafından istenmedikçe, önceden okunmasını istemiyorsanız, DEFREOKA 'yı HAYIR olarak ayarlayarak, önceden okunmanın bir istemci uygulaması tarafından gerektirip okunmamasından bağımsız olarak okunmasını istemiyorsanız, DEVRE Dışı olarak kullanılmasını istemiyorsanız, DEFREADA' da (Hayır) kullanılmasını istemiyorsanız, DEFREADA ' yı (NO) belirleyerek, önceden okuma işlemi yapmanız gerekmez.
- MQOP işlev çağırısındaki MQOO\_NO\_READ\_AHEAD seçeneğini kullanarak uygulama düzeyinde.

İki MQCLOSE seçeneği, kuyruk kapatılırsa, okuma öncesinde okuma arabelleğinde saklanan iletilere ne olacağını yapılandırmanızı sağlar.

- Okuma öbekleri arabelleğindeki iletileri atmak için MQCO\_IMMEDIATE deyimini kullanın.
- İleriye okuma arabelleğindeki iletilerin, kuyruk kapatılmadan önce uygulama tarafından tüketildiğinden emin olmak için MQCO\_QUIESCE seçeneğini kullanın. MQCO\_QUIESCE ile MQCLOSE komutu verildiğinde ve ileriye doğru okuma arabelleğinde kalan iletiler varsa, MQRC\_READ\_AHEAD\_MSGS, MQCC\_UYARI ile döner.

*AIX üzerinde kalıcı olmayan iletiler için performans ayarlaması*

AIX V5.3 ya da sonraki bir sürümünü kullanıyorsanız, ayarlama parametresini, kalıcı olmayan iletiler için tam performans kullanacak şekilde ayarlamayı düşünün.

Ayarlama parametresini hemen yürürlüğe girebilmesi için ayarlamak üzere kök kullanıcı olarak şu komutu verin:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

Eniyileme değiştirgesini hemen yürürlüğe gireceği ve yeniden başlatma işlemi üzerinde devam edecek şekilde ayarlamak için, kök kullanıcı olarak aşağıdaki komutu verin:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Olağan durumda, kalıcı olmayan iletiler yalnızca bellekte tutulur, ancak AIX , kalıcı olmayan iletileri diske yazılacak şekilde zamanlayabileceği durumlar da vardır. Diske yazılması planlanan iletiler, disk yazma işlemi tamamlanincaya kadar MQGET için kullanılamaz. Önerilen ayarlama komutu bu eşik değerini gösterir; 16 kilobayt veri kuyruğa alındığında iletilerin diske yazılacağı zamanlamak yerine, diske yazma

işlemi yalnızca makineden gerçek saklama alanı tam olarak kapandığında gerçekleşir. Bu, genel bir değişiktir ve diğer yazılım bileşenlerini etkileyebilir.

AIXişletim sistemi, çok iş parçacıklı uygulamalar kullanırken ve özellikle birden çok işlemcili makinelerde çalışırken, uygulama başlatılmadan önce ortamda AIXTHREAD\_SCOPE=S ayarını .profile mqm tanıtıcısına ya da AIXTHREAD\_SCOPE=S ayarlamasını, daha iyi performans ve daha sağlam zamanlama için önemle öneririz. Örneğin:

```
export AIXTHREAD_SCOPE=S
```

AIXTHREAD\_SCOPE=S ayarı, varsayılan özniteliklerle yaratılan kullanıcı iş parçacıklarının sistem çapında çekişme kapsamına konacağı anlamına gelir. Sistem çapında çekişme kapsamı ile bir kullanıcı iş parçacığı yaratılırsa, bu iş parçacığı bir çekirdek iş parçacığına bağlanır ve bu iş parçacığın çekirdeğe göre zamanlanır. Temeldeki çekirdek iş parçacığı, başka bir kullanıcı iş parçacığıyla paylaşılmaz.

## Dosya tanımlayıcıları

Aracı işlemi gibi çok iş parçacıklı bir işlemi çalıştırırken, dosya açıklayıcıları için yumuşak sınıra ulaşabilirsiniz. This limit gives you the IBM MQ reason code MQRC\_UNEXPECTED\_ERROR (2195) and, if there are enough file descriptors, an IBM MQ FFST™ file.

Bu sorunu önlemek için, dosya tanımlayıcılarının sayısı için süreç sınırını artırabilirsiniz. To do so, alter the nofiles attribute in /etc/security/limits to 10,000 for the mqm user ID or in the default stanza.

## Sistem Kaynağı Sınırları

Bir komut isteminde aşağıdaki komutları kullanarak, veri bölümü ve yığın bölümü için sistem kaynağı sınırını sınırsız olarak ayarlayın:

```
ulimit -d unlimited  
ulimit -s unlimited
```

## Dizin tipi

Kuyruk özniteliği ( *IndexType*), kuyruk yöneticisinin kuyruklardaki MQGET işlemlerinin hızını artırmak için sürdüreceği dizin tipini belirtir.

**Not:** Yalnızca IBM MQ for z/OSüzerinde desteklenir.

Beş seçeneğiniz var:

Değer	Tanım
YOK	Hiçbir dizin korunmadı. Bu seçeneği, iletileri sırayla alırken kullanın (bkz. <a href="#">“Öncelik” sayfa 811</a> ).
GRUPID	Grup tanıtıcılarından oluşan bir dizin sürdürür. İleti gruplarının mantıksal sıralamasını istiyorsanız, bu dizin tipini kullanmanız gerekir (bkz. <a href="#">“Mantıksal ve fiziksel sıralama” sayfa 811</a> ).
MSGID	İleti tanıtıcılarından oluşan bir dizin tutulur. Use this when retrieving messages using the <i>MsgId</i> field as a selection criterion on the MQGET call (see <a href="#">“Belirli bir iletiyi alma” sayfa 822</a> ).
MSGTOKE	İleti simgeleriyle ilgili bir dizin sürdürür.
CORLONID	Bir ilinti tanıtıcılarının dizini korunur. Use this when retrieving messages using the <i>CorrelId</i> field as a selection criterion on the MQGET call (see <a href="#">“Belirli bir iletiyi alma” sayfa 822</a> ).

**Not:**

1. MSGID seçeneğini ya da CORRELID seçeneğini kullanarak dizinleniyorsanız, MQMD 'deki görelî **MsgId** ya da **CorrelId** değıştîrgelerini ayarlayın. İkisini de ayarlamak faydalı değîl.
2. Bir kuyruk ařağıdaki tüm kořullarla eřleřirse, bir iletiyi bulmak için dizin mekanizmasını kullanır:
  - Dizin tipi MSGID, CORRELID ya da GROUPID içerişor
  - Aynı tip kimlikle göz atılır
  - Yalnızca bir öncelięe sahip iletileri var
3. Bu iřlem yeniden bařlatma süresini etkiledięinden, kuyruklar ( *MsgId* ya da *CorrelId* tarafından dizinlenmiř) binlerce ileti içeren kuyruklardan kaçının. (Bu, yeniden bařlatma sırasında silindikleri için kalıcı olmayan iletiler için geçerli değîldir.)
4. MSGTOKEN, z/OS iř yükü yöneticisi tarafından yönetilen kuyrukları tanımlamak için kullanılır.

**IndexType** öznitelięinin tam açıklaması için bkz. **IndexType**. **IndexType** öznitelîge iliřkin ek bilgi için bkz. "z/OS uygulamaları için tasarım ve performans konuları" sayfa 61.

#### **4 MB ' den büyük iletilerin iřlenmesi**

İletiler, uygulama, kuyruk ya da kuyruk yöneticisi için çok büyük olabilir. Depending on the environment, IBM MQ provides a number of ways of dealing with messages that are longer than 4 MB.

**MaxMsgLength** öznitelîğini, V6 ya da sonraki yayın düzeylerindeki tüm IBM MQ sistemlerinde 100 MB ' ye yükseltebilirsiniz. Kuyruęu kullanan iletilerin büyüklüęünü yansıtmak için bu değeri ayarlayın. IBM MQ for z/OS'diřındaki IBM MQ sistemlerinde de řunları yapabilirsiniz:

1. Kesimlere ayrılmıř iletileri kullanın. (İletiler, uygulama ya da kuyruk yöneticisi tarafından kesimlere ayrılabilir.)
2. Bařvuru iletilerini kullanın.

Bu yaklařımların her biri bu bölümün geri kalan kısmında açıklanmıřtır.

#### **İleti uzunluęu üst sınırını artırma**

**MaxMsgLength** kuyruk yöneticisi öznitelîęi, bir kuyruk yöneticisi tarafından iřlenebilecek bir iletinin uzunluk üst sınırını tanımlar. Benzer řekilde, **MaxMsgLength** kuyruk öznitelîęi, bir kuyruk tarafından iřlenebilecek bir iletinin uzunluk üst sınısıdır. Desteklenen varsayılan ileti uzunluęu üst sınırı, çalıřmakta olduęunuz ortama baęlıdır.

Büyük iletileri ele aldıysanız, bu öznitelikleri z/OS'diřındaki altyapılarda baęımsız olarak değıştîrebilirsiniz. Kuyruk yöneticisi öznitelik değeri 32768 bayt-100 MB aralıęında ayarlayabilirsiniz.



**Uyarı:** On IBM MQ for z/OS the **MaxMsgLength** attribute of the queue manager is hard coded at 100 MB.

Tüm altyapılarda, kuyruk öznitelîęi değeri 0-100 MB aralıęında ayarlayabilirsiniz.

**MaxMsgLength** özniteliklerinin birini ya da her ikisini değıştirdikten sonra, değışikliklerin yürürlüęe girdięinden emin olmak için uygulamalarınızı ve kanallarınızı yeniden bařlatın.

Bu değışiklikler yapıldıęında, ileti uzunluęu hem kuyruktan, hem de kuyruk yöneticisi **MaxMsgLength** özniteliklerinden küçük ya da ona eřit olmalıdır. Ancak, var olan iletiler özniteliklerden daha uzun olabilir.

İleti kuyruk için çok büyükse, MQRC\_MSG\_TOO\_BIG\_FOR\_Q döndürülür. Benzer řekilde, ileti kuyruk yöneticisi için çok büyükse, MQRC\_MSG\_TOO\_BIG\_FOR\_Q\_MGR ileti döndürülür.

Büyük iletilerin ele alma yöntemi kolay ve uygundur. Ancak, ařağıdaki etkenleri kullanmadan önce ařağıdaki etkenleri göz önünde bulundurun:

- Kuyruk yöneticileri arasındaki benzersizlik azaltılır. İleti verilerinin büyüklük üst sınırı, iletinin konacaęı her kuyruk (iletim kuyrukları da içinde olmak üzere) için *MaxMsgLength* tarafından belirlenir. Bu değeri genellikle, kuyruk yöneticisinin *MaxMsgLength*, özellikle de iletim kuyrukları için varsayılan olarak varsayılan olarak belirlenir. Bu, uzak bir kuyruk yöneticisine seyahat etmek olduęunda iletinin çok büyük olup olmadıęını tahmin etmeyi zorlařtırır.

- Sistem kaynaklarının kullanımı artırılır. Örneğin, uygulamaların daha büyük arabelleklere ve bazı platformlarda, paylaşılan saklama alanı kullanımı artırılmış olabilir. Kuyruk depolaması yalnızca daha büyük iletiler için gerekliyse etkilenmelidir.
- Kanal toplu işi etkileniyor. Büyük bir ileti, toplu iş sayısına doğru yalnızca bir ileti olarak sayılır, ancak iletilmeye daha uzun gereksinim duyar, böylece diğer iletiler için yanıt sürelerini artırır.

### Multi

#### İleti bölümlenmesi

Bölüm iletileri hakkında 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 sınırını artırma” sayfa 828 konusunda açıklandığı gibi ileti uzunluğu üst sınırının artırılması bazı olumsuz etkilerine yol açıyor. Ayrıca, ileti kuyruğun ya da kuyruk yöneticisi için çok büyük olmasına neden olabilir. Bu durumda, bir iletiyi bölümlenebilirsiniz. Bölümlerle ilgili bilgi için bkz. “İleti grupları” sayfa 41.

Sonraki bölümler, bölümlenmiş iletiler için ortak kullanımlara bakın. Koyma ve geçici olarak alma işlemi için, *her zaman* MQPUT ya da MQGET çağrılarının bir iş birimi içinde çalışması olduğu varsayılır. Bu tekniği her zaman, ağda eksik grupların mevcut olma olasılığını azaltmak için kullanmayı düşünün. Kuyruk yöneticisi tarafından tek aşamalı kesinleştirme kabul edilir, ancak diğer koordinasyon teknikleri eşit olarak geçerlidir.

Ayrıca, uygulama alma işlemlerinde, aynı kuyruğu birden çok sunucu işliyorsa, her bir sunucu benzer kodu yürütür; böylece, bir sunucu, orada olmayı beklediği bir ileti ya da kesimi bulamazsa (daha önce MQGMO\_ALL\_MSGS\_AVAILD ya da MQGMO\_ALL\_SEGMENTS\_AVAILILIR belirtmiş olduğu için) hiçbir zaman başarısız olur.

## İş birimlerine yayılan bölümlü bir ileti alınıyor ve alınıyor

“İş birimlerine yayılan bir grubu koymak ve almak” sayfa 819' e benzer bir şekilde bir çalışma birimine yayılan bölümlü bir ileti yerleştirebilir ve alabilirsiniz.

Ancak, bölümlenmiş iletileri genel bir iş birimine yerleştiremez ya da bu iletileri bölümlere ayırtamazsınız.

### Multi

#### Kuyruk yöneticisine göre bölümlendirme ve yeniden birleştirme

Bu, bir uygulamanın başka bir uygulama tarafından alınması gereken bir iletiyi yerleştirdiği en basit senaryodur. İleti büyük olabilir: put ya da alma uygulaması tek bir arabelleğiyle başa çıkmak için çok büyük değil, ancak kuyruk yöneticisi için çok büyük ya da iletinin konacağı bir kuyruk için çok büyük.

Bu uygulamalar için gerekli olan tek değişiklikler, uygulamanın kuyruk yöneticisine, gerekli durumlarda segmentasyon gerçekleştirmesi için yetki vermesi amacıyla gerçekleştirilir:

```
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 uygulama alma işlemi için, kuyruk yöneticisine, bölümlenmiş bir ileti varsa, iletiyi yeniden bir araya getirmesini istemesi için:

```
GMO.Options = MQGMO_COMPLETE_MSG | (existing options)
MQGET
```

Bu en basit senaryoda, uygulama, MQPUT çağrısından önce GroupId alanını MQGI\_NONE olarak sıfırlamalıdır, böylece kuyruk yöneticisi her ileti için benzersiz bir grup tanıtıcısı oluşturabilirler. Bu yapılmıyorsa, ilgisiz iletiler aynı grup tanıtıcısına sahip olabilir ve bu da daha sonra yanlış işleme yol açabilir.

Uygulama arabelleği, yeniden birleştirilen iletiyi içerecek kadar büyük olmalıdır (MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneğini eklemediyseniz).

Bir kuyruğun MAXMSGLEN özneliği, ileti bölümlenmesi için değiştirilecek şekilde değiştirilecekse, şunları dikkate alın:

- Yerel kuyruklarda desteklenen ileti kesimi alt sınırı 16 byte 'tır.
- İletim kuyruğu için, MAXMSGLEN, üstbilgiler için gereken alanı da içermelidir. İletim kuyruğuna konabilecek herhangi bir ileti kesiminde, beklenen kullanıcı verisi uzunluğu üst sınırından en az 4000 bayt daha büyük bir değer kullanın.

Veri dönüştürme gerekiyorsa, uygulama alma işlemi MQGMO\_CONVERT belirterek bunu yapmak zorunda kalabilirler. Veri dönüştürme çıkışı eksiksiz bir iletiyle sunulduğu için bu açık bir şekilde olmalıdır. İleti bölümlendiye, verileri gönderen kanalına dönüştürme girişiminde bulunmayın; verilerin biçimi veri dönüştürme çıkışıysa, eksik verilerde dönüştürme işlemini gerçekleştirilemez.

### Multi Uygulama bölümlenmesi

Uygulama bölümlenmesi, kuyruk yöneticisi bölümlenmesi yeterli olmadığı ya da uygulamaların belirli bölüm sınırları ile veri dönüştürme gerektirdiğinde kullanılır.

Uygulama bölümlenmesi iki temel neden için kullanılır:

1. İleti, uygulamalar tarafından tek bir arabellekte işlenmeyecek kadar büyük olduğundan, kuyruk yöneticisi bölümlenmesinin tek başına yeterli olmadığını belirten bir ileti vardır.
2. Veri dönüştürme, gönderen kanalları tarafından gerçekleştirilmelidir ve biçim, bir kesimin tek bir kesimin mümkün olması için kesim sınırlarının nerede olacağını belirtmesi gerektiğini ifade eder.

Ancak, veri dönüştürme bir sorun değilse ya da alma uygulaması her zaman MQGMO\_COMPLETE\_MSG kullanırsa, kuyruk yöneticisi bölümlenmesine izin verilirse, MQMF\_SEGMENTATION\_ALLOWEND belirtilerek izin verilir. Örneğimizde, uygulama iletiyi dört kesime 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
```

If you do not use MQPMO\_LOGICAL\_ORDER, the application must set the *Offset* and the length of each segment. Bu durumda, mantıksal durum otomatik olarak korunmaz.

Uygulama alma işlemi, yeniden birleştirilen bir iletiyi tutacak kadar büyük bir arabelleğe sahip olduğunu garanti edemez. Bu nedenle, kesimleri tek tek işlemeye hazırlanmalıdır.

Bölümlenmiş iletiler için, bu uygulama, mantıksal iletiyi oluşturan tüm kesimler varsa, bir bölümü işlemeye başlamak istemiyor. Bu nedenle, ilk bölüm için MQGMO\_ALL\_SEGMENTS\_AVAILABLE MQGMO\_LOGICAL\_ORDER ögesini belirtirseniz ve yürürlükteki bir mantıksal ileti varsa, MQGMO\_ALL\_SEGMENTS\_AVAILABLE.

Mantıksal iletinin ilk bölümü alındıktan sonra, mantıksal iletinin kalan bölümlerinin sırayla alındığından emin olmak için MQGMO\_LOGICAL\_ORDER kullanın.

Farklı gruplar içindeki iletilere göz önünde bulunmuklanmaz. Bu tür iletiler gerçekleşirse, kuyrukta her iletinin ilk bölümünün gerçekleştirildiği 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 gruptaki mantıksal sırada sürdürülmelidir; bunların bazıları ya da tümü, uygulama bölümlemesi gerektirmeleri için bu kadar büyük olabilir.

Örneğimizde, dört mantıksal ileti içeren bir grup ortaya konabiliyor. Üçüncü iletinin tümü büyük, ancak uygulama koyma işlemi tarafından gerçekleştirilen bölümlemeye gerek duyarsınız:

```
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
```

Uygulama alma uygulamasında, ilk MQGET ' de MQGMO\_ALL\_MSGS\_AVAM değeri belirtildi. Bu, grubun tamamı kullanılabilir oluncaya kadar bir grubun hiçbir ileti ya da kesiminin alınmamasını sağlar. Bir grubun ilk fiziksel ileti 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 ögesini belirtirseniz ve yürürlükteki bir grup varsa, MQGMO\_ALL\_MSGS\_AVAM yok sayılır.

### Başvuru iletileri

Başvuru iletileri hakkında daha fazla bilgi edinmek için bu bilgileri kullanın.

**Not:** IBM MQ for z/OS içinde desteklenmez.

Bu yöntem, büyük bir nesnenin bir düğümden diğerine, kaynak ya da hedef düğümlerde bulunan IBM MQ kuyruklarında saklanmadan başka bir düğümden diğerine aktarılmasına olanak sağlar. Bu, örneğin, posta uygulamaları için başka bir formda (örneğin, başka bir formda) var olduğunda bu avantajdan yararlanmaktan çok daha iyi olur.

Bunu yapmak için, bir kanalın her iki ucunda da bir ileti çıkışı belirtiyorsunuz. Bunun nasıl yapacağını ilişkin bilgi için bkz. [“Kanal ileti çıkış programları” sayfa 1010](#).

IBM MQ , bir başvuru ileti üstbilgisinin biçimini tanımlar (MQRMH). Bu konuya ilişkin açıklamalar için MQRMH başlıklı konuya bakın. Bu, tanımlı bir biçim adıyla tanınır ve gerçek veriler tarafından izlenebilir.

Bir uygulama, büyük bir nesnenin aktarımı başlatmak için, aşağıdaki verileri izleyen herhangi bir veri olmadan, başvuru ileti üstbilgisinden oluşan bir ileti yerleştirebilir. Bu ileti düğümü bıraktıkça, ileti çıkışı nesneyi uygun bir şekilde alır ve başvuru iletiye ekler. Daha sonra iletiyi almak üzere Message Channel Agent 'ı göndermek üzere gönderilen Message Channel Agent 'a (öncekinden daha büyük) iletiyi döndürür.

Alıcı MCA ' da başka bir ileti çıkışı yapılandırılıyor. When this message exit receives one of these messages, it creates the object using the object data that was appended and passes on the reference message *bu*

*olmadan* it. Başvuru iletisi artık bir uygulama tarafından alınabilir ve bu uygulama, bu düğümde nesnenin (ya da en azından bu başvuru iletisi tarafından temsil edilen kısmı) oluşturulduğunu biliyor.

Bir ileti çıkışısının başvuru iletisine ekleyebileceği nesne verisi miktarı üst sınırı, kanal için kararlaştırılan ileti uzunluğu üst sınırı ile sınırlanır. Çıkış, iletileceği her ileti için MCA ' ya yalnızca tek bir ileti döndürebilir; bu nedenle, uygulama koyma işlemi bir nesnenin aktarılmasına neden olacak birkaç ileti koyabilir. Her ileti, sonuna eklenecek nesnenin *mantıksal* uzunluğunu ve görelî konumunu tanımlamalıdır. Ancak, nesnenin toplam büyüklüğünü ya da kanal tarafından izin verilen büyüklük üst sınırını bilmenin mümkün olmadığı durumlarda, gönderme ileti çıkışını tasarlayın; böylece, koyma işlemi tek bir ileti koyar ve çıkış, aktarıldığı iletiye olabildiğince çok veri eklendiğinde, bir sonraki iletiyi iletim kuyruğuna yerleştirir.

Büyük iletilerle ilgili olarak bu yöntemi kullanmadan önce aşağıdaki noktaları göz önünde bulundurun:

- MCA ve ileti çıkışı bir IBM MQ kullanıcı kimliği altında çalışır. İleti çıkışı (ve dolayısıyla, kullanıcı kimliği), gönderme uçta almak ya da alıcı uçta yaratmak için nesneye erişmesi gerekir; bu durum, yalnızca nesnenin evrensel olarak erişilebilir olduğu durumlarda uygulanabilir. Bu da bir güvenlik sorunu yaratıyor.
- Sonuna kadar yığın verileri eklenmiş olan başvuru iletisi, hedefine ulaşmadan önce birkaç kuyruk yöneticisi aracılığıyla seyahat etmelidir; toplu veriler, girişimsel düğümlerdeki IBM MQ kuyruklarında bulunur. Ancak, bu durumlarda özel destek ya da çıkışlar sağlanmaz.
- Yeniden yönlendirme ya da ölü harf kuyruğa alma işlemine izin veriliyorsa, ileti çıkışınızın tasarlanması zorlaşmanızı sağlar. Bu durumlarda, nesnenin bölümleri siparişten çıkabilirler.
- Bir başvuru iletisi hedefine ulaştığında, alıcı ileti çıkışı nesneyi yaratır. Ancak, bu, MCA ' nın iş birimi ile uyumlulaştırılmaz; dolayısıyla, toplu iş yedeklenirse, nesnenin bu aynı bölümünü içeren başka bir başvuru iletisi daha sonraki bir kümeye gelecek ve ileti çıkışı, nesnenin aynı bölümünü yeniden yaratmayı deneyebilir. Nesne örneğin, bir dizi veritabanı güncelleştirmesi ise, bu kabul edilemez olabilir. Bu durumda, ileti çıkışı, güncellemelerin uygulanmış olduğu bir günlüğü tutmalıdır; bu, IBM MQ kuyruğu kullanılmasını gerektirebilir.
- Nesne tipinin özelliklerine bağlı olarak, ileti çıkışlar ve uygulamaların, kullanım sayılarının korunması konusunda işbirliği yapması gerekebilir; böylece, nesne artık gerekmediği zaman silinebilir. Bir yönetim ortamı tanıtıcısı da gerekli olabilir; başvuru iletisi üstbilgisinde bu alan için bir alan sağlanır (bkz. [MQRMH](#) ).
- Bir başvuru iletisi dağıtım listesi olarak konulursa, o düğümdeki her bir dağıtım listesi ya da her bir hedef için nesnenin yeniden alınması gerekir. Kullanım sayılarını korumanız gerekebilir. Ayrıca, bir düğümün listedeki bazı hedefler için son düğüm, ancak diğer kullanıcılar için bir ara düzey düğüm olabileceği olasılığını da göz önünde bulundurun.
- Toplu veriler genellikle dönüştürülmez. Bunun nedeni, dönüştürme *önce* ' un ileti çıkışa çağrıldığı yer almasıdır. Bu nedenle, kaynak gönderen kanalda dönüştürme isteğinde bulunulmamalıdır. Başvuru iletisi bir ara düğümden geçerse, yığın veriler istenirse ara düğümden gönderildiğinde dönüştürülür.
- Başvuru iletileri kesimlere ayırlamaz.

## MQRMH ve MQMD yapılarının kullanılması

Başvuru iletisi üstbilgisindeki alanların ve ileti tanımlayıcısının bir açıklaması için [MQRMH](#) ve [MQMD](#) başlıklı konuya bakın.

MQMD yapısında, *Format* alanını MQFMT\_REF\_MSG\_HEADER olarak ayarlayın. The MQHREF format, when requested on MQGET, is converted automatically by IBM MQ along with any bulk data that follows.

Aşağıda, MQRMH ' nin *DataLogicalOffset* ve *DataLogicalLength* alanlarının kullanımına ilişkin bir örnek yer almaktadır:

Bir uygulama koyma işlemi şu iletiyle bir başvuru iletisi gönderebilir:

- Fiziksel veri yok
- *DataLogicalLength* = 0 (bu ileti, tüm nesneyi gösterir)
- *DataLogicalOffset* = 0.



Nesnenin 70 000 bayt uzunluğunda olduğunu varsayarsak, ileti gönderme çıkışı kanal boyunca ilk 40 000 baytı aşağıdaki ileti içeren bir başvuru iletilerinde gönderir:

- MQRMH ' nin ardından 40 000 bayt fiziksel veri
- *DataLogicalLength* = 40000
- *DataLogicalOffset* = 0 (nesnenin başlangıcından).

Daha sonra, aşağıdaki ileti içeren iletim kuyruğunda 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 gönderme iletilerinde çıkışı tarafından görüldüğünde, geri kalan 30 bin bayt veri sonuna eklenir ve bu alanlara aşağıdaki alanlar ayarlanır:

- MQRMH ' nin ardından 30 bin bayt fiziksel veri
- *DataLogicalLength* = 30000
- *DataLogicalOffset* = 40000 (bu noktadan başlayarak).

MQRMHF\_SON işareti de ayarlıdır.

Başvuru iletileri kullanımı için sağlanan örnek programların bir açıklaması için bkz. "[Örnek Programların Çoklu Platformlar Üzerinde Kullanılması](#)" sayfa 1093.

## **İleti bekleniyor**

Bir programın kuyruğa bir ileti gelene kadar beklemesini istiyorsanız, MQGMO yapısının *Options* alanında MQGMO\_WEKE seçeneğini belirtin.


Belirtmek için MQGMO yapısının *WaitInterval* alanını kullanın. MQGET çağrısının bir iletilerin kuyruğa gelmesini beklemesini istediğiniz süre üst sınırı (milisaniye cinsinden).

İleti bu süre içinde gelmezse, MQGET çağrısı MQRC\_NO\_MSG\_AVAILABLE neden koduyla tamamlanır.

*WaitInterval* alanında, sabit MQWI\_UNESSINI değerini kullanarak sınırsız bekleme aralığı belirtebilirsiniz. Ancak, denetiminiz dışındaki olaylar programınızın uzun bir süre beklemesine neden olabilir, bu nedenle bu değışımezi dikkatli kullanın. IMS applications must not specify an unlimited wait interval because this would prevent the IMS system terminating. ( IMS sona erdirildiğinde, tüm bağımlı bölgelerin sona ermesini gerektirir.) Bunun yerine, IMS uygulamaları sonlu bekleme aralığını belirtebilir; daha sonra, arama tamamlandıktan sonra bir ileti alınmadan arama tamamlanırsa, bekleme seçeneğiyle başka bir MQGET çağrısı yayınlayın.

**Not:** Bir iletiyi *kaldırmak* için aynı paylaşılan kuyruktan birden fazla program bekliyorsa, gelen bir ileti tarafından yalnızca bir program etkinleşir. Ancak, bir iletiye göz atmak için birden fazla program bekliyorsa, tüm programlar etkinleştirilebilir. Daha fazla bilgi için, MQGMO' daki MQGMO yapısının *Options* alanının açıklamasına bakın.

Kuyruğun durumu ya da bekleme süresi dolmadan önce kuyruk yöneticisi değışıirse, 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 işlemi beklemeye devam eder.
-  z/OS üzerinde, bağlantı ( CICS ya da IMS uygulaması için) susturma durumuna girer ve MQGMO\_FAIL\_IF\_QUIESCING seçeneğini kullandıysanız, bekleme iptal edilir ve MQGET çağrısı, MQRC\_CONN\_QUIESCING neden koduyla tamamlanır. Bu seçenek olmadan, arama işlemi beklemeye devam eder.
- Kuyruk yöneticisi durdurulmaya zorlandıysa ya da iptal edildiyse, MQGET çağrısı MQRC\_Q\_MGR\_STOPPING ya da MQRC\_CONNECTION\_BROKEN neden koduyla tamamlanır.

- İsteklerin artık engellenebilmesi için kuyruğun öznitelikleri (ya da kuyruk adı çözümleyicilerinin bulunduğu bir kuyruk) varsa, bekleme iptal edilir ve MQGET çağrısı, MQRC\_GET\_INHIBITED neden koduyla tamamlanır.
- FORCE seçeneğinin gerekli olduğu bir şekilde kuyruğun öznitelikleri (ya da kuyruk adı çözümleyicilerinin bulunduğu bir kuyruk) değişirse, bekleme işlemi iptal edilir ve MQGET çağrısı MQRC\_OBJECT\_CHANGED neden koduyla tamamlanır.

**z/OS** If you want your application to wait on more than one queue, use the signal facility of IBM MQ for z/OS (see “Sinyalizasyon” sayfa 834 ). Bu işlemlerin gerçekleştirildiği durumlara ilgili daha fazla bilgi için bkz. [MQGMO](#).

## Sinyalizasyon

Sinyalizasyon yalnızca IBM MQ for z/OS üzerinde desteklenir.

Sinyalizasyon, işletim sisteminin bildirilmesine (ya da *sinyal*) izin vermek için MQGET çağrısındaki bir seçenektir. Kuyruğun üzerinde beklenen bir ileti geldiğinde program. This is like the *Bekle biraz*. function described in topic “İleti bekleniyor” sayfa 833 because it allows your program to continue with other work while waiting for the signal. Ancak, sinyal gönderme kullanıyorsanız, uygulama iş parçacığında serbest ve işletim sistemine güvenerek, bir ileti geldiğinde programı bildirebilirsiniz.

## Sinyal ayarlamak için

Bir sinyal ayarlamak için, MQGET çağrısında kullandığınız MQGMO yapısında aşağıdaki işlemi gerçekleştirin:

1. *Options* alanında MQGMO\_SET\_SIGNAL seçeneğini ayarlayın.
2. Set the maximum life of the signal in the *WaitInterval* field. Bu değer, IBM MQ ' un kuyruğun izlemesini istediğiniz süreyi (milisaniye olarak) belirler. Sınırsız bir yaşam belirtmek için MQWI\_UNLIMITED değerini kullanın.

**Not:** IMS uygulamalarının IMS sisteminin sona ermesini engelleyeceği için, uygulamalarının sınırsız bekleme aralığı belirtmemesi gerekir. ( IMS sona erdirildiğinde, tüm bağımlı bölgelerin sona ermesini gerektirir.) Bunun yerine, IMS uygulamaları düzenli aralıklarla ECB durumunu inceleyebilir (adım 3 'e bakın). Bir program, aynı anda birkaç kuyruk tanıtıcısı üzerinde ayarlanmış sinyallere sahip olabilir:

3. *Signal1* alanındaki *Olay Denetimi Bloğu* ' nun (ECB) adresini belirtin. Bu, sinyalinizin sonucunu size bildirir. Kuyruk kapatılıncaya kadar ECB depolaması kullanılabilir durumda kalmalıdır.

**Not:** MQGMO\_SET\_SIGNAL seçeneğini MQGMO\_WEKE seçeneğiyle kullanamazsınız.

## İleti geldiğinde

Uygun bir ileti geldiğinde, ECB ' ye bir tamamlanma kodu döndürülür.

Tamamlanma kodu aşağıdakilerden birini açıklar:

- Sinyali ayarladığınız ileti kuyruğa ulaştı. İleti, bir işaret isteyen program için ayrıldı; bu nedenle, iletiyi almak için programın bir MQGET çağrısını yayınlaması gerekir.

**Not:** Başka bir uygulama, sinyali almanızın ve başka bir MQGET çağrısının verilmesi arasındaki sürede iletiyi alabilirdi.

- Ayarladığınız bekleme aralığı süresi doldu ve sinyali ayarladığınız ileti kuyruğa ulaşmadı. IBM MQ sinyali iptal etti.
- Sinyal iptal edildi. Bu durum, örneğin kuyruk yöneticisi durursa ya da kuyruğun özniteliği değişirse, MQGET çağrılarında izin verilmez.

Kuyrukta uygun bir ileti bulunduğunda, MQGET çağrısı bir MQGET çağrısıyla aynı şekilde sinyal göndermeden tamamlanır. Ayrıca, hemen bir hata saptanırsa, arama tamamlanır ve dönüş kodları ayarlanır.

Arama kabul edildiğinde ve hemen kullanılabilir bir ileti olmadığına, diğer çalışmalarla devam edebilmesi için programa, denetim döndürülür. İleti tanımlayıcısındaki çıkış alanlarının hiçbiri ayarlanmadı, ancak **CompCode** parametresi MQCC\_UYARI olarak ayarlıdır ve **Reason** parametresi MQRC\_SIGNAL\_REQUEST\_ACCEPTED olarak ayarlıdır.

For information about what IBM MQ can return to your application when it makes an MQGET call using signaling, see [MQGET](#).

Programın, ECB 'nin gönderilmesini beklerken yapacak başka bir işi yoksa, ECB' nin aşağıdakileri kullanarak bekleyebileceği:

- For a CICS Transaction Server for z/OS program, the EXEC CICS WAIT EXTERNAL command
- Toplu ve IMS programları için, z/OS WAIT makrosu

Sinyal ayarlandığında (yani ECB henüz gönderilmediyse), kuyruğun durumu ya da kuyruk yöneticisi değişirse, aşağıdaki işlemler gerçekleşir:

- Kuyruk yöneticisi susturma durumuna girerse ve MQGMO\_FAIL\_IF\_QUIESCING seçeneğini kullandıysanız, sinyal iptal edilir. ECB, MQEC\_Q\_MGR\_QUIESCING tamamlanma kodu ile birlikte gönderilir. Bu seçenek olmadan, sinyal kalır.
- Kuyruk yöneticisi zorlamalı olarak durdurulacaksa ya da iptal edildiyse, sinyal iptal edilir. Sinyal, MQEC\_WAIT\_EVRED tamamlanma kodu ile birlikte teslim edilir.
- İsteklerin artık engellenebilmesi için kuyruğun öznitelikleri (ya da kuyruk adı çözümleyicilerinin bulunduğu bir kuyruk) engelleniyorsa, sinyal iptal edilir. Sinyal, MQEC\_WAIT\_EVRED tamamlanma kodu ile birlikte teslim edilir.

#### **Not:**

1. Birden çok program, iletiyi kaldırmak için aynı paylaşılan kuyruğa işaret verdiyse, gelen bir ileti tarafından yalnızca bir program etkinleşir. Ancak, bir iletiye göz atmak için birden fazla program bekliyorsa, tüm programlar etkinleştirilebilir. Hangi uygulamaların etkinleştirileceğine karar verilirken kuyruk yöneticisinin izlediği kurallar, bekleyen uygulamalar için aynıdır: daha fazla bilgi için, [MQGMO-Get-message options](#) içindeki MQGMO yapısının *Options* alanının açıklamasına bakın.
2. Aynı iletiyi bekleyen birden fazla MQGET çağrısı varsa, bekleme ve işaret seçenekleri karışımıyla, bekleyen her çağrı eşit olarak kabul edilir. Daha fazla bilgi için, [MQGMO-Get-message options](#) içinde MQGMO yapısının *Options* alanının açıklamasına bakın.
3. Bazı koşullar altında, hem bir MQGET çağrısı hem de bir iletiyi almak için (aynı iletinin gelmesinden kaynaklanan) bir sinyal almak için mümkündür. Başka bir deyişle, programınız başka bir MQGET çağrısını yayınlarken (sinyal teslim edildiği için), kullanılacak herhangi bir ileti olmayabilir. Programınızı bu durum için test etmek üzere tasarlayın.

For information about how to set a signal, see the description of the MQGMO\_SET\_SIGNAL option and the *Signal1* field in [Signal1](#).

### **Geri alma işlemi atlanıyor**

Bir uygulama programının MQGET çağrısında **MQGMO\_MARK\_SKIP\_BACKOUT** seçeneğini belirterek bir *MQGET-error-backout* döngüye girmesini önleyebilirsiniz.

**Not:** Yalnızca IBM MQ for z/OS üzerinde desteklenir.

Bir uygulama programı, bir iş biriminin bir parçası olarak, kuyruktan ileti almak için bir ya da daha fazla MQGET çağrısını yayınlatabilir. Uygulama programı bir hata saptarsa, iş birimini geri alabilirler. Bu işlem, o iş birimi sırasında güncellenen tüm kaynakları, çalışma birimi başlatılmadan önce bulunduğu duruma geri yükler ve MQGET çağrıları tarafından alınan iletileri yeniden yürürlüğe kaydeder.

Bu iletiler, yeniden yürürlüğe girdikten sonra, uygulama programı tarafından yayınlanan sonraki MQGET çağrılarında kullanılabilir. Birçok durumda, bu durum uygulama programı için sorun yaratmaz. Ancak, geriletme sırasında ortaya çıkan hatanın çevresini geçersiz kılamaması durumunda, kuyruktan iletinin yeniden yürürlüğe girmesi uygulama programının bir *MQGET-error-backout* döngüye girmesine neden olabilir.

Bu sorunu önlemek için, MQGET çağrısında MQGMO\_MARK\_SKIP\_BACKUT seçeneğini belirtin. Bu, uygulama tarafından başlatılan geriletme içine dahil edilmediği için MQGET isteğini işaretler; yani, yedeklenmemesi gerekir. Bu seçeneğin kullanılması, bir geri alma gerçekleştiğinde diğer kaynaklara ilişkin güncellemelerin gerektiği gibi yedekleneceği anlamına gelir, ancak işaretli iletinin yeni bir iş birimi altında alınmış gibi işlem göreceği anlamına gelir.

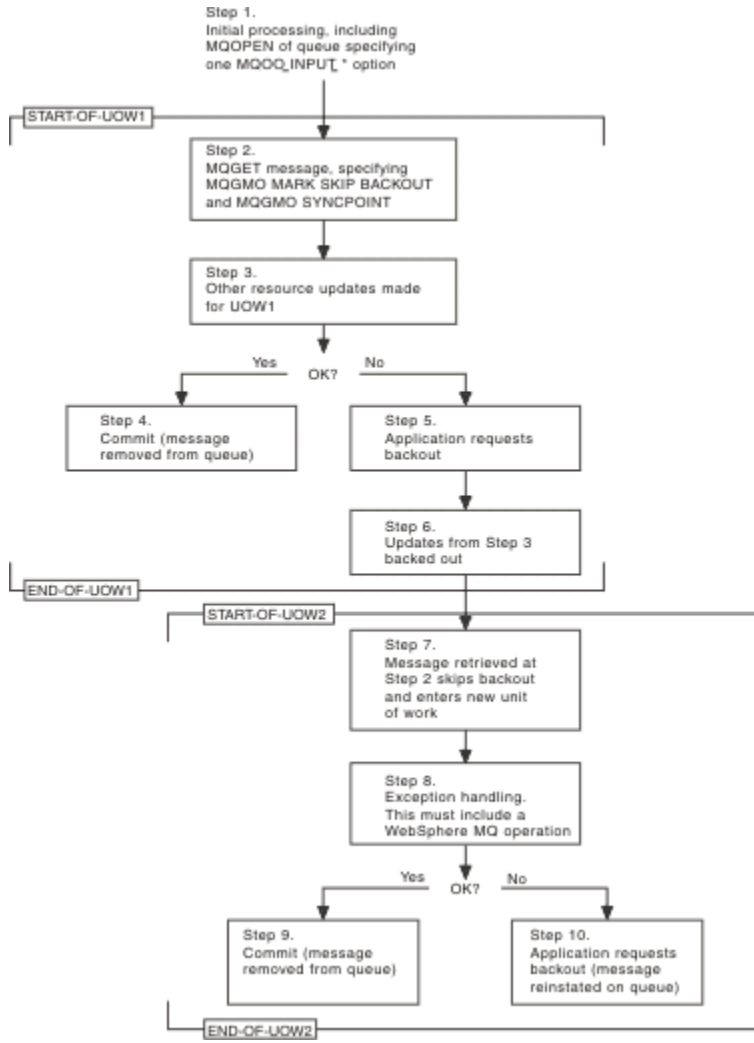
Uygulama programı, yeni iş birimini kesinleştirmek ya da yeni iş birimini yedeklemek için bir IBM MQ çağrısı yayınlamalıdır. Örneğin, program, iletiyi başlatan anda iletinin atıldığını bildiren bir kural dışı durum işleme işlemini gerçekleştirebilir ve iletiyi kuyruktan kaldırmak için iş birimini kesinleştirebilir; yeni iş birimi geriletirse (herhangi bir nedenle) ileti kuyruktan yeniden yürürlüğe girilir.

Bir iş birimi içinde, geri alma işlemi atlanıyor olarak işaretlenen yalnızca bir MQGET isteği olabilir; ancak, geri alma işlemi atlanıyor olarak işaretlenmemiş başka iletiler de olabilir. Bir ileti atlanıyor olarak imlendikten sonra, iş birimi içinde MQGMO\_MARK\_SKIP\_BACKUT değerini belirten tüm MQGET çağrıları, MQRC\_SECOND\_MARK\_NOT\_ALLOWED neden koduyla başarısız olur.

**Not:**

1. İmli ileti, yalnızca bu iletiyi içeren iş birimi, bir uygulama isteği tarafından sona erdirildiyse yedeklenir. İş birimi başka herhangi bir nedenle yedeklendiyse, ileti, geri alma işlemi atlamak için işaretlenmemiş olması durumunda, kuyruğun üzerinde aynı şekilde yedeklenir.
2. Atlama geri alma, RRS tarafından denetlenen iş birimlerine katılan Db2 saklanmış yordamlarında desteklenmez. Örneğin, MQGMO\_MARK\_SKIP\_BACKUT seçeneğiyle bir MQGET çağrısı başarısız olur ve neden kodu MQRC\_OPTION\_ENVIRONMENT\_ERROR ile başarısız olur.

Şekil 68 sayfa 837 , geri alma işlemi atlamak için bir MQGET isteği gerekirken uygulama programının içerebileceği tipik bir adım dizisi gösterir.



Şekil 68. MQGMO\_MARK\_SKIP\_BACKOUT kullanılarak geriletme atlanıyor

Şekil 68 sayfa 837 içindeki adımlar şunlardır:

### 1. Adım

İlk işlem, kuyruğun açılması için MQOPEN çağrısı da içinde olmak üzere, hareket içinde gerçekleşir (2. Adımdaki kuyruktan ileti almak için MQO\_INPUT\_\* seçeneklerinden birini belirtme).

### 2. Adım

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 68 sayfa 837 içinde bu iş birimi UOW1 olarak adlandırılır.

### Adım 3

Diğer kaynak güncellemeleri UOW1' in bir parçası olarak yapılır. Bu, MQGET çağrılarını daha da içerebilir (MQGMO\_MARK\_SKIP\_BACKOUT olmadan verilir).

### Adım 4

2. ve 3. adımlardaki tüm güncellemeler gerektiği şekilde tamamlar. Uygulama programı güncellemeleri kesinleştirir ve UOW1 sona erer. Adım 2 'de alınan ileti kuyruktan kaldırılır.

### Adım 5

2. ve 3. adımlardaki güncellemelerin bazıları gerektiği gibi tamamlanmaz. Uygulama programı, bu adımlar sırasında yapılan güncellemelerin yedekleneceğini ister.

### Adım 6

Adım 3 'te yapılan güncellemeler geriletilir.

## Adım 7

2. Adımda yapılan MQGET isteği arka arkaya atlanır ve yeni bir iş biriminin ( UOW2) bir parçası olur.

## Adım 8

UOW2 , yedeklenmekte olan UOW1 ' a yanıt olarak kural dışı durum işlemeyi gerçekleştirir. (Örneğin, başka bir kuyruğa MQPUT çağrısı, UOW1 ' in yedeklenmesine neden olan bir sorunun ortaya çıktığını gösterir.)

## Adım 9

Adım 8 gerektiği şekilde tamamlanır, uygulama programı etkinliği kesinleştirir ve UOW2 sona erer. MQGET isteği UOW2 ' nin bir parçası olduğundan (Adım 7 'ye bakın), bu kesinleştirme iletinin kuyruktan kaldırılmasına neden olur.

## Adım 10

Adım 8 gerektiği gibi tamamlanmaz ve uygulama programı UOW2' yi yedeklemektedir. İleti alma isteği UOW2 ' nin bir parçası olduğundan (bkz. Adım 7), o da geriletilir ve kuyruğun yeniden yürürlüğe girmesini sağlar. Artık, bu ya da başka bir uygulama programı tarafından yayınlanan MQGET çağrılarını (kuyrukta başka bir iletiyle aynı şekilde) başka bir uygulama programı tarafından yayınlanabilir.

## Uygulama verileri dönüştürme

Gerektiğinde, ileti tanımlayıcısı ve üstbilgi verilerini gerekli karakter kümesine ve kodlamaya dönüştüren MCA ' lar. Bağlantının her iki ucu (yani, yerel MCA ya da uzak MCA) dönüştürmeyi yapabilir.

Bir uygulama bir kuyruğa ileti yerleştirdiğinde, kuyruk yöneticileri ve MCA ' lar tarafından işlendiklerinde iletilerin denetimini kolaylaştırmak için, yerel kuyruk yöneticisi ileti tanımlayıcılara denetim bilgileri ekler. Ortama bağlı olarak, ileti üstbilgisi veri alanları, yerel sistemin karakter kümesi ve kodlamasında yaratılır.

Sistemler arasında ileti taşıdığınızda, bazen uygulama verilerini giriş sisteminin gerektirdiği karakter kümesiyle ve kodlamaya dönüştürmeniz gerekir. Bu işlem, alma sistemindeki uygulama programlarından ya da gönderme sistemindeki MCA ' lar tarafından yapılabilir. Alma sisteminde veri dönüştürme destekleniyorsa, gönderme sisteminde önceden ortaya çıkan dönüştürmeye bağlı olarak, uygulama verilerini dönüştürmek için uygulama programlarını kullanın.

Uygulama verileri bir uygulama programı içinde dönüştürülürken, MQGMO yapısının *Options* alanında MQGET çağrısına aktarılan MQGMO\_CONVERT seçeneği belirtildiğinde ve tüm aşağıdaki deyimler doğru olduğunda, uygulama verileri dönüştürülür:

- The *CodedCharSetId* or *Encoding* fields set in the MQMD structure associated with the message on the queue differ from the *CodedCharSetId* or *Encoding* fields set in the MQMD structure specified on the MQGET call.
- İletiyile ilişkilendirilen MQMD yapısındaki *Format* alanı MQFMT\_NONE değil.
- MQGET çağrısında belirtilen *BufferLength* sıfır değil.
- İleti verisi uzunluğu sıfır değil.
- Kuyruk yöneticisi, iletiyle ilişkili MQMD yapılarında belirtilen *CodedCharSetId* ve *Encoding* alanları arasındaki dönüştürmeyi destekler ve MQGET çağrısını destekler. Desteklenen kodlanmış karakter takımı tanıtıcıları ve makine kodlamalarıyla ilgili ayrıntılar için [CodedCharSetId](#) ve [Encoding](#) başlıklı konuya bakın.
- Kuyruk yöneticisi ileti biçiminin dönüştürülmesini destekler. İletiyile ilişkilendirilen MQMD yapısının *Format* alanı, yerleşik biçimlerden biriye, kuyruk yöneticisi iletiyi dönüştürebilir. *Format* yerleşik biçimlerden biri değilse, iletiyi dönüştürmek için bir veri dönüştürme çıkışı yazmanız gerekir.

MCA ' yı gönderme işlemi verileri dönüştürecekse, dönüştürmenin gerekli olduğu her bir gönderenin ya da sunucu kanalının tanımında CONVERT (YES) anahtar sözcüğünü belirleyin. Veri dönüştürme başarısız olursa, ileti gönderen kuyruk yöneticisinde DLQ ' ya gönderilir ve MQDLH yapısının *Feedback* alanı nedeni gösterir. İleti DLQ ' ya (DLQ) yerleştirilemiyorsa, kanal kapanır ve dönüştürülemez ileti iletim kuyruğunda kalır. MCA ' ları göndermek yerine, uygulamalar içinde veri dönüştürme işlemi bu durumdan kaçınıyor.

Kural olarak, yerleşik biçim ya da veri dönüştürme çıkışı tarafından *karakter* olarak tanımlanan iletiden alınan veriler, ileti tarafından istenen ileti tarafından kullanılan kodlanmış karakter kümesinden ve *sayısal* alanlar, istenen kodlamaya dönüştürülmektedir.

Yerleşik biçimler dönüştürülürken kullanılan dönüştürme işleme kurallarına ilişkin ek ayrıntılar için ve kendi veri dönüştürme çıktılarınızı yazmaya ilişkin bilgi için bkz. [“Veri dönüştürme çıktıları yazılıyor” sayfa 1014](#). Dil desteği tablolarıyla ve desteklenen makine kodlamalarıyla ilgili bilgi için ayrıca bkz. [Ulusal diller ve Makine kodlamaları](#).

## EBCDIC yeni satır karakterlerinin dönüştürülmesi

EBCDIC platformundan ASCII ' ye gönderdiğiniz verilerin yeniden geri aldığınız verilerle özdeş olduğundan emin olmanız gerekiyorsa, EBCDIC yeni satır karakterlerini dönüştürmeyi denetlemeniz gerekir.

Bunu, IBM MQ ' u değiştirmemiş dönüştürme çizelgelerini kullanmak için zorlayan, platforma bağlı bir anahtar kullanarak yapabilirsiniz, ancak sonuçlanabilen tutarsız davranışlardan haberdar olmanız gerekir.

EBCDIC yeni satır karakteri, altyapılarda ya da dönüştürme çizelgelerinde tutarlı olarak dönüştürülmediği için sorun ortaya çıkar. Sonuç olarak, veriler bir ASCII altyapısında görüntüleniyorsa, biçimlendirme yanlış olabilir. Örneğin, RUNMQSC kullanan bir ASCII altyapısından bir IBM i sistemini uzaktan denetlemek, örneğin, bu işlemi zorlaştırır.

EBCDIC biçimli verileri ASCII biçimine dönüştürmeye ilişkin ek bilgi için [Veri dönüştürme](#) başlıklı konuya bakın.

## Kuyruklardaki İletilere Göz Atma

MQGET çağrısını kullanarak kuyruklardaki iletilere göz atmaya ilişkin bilgileri bulmak için bu bilgileri kullanın.

Bir kuyruktaki iletilere göz atmak için MQGET çağrısını kullanmak için:

1. MQOO\_BROWSE seçeneğini belirterek, göz atma için kuyruğu açmak için MQOPEN ' i çağırın.
2. Kuyruktaki ilk iletiye göz atmak için, MQGET ' i MQGMO\_BROWSE\_FIRST seçeneğiyle çağırın. İstedığınız iletiyi bulmak için, birçok iletiyi adımlamak için MQGET ' i MQGMO\_BROWSE\_NEXT seçeneğiyle sürekli olarak çağırın.  
Tüm iletileri görmek için, MQMD yapısındaki *MsgId* ve *CorrelId* alanlarını boş değere (null) ayarlamanız gerekir.
3. Kuyruğu kapatmak için MQCLOSE ' yi çağırın.

### Göz at imleci

Göz atma için bir kuyruk açtığınızda (MQOPEN), arama, göz atma seçeneklerinden birini kullanan MQGET çağrılarını kullanılmak üzere bir göz atma imleçle oluşturur. Göz atma imlecini, kuyruğun ilk iletisine göre konumlandırılmış bir mantıksal gösterge olarak düşünebilirsiniz.

Aynı kuyruk için birden çok MQOL isteği yayınlayarak, birden çok göz atma imleciniz (tek bir programdan) olabilir.

Göz atma için MQGET ' i aradığınızda, MQGMO yapılarınızda aşağıdaki seçeneklerden birini kullanın:

### MQGMO\_BROWSE\_FIRST

MQMD yapılarınızda belirtilen koşulları karşılayan ilk iletinin bir kopyasını alır.

### MQGMO\_BROWSE\_NEXT

MQMD yapılarınızda belirtilen koşulları karşılayan bir sonraki iletinin bir kopyasını alır.

### MQGMO\_BROWSE\_MSG\_UNDER\_CURSOR

İmlecin şu anda gösterdiği iletinin bir kopyasını alır. Bu, en son MQGMO\_BROWSE\_FIRST ya da MQGMO\_BROWSE\_NEXT seçeneğini kullanarak alınan iletinin bir kopyasını alır.

Tüm durumlarda, ileti kuyrukta kalır.

Bir kuyruğu açtığınızda, imleç, kuyruktaki ilk iletiden hemen önce mantıksal olarak konumlandırılır. Bu, MQGET çağrısından hemen sonra MQGET çağrısını yaparsanız, ilk iletiye göz atmak için MQGMO\_BROWSE\_NEXT seçeneğini kullanabilirsiniz; MQGMO\_BROWSE\_FIRST seçeneğini kullanmanız gerekmez.

İletilerin kuyruktan kopyalandığı sıra, kuyruğun **MsgDeliverySequence** özniteliğinden belirlenir. (Daha fazla bilgi için bkz. [“İletilerin kuyruktan alınma sırası” sayfa 811.](#))

- [“FIFO 'daki \(ilk giren, ilk çıkar\) kuyruklar” sayfa 840](#)
- [“Öncelik sırasına göre kuyruklar” sayfa 840](#)
- [“Kesinleştirilmemiş iletiler” sayfa 840](#)
- [“Kuyruk sırasına değiştir” sayfa 840](#)
- [“Kuyruğun dizinini kullanma” sayfa 840](#)

## **FIFO 'daki (ilk giren, ilk çıkar) kuyruklar**

Kuyrukta kuyrukta bulunan ilk ileti kuyrukta en uzun olan iletidir.

Kuyrukta sıralı olarak iletileri okumak için MQGMO\_BROWSE\_NEXT kullanın. Bu sırayla bir kuyruk olarak, kuyruğa göz atarken, kuyrukta kuyruğa yerleştirdiğiniz iletiler de sona ermiş olur. İmleç, kuyruğun sonuna ulaştığını algıladığında, imleç bulunduğu yerde kalır ve MQRC\_NO\_MSG\_AVAILABLE ile geri döndürür. Bundan sonra iletiyi daha fazla ileti bekliyor ya da MQGMO\_BROWSE\_FIRST çağrısıyla kuyruğun başlangıcına sıfırlayabilirsiniz.

## **Öncelik sırasına göre kuyruklar**

Kuyrukta kuyrukta bulunan ilk ileti, kuyrukta en uzun süredir yer alan ve MQOPEN çağrısının verildiği sırada en yüksek önceliğe sahip olan iletidir.

Kuyruktaki iletileri okumak için MQGMO\_BROWSE\_XT seçeneğini kullanın.

Göz atma imleci, ilk iletinin önceliğiyle en düşük önceliğe kadar olan iletinin önceliğiyle çalışarak sonraki iletiyi işaret eder. Bu süre içinde, yürürlükteki göz atma imlecinin belirlediği iletiye, bu süre boyunca kuyruğa konması gereken iletiler, bu süre içinde kuyruğa konması ya da daha düşük olduğu sürece göz atabilir.

Daha yüksek önceliğe sahip kuyruğa gönderilen iletiler yalnızca aşağıdaki şekilde göz atılabilir:

- Yeniden göz atma için kuyruğu açma işlemi, yeni bir göz atma imlecinin kurulduğu noktadır.
- MQGMO\_BROWSE\_FIRST seçeneğinin kullanılması

## **Kesinleştirilmemiş iletiler**

Kesinleştirilmemiş bir ileti bir göz atma için hiçbir zaman görünür değildir; göz atma imleci geçmişteki atlamadır.

Bir iş birimi içindeki iletiler, iş birimi kesinleştirilinceye kadar göz atılamaz. İletiler, kesinleştirildiğinde kuyruklardaki konumlarını değiştirmez; bu nedenle, MQGMO\_BROWSE\_FIRST seçeneğini kullanmazsanız ve kuyruk yeniden çalışsa da, kesinleştirilmemiş iletiler görülmez, kesinleştirilmemiş iletiler *işlenmez*.

## **Kuyruk sırasına değiştir**

Kuyrukta ileti varken, ileti teslimi işlemi öncelikten FIFO 'ya çevrilirse, kuyruğa yollanan iletilerin sırası değiştirilmez. Kuyruğa daha sonra eklenen iletiler, kuyruğun varsayılan önceliğini alır.

## **Kuyruğun dizinini kullanma**

Yalnızca tek bir öncelik (kalıcı ya da kalıcı olmayan ya da her ikisi) iletileri içeren dizinlenmiş bir kuyruğa göz attığınızda, kuyruk yöneticisi belirli göz atma formlarının kullanıldığı zamanlara göz atmak için dizini kullanır.

**Not:** Yalnızca IBM MQ for z/OS üzerinde desteklenir.

Dizinlenen bir kuyruk yalnızca tek önceliğe ilişkin iletiler içerdiğinde, aşağıdaki göz atma biçimlerinden herhangi biri kullanılır:

1. Kuyruk MSGID tarafından dizinlendiyse, hedef iletiyi bulmak için izin kullanılarak MQMD yapısındaki bir MSGID geçiren isteklere göz atın.



2. Kuyruk, CORRELID tarafından dizinlendiyse, hedef iletiyi bulmak için dizin kullanılarak MQMD yapısındaki bir CORRELID geçiren isteklere göz atın.
3. Kuyruk GROUPID tarafından dizinlendiyse, hedef iletiyi bulmak için, dizin kullanılarak MQMD yapısındaki bir GROUPID geçiren isteklere göz atın.

Göz atma isteği MQMD yapısındaki bir MSGID, CORRELID ya da GROUPID iletiliyorsa, kuyruk dizinlenir ve bir ileti döndürülür, ileti için dizin girdisi bulunmalı ve bu ileti içindeki bilgiler, göz atma imlecini güncelleştirmek için kullanılır. Dizın deęerleri için geniş bir seçim kullanırsanız, bu deęer göz atma isteęine önemli bir ek işlem eklemez.

*İleti uzunluęu bilinmedięi zaman iletilere göz atmanızı saęlar*

İletin boyutunu bilmedięinizde bir iletiye göz atmak ve iletiyi bulmak için *MsgId*, *CorrelId* ya da *GroupId* alanlarını kullanmak istemiyorsanız, MQGMO\_BROWSE\_MSG\_UNDER\_CURSOR seçeneęini kullanabilirsiniz:

1. Bununla birlikte bir MQGET komutu verin:
  - MQGMO\_BROWSE\_FIRST ya da MQGMO\_BROWSE\_NEXT seçeneęi
  - MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneęi
  - Arabellek uzunluęu sıfır

**Not:** Başka bir program da aynı iletiyi alacaksa, MQGMO\_LOCK seçeneęini de kullanmayı düşünün. MQRC\_TRUNCATED\_MSG\_ACCEPTED geri döndürülemelidir.

2. Gereken depolama alanını ayırmak için döndürülen *DataLength* ' i kullanın.
3. MQGMO\_BROWSE\_MSG\_UNDER\_CURSOR ile bir MQGET yayınlayın.

Alınan ileti son olarak alındı; göz atma imleci hareket ettirilmedi. MQGMO\_LOCK seçeneęini kullanarak iletiyi kilitlemek ya da MQGMO\_UNLOCK seçeneęini kullanarak kilitli bir iletinin kilidini açmak için bu seçeneęi belirleyebilirsiniz.

Kuyruk açıldıęından, MQGMO\_BROWSE\_FIRST ya da MQGMO\_BROWSE\_NEXT seçeneklerinde MQGET işlemi yapılmazsa, çağrı başarısız olur.

*Göz attıęınız bir iletinin kaldırılması*

Kuyruktan, iletileri kaldırmak için ve göz atılmasına ilişkin kuyruęu açtıęınız için, önceden göz attıęınız bir ileti kuyruęundan kaldırabilirsiniz. (MQOO\_Browse seçeneęinin yanı sıra, MQOO\_INPUT\_ \* seçeneklerinden birini ve MQOO\_Browse seçeneęini belirtmelisiniz.)

İletiyi kaldırmak için, MQGET ' yi yeniden çağırın, ancak MQGMO yapısının *Options* alanında MQGMO\_MSG\_UNDER\_CURSOR deęerini belirtin. Bu durumda, MQGET çağırısı MQMD yapısındaki *MsgId*, *CorrelId* ve *GroupId* alanlarını yoksayar.

Göz atma ve kaldırma adımlarınız arasında, başka bir program kuyruktan iletileri kaldırmış olabilir; bu iletiler göz atma imlecinizin altındaki ileti de içinde olmak üzere, iletileri kaldırmış olabilir. Bu durumda, MQGET çağırınız, iletinin kullanılmadıęını bildiren bir neden kodu döndürür.

*Mantıksal düzende iletilere göz atma*

“Mantıksal ve fiziksel sıralama” sayfa 811 , kuyruklardaki iletilerin mantıksal ve fiziksel sırası arasındaki farkı açıklar. Bu ayırım özellikle bir kuyruęa göz atılırken önemlidir; çünkü, genel olarak, iletiler silinmez ve göz atma işlemleri kuyruęun başlangıcında başlamalarına da gerek yoktur.

If an application browses through the various messages of one group (using logical order), it is important that logical order should be followed to reach the start of the next group, because the last message of one group might occur physically *bundan sonra* the first message of the next group. MQGMO\_LOGICAL\_ORDER seçeneęi, bir kuyruk taranırken mantıksal sıraların izlenmesini saęlar.

Göz atma işlemleri için MQGMO\_ALL\_MSGS\_AVALANABILIR (ya da MQGMO\_ALL\_SEGMENTS\_AVALABILIR) kullanın. Mantıksal ileti vakasını MQGMO\_ALL\_MSGS\_AVALABILIR ile göz önünde bulundurun. Bunun etkisi, mantıksal bir iletinin yalnızca gruptaki kalan tüm iletilerin de mevcut olması durumunda kullanılabilir olması. Eęer deęillerse, mesaj

iletilir. Bu, eksik iletiler daha sonra geldiğinde, bir sonraki işlem tarafından fark edilmemeleri anlamına gelebilir.

Örneğin, aşağıdaki mantıksal iletiler mevcutsa,

```
Logical message 1 (not last) of group 123
Logical message 1 (not last) of group 456
Logical message 2 (last) of group 456
```

ve bir göz atma işlevi MQGMO\_ALL\_MSGS\_AVALABILIR ile verilir, 456 grubunun ilk mantıksal iletileri döndürülür; bu mantıksal iletiye göz atma işlevi bırakılır. 123 grubunun ikinci (son) iletileri şimdi varırsa:

```
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 gözetme işlevi yayınlansa da, bu grubun ilk iletileri önce olduğu için 123 grubunun artık tamamlanmış olduğunu fark etmemektedir.

Bazı durumlarda (örneğin, grup tümüyle yok edici olarak alındıysa), MQGMO\_ALL\_MSGS\_AVAM değerini önce MQGMO\_BROWSE\_FIRST ile birlikte kullanabilirsiniz. Tersi durumda, gözden kaçan yeni gelen iletileri not almak için taramayı yinelemeniz gerekir; MQGMO\_BROWSE\_NEXT ve MQGMO\_ALL\_MSGS\_AVAILLY ile birlikte MQGMO\_BEKE yayınının verilmesi, bu iletilerin dikkate alınmadığını gösterir. (Bu durum, iletilerin taranmasından sonra varabilecek daha yüksek öncelikli iletilere de oluşur.)

Sonraki bölümler, kesimlere ayrılmış iletiler ile ilgili göz atma örneklerine göz atmanızı sağlar; bölümlenmiş iletiler benzer ilkeleri izler.

#### *Gruplardaki iletilere göz atma*

Bu örnekte, uygulama kuyrukta, mantıksal sırada her bir iletiyle göz atılıyor.

Kuyruktaki iletiler gruplandırılmış olabilir. Gruplanmış iletiler için uygulama, içindeki tüm iletiler gelene kadar, herhangi bir grubu işlemeye başlamak istemiyor. Bu nedenle, gruptaki ilk ileti için MQGMO\_ALL\_MSGS\_AVALABILIR belirtildi; gruptaki sonraki iletiler için bu seçenek gereksiz.

Bu örnekte MQGMO\_WAN kullanıldı. However, although the wait can be satisfied if a new group arrives, for the reasons in [“Mantıksal düzende iletilere göz atma” sayfa 841](#), it is not satisfied if the browse cursor has already passed the first logical message in a group, and the remaining messages now arrive. Bununla birlikte, uygun bir aralık beklemek, yeni iletiler ya da kesimler beklenirken uygulamanın sürekli olarak döngüye girmemesini sağlar.

MQGMO\_LOGICAL\_ORDER, taramanın mantıksal sırada olduğundan emin olmak için kullanılır. Bu yıkıcı MQGET örneğiyle, her grubun kaldırıldığı için, bir gruptaki ilk (ya da tek) iletiyi ararken MQGMO\_LOGICAL\_ORDER kullanılmaz.

Uygulamanın arabelleğinin tüm iletiyi tutabilmek için her zaman yeterince büyük olduğu varsayılır. İletinin bölümlenip kesilmediği varsayılır. Bu nedenle, her MQGET üzerinde MQGMO\_COMPLE\_MSG belirtildi.

Aşağıda, bir gruptaki mantıksal iletilere göz atmanın bir örneği verilmektedir:

```
/* 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 iade edilinceye kadar yinelenir.

### İmha edici olarak göz atma ve alma

Bu örnekte uygulama, grup içindeki her bir mantıksal iletiyi, bu grubun yok edici bir şekilde almaya karar vermeden önce göz önünde bulunmaya devam eder.

Bu örneğe ilişkin ilk bölüm öncekine benzer. Ancak, bu durumda, bütün bir grubu göz altında bulunca, geri dönüp onu yok edici bir şekilde geri almaya karar veriyoruz.

Bu örnekte her grup kaldırıldığı için, bir gruptaki ilk ya da tek ileti aranırken MQGMO\_LOGICAL\_ORDER kullanılmaz.

Aşağıda bir göz atma örneği verilir ve daha sonra yok edici duruma gelir elde edin:

```
GMO.Options = MQGMO_BROWSE_NEXT | MQGMO_COMPLETE_MSG | MQGMO_LOGICAL_ORDER
              | MQGMO_ALL_MESSAGES_AVAILABLE | 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 */
        ...
```

### Browsed iletilerinin sürekli olarak sunulmasını önleme

Belirli açık seçenekleri ve ileti alma seçeneklerini kullanarak, iletileri geçerli ya da diğer işbirliği uygulamaları tarafından yeniden alınmaması için göz atıldığı gibi işaretleyebilirsiniz. İletiler, göz atma için açık bir şekilde ya da otomatik olarak yeniden kullanılabilir duruma getirmek için otomatik olarak işaretleyemez.

Bir kuyruktaki iletilere göz atsanız, onları yok edici bir şekilde elde etmeniz, bunları alabileceğiniz sıraya göre farklı bir sırayla alabilirsiniz. Özellikle, aynı iletiye birden çok kez göz atabilirsiniz; bu, kuyruktan kaldırılrsa da olanaklı değildir. Bundan kaçınmak için, iletileri göz atıkları gibi *işaretleyebilir* ve işaretli iletileri almayı önlemeniz gerekir. Bu, bazen *işaretle göz atolarak* da adlandırılır. Göz atılan iletileri işaretlemek için, MQGMO\_MARK\_BROWSE\_HANDLE ileti alma seçeneğini kullanın ve yalnızca işaretlenmemiş iletileri almak için MQGMO\_UNMARKET\_BROWSE\_MSG kullanın. MQGMO\_BROWSE\_FIRST, MQGMO\_UNMARKED\_BROWSE\_MSG ve MQGMO\_MARK\_BROWSE\_HANDLE seçeneklerini ve yinelenen MQGES komutunu kullanırsanız, kuyruktaki bulunan her iletiyi sırayla alırsınız. Bu, iletilerin atlanmamasını sağlamak için MQGMO\_BROWSE\_FIRST kullanılsa da iletilerin yinelenmesini önler. Bu seçenekler birleşimi, tek bir sabit MQGMO\_BROWSE\_HANDLE ile gösterilebilir. Kuyruğunda göz atılmamış ileti olmadığında, MQRC\_NO\_MSG\_AVAILABLE iletileri döndürülür.

Birden çok uygulama aynı kuyruğa göz atıyorsa, kuyrukları MQOO\_CO\_OP ve MQOO\_BROOK seçenekleriyle açabilir. Her bir MQOPER tarafından döndürülen nesne tanıtıcısı, işbirliği yapan grubun bir parçası olarak kabul edilir. MQGMO\_MARK\_BROWSE\_CO\_OP seçeneğini belirten bir MQGET çağrısının döndürdüğü herhangi bir ileti, bu işbirliği işlemi kümesi için işaretlendi olarak kabul edilir.

Bir ileti bir süredir imlendiyse, kuyruk yöneticisi tarafından otomatik olarak imlenemez ve göz atmak için kullanılabilir kılınabilir. Kuyruk yöneticisi özniteliği MsgMarkBrowseInterval , iş birliği tanıtıcısı olarak iş birliği yapmak için bir iletinin gösterileceği süreyi milisaniye cinsinden verir. Bir MsgMarkBrowseInterval /-1, iletilerin hiçbir zaman otomatik olarak işaretlenmediği anlamına gelir.

Tek bir işlem ya da işbirliği süreci kümesi iletileri durdururken, işaretlenen iletiler işaretsiz 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, kuyruğun MQOO\_CO\_OP ile birlikte açılmasını sağlar. Bu, dağıtıcıların işbirliği yaptığını ve birbirlerinin işaretlenen iletilerinden haberdar olacağını gösterir. Daha sonra, MQGMO\_BROWSE\_FIRST, MQGMO\_UNMARKED\_BROWSE\_MSG ve MQGMO\_MARK\_BROWSE\_CO\_OP seçeneklerini belirterek, her dağıtıcı yinelenen MQGET çağrıları yapar (bu seçenek bileşimini göstermek için tek bir sabit MQGMO\_BROWSE\_CO\_OP kullanabilirsiniz). Daha sonra, her dağıtıcı uygulaması yalnızca başka işbirliği dağıtıcıları tarafından işaretlenmemiş iletileri alır. Dağıtıcı, bir tüketici başlatır ve MQGET tarafından döndürülen MsgToken ' ı, iletiyi yok edici olarak kuyruktan alan tüketiciye iletir. Tüketici iletinin MQGET ' ini yedeklerse, bu ileti artık işaretlenmediği için, tarayıcılardan biri için yeniden gönderimi yapmak için ileti kullanılabilir. Tüketici iletide bir MQGET işlemi yapmazsa, MsgMarkBrowseInterval iletildikten sonra kuyruk yöneticisi, iş birliği yapan tutamaçlar kümesi için iletiyi işaretler ve yeniden dağıtılabılır.

Aynı dağıtıcı uygulamasının birden çok kopyası yerine, kuyruğa göz atan, kuyruklardaki iletilerin bir alt kümesini işlemek için uygun olan farklı dağıtıcı uygulamalarından birine sahip olabilirsiniz. Her dağıtıcıda, kuyruğun MQOO\_CO\_OP ile birlikte açılmasını sağlar. Bu, dağıtıcıların işbirliği yaptığını ve birbirlerinin işaretlenen iletilerinden haberdar olacağını gösterir.

- Tek bir dağıtıcıya ilişkin 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, MQGET çağrılarını yineledi. Bu dağıtıcının işlemesi için browsed iletileri uygunsa, MQMO\_Match\_MSG\_XX\_ENCODE\_CASE\_CAPS\_LOCK\_ON token, mqgmo\_mark\_browse\_co\_op ve önceki MQGET çağrısının döndürdüğü MsgToken adlı bir MQGET çağrısı yapar. Arama başarılı olursa, dağıtıcı tüketiciyi kullanıma hazırlar ve MsgToken ' ı buna 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 işleyemediği bir iletiyi göz atarsa, MQGET komutunu MQMO\_MATCH\_MSG\_TOKEN, MQGMO\_UNMARK\_BROWSE\_CO\_OP seçeneğiyle çağırarak, daha önce döndürülen MsgToken ile iletiyi işaretler.

### **MQGET çağrısının başarısız olduğu bazı durumlar**

Bir kuyruğun belirli öznitelikleri bir MQOPER ve MQGET çağrısının verilmesi arasındaki bir komutta FORCE seçeneği kullanılarak değiştirilirse, MQGET çağrısı başarısız olur ve MQRC\_OBJECT\_CHANGED neden kodunu döndürür.

Kuyruk yöneticisi, nesne tanıtıcısını artık geçerli değil olarak işaretler. Bu durum, değişikliklerin kuyruk adının çözümlendiği herhangi bir kuyruğa uygulanırsa da olur. Bu şekilde, tanıtıcıyı etkileyen öznitelikler, MQOPEN' da MQOPER çağrısının tanımında listelenir. Aramanız MQRC\_OBJECT\_CHANGED neden kodunu döndürürse, kuyruğu kapatın, yeniden açın ve bir ileti almaya yeniden çalışın.

İletileri alma girişiminde bulunduğunuz bir kuyruk için (ya da kuyruk adının çözümlendiği herhangi bir kuyruk) alma işlemleri engelleniyorsa, MQGET çağrısı başarısız olur ve MQRC\_GET\_INHIBITED neden kodunu döndürür. Bu durum, göz atma için MQGET çağrısını kullanıyor olsanız bile ortaya çıktı. Daha sonra MQGET çağrısını denerseniz başarıyla bir ileti alabilirsiniz; uygulamanın tasarımı diğer programlar kuyrukların özniteliklerini düzenli olarak değiştiriyorsa, bu iletiyi başarıyla alabilirsiniz.

Dinamik bir kuyruk (geçici ya da kalıcı) silindiyse, önceden edinilmiş bir nesne tanıtıcısı kullanılarak MQGET çağruları başarısız olur ve MQRC\_Q\_DELETED neden kodunu geri döndürür.

## **Yayınlama/abone olma uygulamaları yazılıyor**

Yayınlama/abone olma IBM MQ uygulamalarını yazmaya başlayın.

Yayınlama/abone olma kavramlarına genel bakış için [Yayınlama/abone olma ileti alışverişibaşlıklı konuya](#) bakın.

Farklı yayınlama/abone olma uygulamaları yazılmasına ilişkin bilgi edinmek için aşağıdaki konulara bakın:

- [“Yayınlayıcı uygulamaları yazılıyor” sayfa 845](#)
- [“Abone uygulamaları yazılıyor” sayfa 852](#)
- [“Yaşam çevrimlerini yayınla/abone ol” sayfa 869](#)
- [“İleti özelliklerini yayınla/abone ol” sayfa 874](#)
- [“İleti sıralaması” sayfa 875](#)
- [“Yayınlara ele geçirmesi” sayfa 876](#)
- [“Yayın seçenekleri” sayfa 883](#)
- [“Abonelik seçenekleri” sayfa 883](#)

### **İlgili kavramlar**

[“Uygulama geliştirme kavramları” sayfa 7](#)

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz.

IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.

[“IBM MQ için uygulama geliştirilmesi” sayfa 5](#)

İletileri göndermek ve almak, kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için uygulamalar geliştirebilirsiniz. IBM MQ , birçok farklı dil ve çerçeve içinde yazılmış uygulamaları destekler.

[“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#)

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğinize karar verdiğinizde, IBM MQ tarafından sunulan özelliklerin nasıl kullanılacağına karar vermeniz gerekir.

[“Kuyruğa alma için bir yordamsal uygulama yazma” sayfa 759](#)

Kuyruk yöneticisi, yayınlama/abone olma, nesnelere açma ve kapama nesnelere bağlanma ve bağlantıyı kesme, kuyruğa alma ve bağlantı kesme hakkında bilgi edinmek için bu bilgileri kullanın.

[“İstemci yordamsal uygulamaları yazılıyor” sayfa 945](#)

Bir yordamsal dil kullanarak IBM MQ üzerinde istemci uygulamaları yazmak için bilmeniz gerekenler.

[“Yordamsal uygulama oluşturulması” sayfa 1030](#)

Bir IBM MQ uygulamasını birden çok yordamsal dilden birine yazabilir ve uygulamayı farklı platformlarda çalıştırabilirsiniz.

[“Yordamsal program hatalarının işlenmesi” sayfa 1073](#)

Bu bilgiler, bir çağrı yaparken ya da ileti son hedefine teslim edildiğinde, uygulama MQI çağrılarınızla ilişkili hataları açıklar.

### **İlgili görevler**

[“IBM MQ örnek yordamsal programlarının kullanılması” sayfa 1092](#)

Bu örnek programlar yordamsal dillere yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımları gösterir. Farklı platformlardaki IBM MQ programları.

## **Yayınlayıcı uygulamaları yazılıyor**

İki örnek üzerinde çalışarak yayınlayıcı uygulamaları yazmaya başlayın. İlki, bir kuyruğa ileti yerleştirmek için bir noktadan noktaya mümkün olduğunca yakın bir şekilde modellenir ve ikinci olarak konular, yayınlayıcı uygulamaları için dinamik olarak daha yaygın bir kalıba yol açmaktadır.

Writing a simple IBM MQ publisher application is just like writing an IBM MQ point to point application that puts messages to a queue ([Çizelge 125 sayfa 846](#)). The difference is you MQPUT messages to a topic, not to a queue.

<i>Çizelge 125. Puana karşılık publish/subscreen IBM MQ program örüntüni göster.</i>		
<b>Adım</b>	<b>Nokta MQ Çağrısı</b>	<b>MQ Call 'ı Yayınla</b>
<b>Kuyruk yöneticisine bağlan</b>	MQCONN	MQCONN
<b>Kuyruğu aç</b>	MQOPEN	
<b>Konuyu aç</b>		MQOPEN
<b>İleti (lar) koy</b>	MQPUT	MQPUT
<b>Konuyu kapat</b>		MQCLOSE
<b>Kuyruğu kapat</b>	MQCLOSE	
<b>Kuyruk yöneticisinden bağlantıyı kes</b>	MQDISC	MQDISC

Bu betonu yapmak için, hisse senedi fiyatlarını yayınlatabilmek için iki uygulama örneği vardır. İlk örnekte ([“Örnek 1: Sabit bir konuya yayınlayıcı” sayfa 846](#)), bir kuyruğa ileti yerleştirerek yakından modellenen yönetici, kuyruk yaratmak için benzer bir şekilde bir konu tanımlaması yaratır. MQPUT programlayıcı kodları, iletileri bir kuyruğa yazmak yerine konuya yazmak için kodlamayı sağlar. İkinci örnekte ([“Örnek 2: Bir değişken konusuna yayınlayıcı” sayfa 849](#)), programın IBM MQ ile olan etkileşiminin örünmesi benzerdir. Fark, programcının, yöneticinin yerine, iletinin yazıldığı konuyu yönetici yerine getirmektedir. Uygulamada genellikle, konu dizgisinin bir tarayıcı aracılığıyla insan girişi gibi başka bir kaynak tarafından ya da başka bir kaynak tarafından sağlandığı anlamına gelir.

### **İlgili kavramlar**

[“Abone uygulamaları yazılıyor” sayfa 852](#)

Get started with writing subscriber applications by studying three examples: an IBM MQ application consuming messages from a queue, an application that creates a subscription and requires no knowledge of queuing, and finally an example that uses both queuing and subscriptions.

### **İlgili başvurular**

[KONUYU TANIMLA](#)

[KONUYU GÖRÜNTÜLE](#)

[TANITIM](#)

*Örnek 1: Sabit bir konuya yayınlayıcı*

Bir yönetsel olarak tanımlanmış bir konuya yayınlama göstermek için bir IBM MQ programı.

**Not:** Sıkıştırılmış kodlama biçemi, üretim kullanımına hazır olmamaya yöneliktir.

See the output in [Şekil 70 sayfa 847](#)

```
#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 69. Basit IBM MQ yayıncısı sabit bir konuya yayındır.*

```
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 70. İlk yayıncıya örneğinden alınan örnek çıktı*

Aşağıdaki seçili kod satırları, IBM MQ için bir yayıncı uygulaması yazılmasına ilişkin bakış açılarını göstermektedir.

**char topicNameDefault[] = "IBMSTOCKPRICE";**

Programda varsayılan bir konu adı tanımlanıyor. Programa ilişkin ilk bağımsız değişken olarak farklı bir konu nesnesinin adını belirterek bu değeri geçersiz kılabilirsiniz.

**MQCHAR resTopicStr[151];**

resTopicStr is pointed at by td.ResObjectString.VSPtr and is used by MQOPEN to return the resolved topic string. Make the length of resTopicStr one larger than the length passed in td.ResObjectString.VSBufSize to give space for null termination.

**memset (resTopicStr, 0, sizeof(resTopicStr));**

Initialize resTopicStr to nulls to ensure the resolved topic string returned in an MQCHARV is null terminated.

**td.ObjectType = MQOT\_TOPIC**

Yayınla/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* ' ü kullanmanız gerekir.

**strncpy(td.ObjectName, topicName, MQ\_OBJECT\_NAME\_LENGTH);**

topicName , bir konu nesnesinin adıdır; bazen de bir denetim konusu nesnesi olarak adlandırılır. In the example the topic object needs to be created beforehand, using IBM MQ Explorer or this MQSC command,

```
DEFINE TOPIC(IBMSTOCKPRICE) TOPICSTR(NYSE/IBM/PRICE) REPLACE;
```

**td.ResObjectString.VSPtr = resTopicStr;**

Çözümlenen konu dizisi, programda son printf ' da 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; örneğin, çıkış kuyruğu açmak gibi.

**pmo.Options = MQPMO\_FAIL\_IF QUIESCING | MQPMO\_RETAIN;**

Yeni abonelerin yayını almasını ve yayınlayıcıda MQPMO\_RETAIN belirtilmesini belirterek, abone başlatılmadan önce yayınlanan en son yayını alır ve ilk eşleşen yayını alır. alternatifi ise abonelerin sadece abone başladıktan sonra yayınlanan yayınlarla sağlanmasıdır. Ek olarak bir abonede, abonelikte MQSO\_NEW\_PUBLICATIONS\_ONLY belirterek alıkonan bir yayını alma seçeneği de reddedilir.

**MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);**

Add 1 to the length of the string passed to MQPUT to pass the null termination character to IBM MQ as part of the message buffer.

İlk örnek ne gösteriyor? The example imitates as closely as possible the tried and tested traditional pattern for writing point to point IBM MQ programs. IBM MQ programlama örüntüsünün önemli bir özelliği, programcının iletilerin gönderileceği yerde ilgilenmemesidir. Programcının görevi, bir kuyruk yöneticisine bağlanmaktan ve alıcılara dağıtılacak olan iletileri iletmekten geçer. Noktadan noktaya paradigmadaki, programcı yöneticinin yapılandırdığı bir kuyruğu (büyük olasılıkla bir diğer ad kuyruğu) açar. Diğer ad kuyruğu, iletileri yerel kuyruk yöneticisinde ya da uzak bir kuyruk yöneticisinde hedef kuyruğa yönlenebilir. İletiler teslim edilmeyi beklerken, kaynak ve hedef arasında bir yerde kuyruklar üzerinde depolanır.

Yayınla/abone olma örüntüsünde, kuyruk açmak yerine, programcı bir konu açar. Örneğimizde konu, yönetici tarafından bir konu dizisiyle ilişkilendirilir. Kuyruk yöneticisi, kuyrukları kullanarak yayını, yayının konu dizisiyle eşleşen abonelikleri olan yerel ya da uzak abonelere iletir. Yayınlar alıkonursa, kuyruk yöneticisi artık aboneliği olmasa da yayının en son kopyasını alıkonur. Alıkonan yayın, gelecekteki abonelere iletilebilecek şekilde kullanılabilir. Yayıncı uygulaması, yayını bir hedefe seçmede ya da yönlendirmede hiçbir rol oynamaz; görevi, yayın oluşturmak ve yönetici tarafından tanımlanan konulara yayınlar koymaktır.

Bu sabit konu örneği, birçok yayınla/abone olunan uygulamanın (atypic) atomlu bir konudur: statik. Bir sistem yöneticisinin konu dizilerini tanımlamasını ve yayınlandığı konuları değiştirmesini gerektirir.



Genel olarak yayınlama/abone olma uygulamalarının bazı ya da tüm konu ağacını tanıması gerekir. Konular sık sık değişir ya da konular fazla değişirse de, konu bileşimlerinin sayısı büyük ve bir denetimci, yayınlanmasına gerek duyacak her konu dizisi için bir konu düğümü tanımlamada çok zahmetli olur. Belki de konu dizileri yayınlandığı zaman bilinmez; bir yayıncı uygulaması, bir konu dizisi belirtmek için yayın içeriğindeki bilgileri kullanabilir ya da bir tarayıcıdan insan girişi gibi başka bir kaynaktan yayınlatabileceğiniz konu dizileriyle ilgili bilgileri olabilir. Daha dinamik yayınlama biçimleri için bir sonraki örnek, yayıncı uygulamasının bir parçası olarak konuların dinamik olarak nasıl yaratılacağı gösterilir.

Birkaç yayıncı ve aboneyi bir araya getiririz. Konuları adlandırma ve konu ağaçlarıyla düzenlemek için kuralları ya da mimariyi tasarlamak, yayınlama/abone olma çözümünün geliştirilmesinde önemli bir adımdır. Konu ağacının hangi kuruluşun yayıncı ve abone programlarını birbirine bağlayıp birleştirdiği ve bunları konu ağacının içeriğine bağlayacak ölçüde dikkatli bir şekilde bakın. Konu ağacındaki değişikliklerin yayıncıyı ve abone uygulamalarını etkilemesini ve etkiyi nasıl en aza indirebileceğinin sorusunu kendinize sorun. IBM MQ yayınlama/abone olma modelinin mimarisine yerleşik olarak, bir konunun kök kısmını ya da kök alt ağacını sağlayan bir denetim konusu nesnesi kavramıdır. Konu nesnesi, uygulama programlama ve işlemlerini basitleştiren ve bunun sonucunda maintaineteneği iyileştiren, konu ağacı yönetiminin kök kısmını tanımlama seçeneğini size sunar. Örneğin, yalıtılmış konu ağaçlarına sahip birden çok yayınlama/abone olma uygulaması konuşlandırılıyorsa, o zaman konu ağacının kök kısmını yöneterek, farklı uygulamalar tarafından benimsenen konu adlandırma kurallarında tutarlılık olmasa da, konu ağaçlarının yalıtılmasını garanti edebilirsiniz.

Uygulamada, yayıncı uygulamaları, bu örnekte olduğu gibi, yalnızca sabit konuları ve bir sonraki değişken konuları kullanarak bir yelpazeyi kapsamaya devam eder. [“Örnek 2: Bir değişken konusuna yayıncı” sayfa 849](#) , konuların ve konu dizilerinin kullanımını birleştirmeyi de gösterir.

### **İlgili kavramlar**

[“Örnek 2: Bir değişken konusuna yayıncı” sayfa 849](#)

Programsal olarak tanımlanmış bir konuya yayınlama göstermek için bir WebSphere MQ programı.

[“Abone uygulamaları yazılıyor” sayfa 852](#)

Get started with writing subscriber applications by studying three examples: an IBM MQ application consuming messages from a queue, an application that creates a subscription and requires no knowledge of queuing, and finally an example that uses both queuing and subscriptions.

*Örnek 2: Bir değişken konusuna yayıncı*

Programsal olarak tanımlanmış bir konuya yayınlama göstermek için bir WebSphere MQ programı.

**Not:** Sıkıştırılmış kodlama biçemi, üretim kullanımına hazır olmamaya yöneliktir.

See the output in Şekil 72 sayfa 850.

```
#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 71. Basit IBM MQ yayıncısı bir değişken konusuna yayındır.

```
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 72. İkinci yayıncı örneğinden alınan örnek çıktı

Bu örneğe dikkat etmek için birkaç nokta vardır.

```
char topicNameDefault[] = "STOCKS";
```

Varsayılan konu adı STOCKS , konu dizgisinin bir kısmını tanımlar. Bu adı, programa ilk bağımsız değişken olarak belirterek ya da ilk parametre olarak / belirterek konu adının kullanımını ortadan kaldırarak bu konu adını geçersiz kılabilirsiniz.

```
char topicString[101] = "IBM/PRICE";
```

IBM/PRICE , varsayılan konu dizisidir. Bu konu dizilimini, programa ikinci bağımsız değişken olarak belirterek geçersiz kılabilirsiniz.

Kuyruk yöneticisi, STOCKS konu nesnesi ( "NYSE") tarafından sağlanan konu dizisini "IBM/PRICE" programı tarafından sağlanan konu dizisiyle birleştirir ve iki konu dizisi arasına bir "/" ekler. Sonuç, çözümlenen konu dizisidir "NYSE/IBM/PRICE". Sonuçta elde edilen konu dizisi, IBMSTOCKPRICE konu nesnesinde tanımlı olan ve tam olarak aynı etkiye sahip olan dizedir.

Çözülen konu dizisiyle ilişkilendirilen denetim konusu nesnesinin, yayıncı tarafından MQOPEN ' a iletildiği gibi aynı konu nesnesi olması gerekmez. IBM MQ , çözümlenen konu dizisinde, yayıncı ile ilgili özellikleri hangi denetim konusu nesnesinin tanımladığı şekilde çalışmak için, çözümlenen konu dizisinde ağaç örtülü olarak kullanır.

Suppose there are two topic objects A and B, and A defines topic "a", and B defines topic "a/b" ( Şekil 73 sayfa 851 ). If the publisher program refers to topic object A and provides topic string "b", resolving the topic to the topic string "a/b", then the publication inherits its properties from topic object B because the topic matches the topic string "a/b" defined for B.

```
if (strcmp(argv[1],"/"))
```

argv[1] , isteğe bağlı olarak sağlanan topicName' dir. "/" bir konu adı olarak geçersiz; burada herhangi bir konu adının olmadığını ve konu dizgisinin tamamen program tarafından sağlandığı anlamına gelir. Şekil 72 sayfa 850 içindeki çıkış, program tarafından devingen olarak sağlanan tüm konu dizisini gösterir.

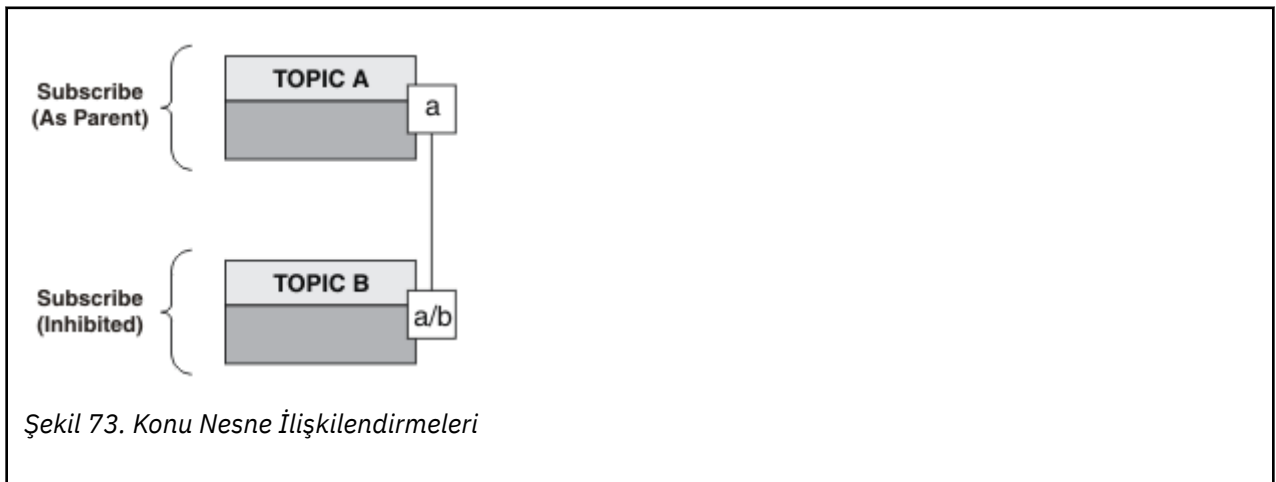
```
strncpy(td.ObjectName, topicName, MQ_OBJECT_NAME_LENGTH);
```

For the default case, the optional topicName needs to be created beforehand, using IBM MQ Explorer or this MQSC command:

```
DEFINE TOPIC(STOCKS) TOPICSTR(NYSE) REPLACE;
```

```
td.ObjectString.VSPtr = topicString;
```

Konu dizisi, konu tanımlayıcısındaki bir MQCHARV alanıdır.



İkinci örnek ne gösteriyor? her ne kadar kod ilk örneğe çok benzer olsa da-etkili bir şekilde sadece iki satır farkı vardır-sonuç, ilk olarak önemli ölçüde farklı bir programdır. Programcılar, yayınların gönderildiği hedeflerin denetlenmesini sağlar. Abone uygulamalarını tasarlamak için kullanılan en düşük yönetici girişleriyle birlikte, yayıncılardan abonelere yayınların yönlendirilmesi için hiçbir konu ya da kuyruk önceden tanımlanmaz.

Noktadan noktaya ileti sistemi paradigması içinde, iletilerin akabilmesi için önce kuyruklar tanımlanmalıdır. For publish/subscribe, they do not, although IBM MQ implements publish/subscribe using its underlying queuing system; the benefits of guaranteed delivery, transactionality and loose coupling associated with messaging and queuing are inherited by publish/subscribe applications.

Bir tasarımcı yayıncının, abonenin, programların temel konu ağacından haberdar olup olmadığına karar vermek zorundadır. Ayrıca abone programlarının kuyruğa alma ya da kuyruktan haberdar olup olmadığı da dikkate alır. Sonraki abonelerin örnek uygulamalarını araştır. Bu örnekler, genellikle NYSE/IBM/PRICE' e abone olmak ve abone olmak üzere yayıncı örnekleriyle birlikte kullanılmak üzere tasarlanmıştır.

### İlgili kavramlar

“Örnek 1: Sabit bir konuya yayıncı” sayfa 846

Bir yönetimsel olarak tanımlanmış bir konuya yayıncı göstermek için bir IBM MQ programı.

“Abone uygulamaları yazılıyor” sayfa 852

Get started with writing subscriber applications by studying three examples: an IBM MQ application consuming messages from a queue, an application that creates a subscription and requires no knowledge of queuing, and finally an example that uses both queuing and subscriptions.

### Abone uygulamaları yazılıyor

Get started with writing subscriber applications by studying three examples: an IBM MQ application consuming messages from a queue, an application that creates a subscription and requires no knowledge of queuing, and finally an example that uses both queuing and subscriptions.

Çizelge 126 sayfa 852 içinde, tüketicinin ya da abonenin üç stili, bunları karakterize eden IBM MQ işlev çağrılarını sıralarıyla birlikte listelenir.

1. İlk stil, MQ Publication Consumer (Yayın Tüketicisi), yalnızca MQGET yaptığı MQ programını noktalamak için kullanılan bir noktayla aynıdır. Uygulamanın yayıncı tüketmekte olduğu bilgisi yoktur; yalnızca kuyruktan ileti okumaktadır. The subscription that causes publications to get routed to the queue is created administratively using IBM MQ Explorer or a command.
2. İkinci biçim, çoğu abone uygulaması için tercih edilen örüntüdür. Abone uygulaması aboneliği oluşturur ve sonra yayıncı alır. Kuyruk yöneticisi, kuyruk yöneticisi tarafından gerçekleştirilir.
3. Üçüncü stilde, abone uygulaması, yayıncı doldurmak için kullanılan temel kuyruğu açmayı ve kapatmayı ve kuyruğun yayıncı ile doldurulması için abonelikler verir.

One way to understand these styles is to study the example C programs listed in Çizelge 126 sayfa 852 for each of the styles. Örnekler, “Yayıncı uygulamaları yazılıyor” sayfa 845 içinde bulunan yayıncı örneğiyle birlikte çalıştırılacak şekilde tasarlanmıştır.

Adım	MQ ileti tüketicisi	“Örnek 1: MQ Publication consumer” sayfa 853	“Örnek 2: Yönetilen MQ aboneliği” sayfa 855	“Örnek 3: Yönetilmeyen MQ aboneliği” sayfa 860
Kuyruk yöneticisine bağlan	MQCONN	MQCONN	MQCONN	MQCONN
Kuyruğu aç	MQOPEN	MQOPEN		MQOPEN
Abone Ol			MQSUB	MQSUB
İleti al	MQGet	MQGet	MQGet	MQGet
Kuyruğu kapat	MQCLOSE	MQCLOSE	(MQCLOSE)	MQCLOSE
Aboneliği kapat			MQCLOSE	MQCLOSE
Kuyruk yöneticisinden bağlantıyı kes	MQDISC	MQDISC	MQDISC	MQDISC

Kaynakları serbest bırakmak, MQCLOSE seçeneklerini ya da yalnızca MQOL ile simetri için, MQCLOSE 'nin kullanılması her zaman isteğe bağlıdır. Yönetilen MQ abonesi vakasında abonelik kuyruğu kapatıldığında ve simetri bağımsız değişkeni ilgili değilse, MQCLOSE seçeneklerini belirtme olasılığının düşük olduğu için, abonelik kuyruğu Örnek 2: Yönetilen MQ abonesi içinde belirtik olarak kapatılmaz.

Yayınlama/abone olma uygulama kalıplarını anlamamanın başka bir yolu da dahil olan farklı varlıklar arasındaki etkileşimlere çok fazla göz atmaktadır. Ömür çizgisi ya da UML sıra çizgeleri etkileşimleri incelemek için iyi bir yoldur. “Yaşam çevrimlerini yayınla/abone ol” sayfa 869 içinde üç adet ömür çizgisi örneği anlatılır.

#### *Örnek 1: MQ Publication consumer*

MQ Publication tüketicisi, konuların kendisine abone olmayan bir IBM MQ ileti tükettidir.

To create the subscription and publication queue for this example run the following commands, or define the objects using IBM MQ Explorer.

```
DEFINE QLOCAL(STOCKTICKER) REPLACE;  
DEFINE SUB(IBMSTOCKPRICESUB) DEST(STOCKTICKER) TOPICOBJ(IBMSTOCKPRICE) REPLACE;
```

The IBMSTOCKPRICESUB subscription references the IBMSTOCK topic object created for the publisher example and the local queue STOCKTICKER. The topic object IBMSTOCK defines the topic string that is used in the subscription, NYSE/IBM/PRICE. Konu nesnesinin ve yayınların alınması için kullanılan kuyruğun, abonelik yaratılmadan önce tanımlanması gerektiğini unutmayın.

MQ yayın tüketicisi örüntüleri için bir dizi değerli kategori var:

1. Çoklu işlem: Yayınların okuma yazma işlerinden paylaşılması. Yayınlar, abonelik konusuyla ilişkili tek kuyruğa gider. Multiple consumers can open the queue using MQ00\_INPUT\_SHARED.
2. Merkezi olarak yönetilen abonelikler. Uygulamalar kendi abonelik konularını ya da aboneliklerini oluşturmaz; yayınların gönderildiği yerde yönetici sorumludur.
3. Abonelik yoğunluğu: Birden çok farklı abonelik tek bir kuyruğa gönderilebilir.
4. Abonelik dayanıklılığı: Kuyruğun, tüketicilerin etkin olup olmadığı tüm yayınları alır.
5. Geçiş ve birlikte kullanım: tüketicisi kodu, bir noktadan noktaya iletişim ve yayınlama/abone olma senaryosu için eşit derecede iyi çalışır.

The subscription creates a relationship between the topic string NYSE/IBM/PRICE and the queue STOCKTICKER. Yürürlükte tutulan yayın da içinde olmak üzere, yayınlar, abonelik yaratıldığı andan itibaren STOCKTICKER 'e iletilir.

Yönetimsel olarak oluşturulan bir abonelik, yönetilebilir ya da yönetilmeyen olabilir. Yönetilen abonelik, yönetilmeyen bir abonelik gibi, yaratıldığı anda yürürlüğe girer. Tüm örüntü kategorileri yönetilen bir abonelik için kullanılabilir değil. Bkz. “Örnek 3: Yönetilmeyen MQ abonesi” sayfa 860

**Not:** Sıkıştırılmış kodlama biçemi, üretim kullanımına hazır olmamaya yöneliktir.

Sonular Őekil 75 sayfa 854iinde gsterilir.

```
#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);
}
```

Őekil 74. MQ yayın tketicisi.

```
X:\Subscribe1\Debug>Subscribe1
Optional parameter: subscriptionQueue
Waiting 10 seconds for publications from STOCKTICKER
Received publication "129"
Completion code 0 and Return code 0
```

Őekil 75. MQ yayın tketicisinin ıkıŐı

AŐaŐıdakilerin farkında olmak iin bir ift standart IBM MQ C dil programlama ipucu vardır:

### **memset(publication, 0, sizeof(publicationBuffer));**

Ensure the message has a trailing null for easy formatting using printf. Yayınlayıcı örneği, 1-strlen(publication)eklenerek MQPUT ' a geçirilen ileti arabelleğindeki sondaki boş değeri içerir. MQCHAR arabelleklerinin boş değere ayarlanması, dizgileri saklamak için arabellekleri kullanan IBM MQ C programları için iyi bir programlama stildir ve boş değer, arabelleği tam olarak dolduramayan bir karakter dizisini izlemektedir.

### **MQGET(Hconn, Hobj, &md, &gmo, sizeof(publicationBuffer)-1, publication, &messlen, &CompCode, &Reason);**

Döndürülebilmek için, ileti arabelleğinin sonunda bir boş (null) değeri, if (messlen == strlen(publication)); ' un true (doğru) olması durumunda döndürülen iletinin boş değeri olduğunu doğrulayın. Bu ipucu, önceki bir ipucunu tamamlar ve publicationBuffer ' ta, publicationiçeriğinin üzerine yazılmamış en az bir boş değer olmasını sağlar.

### **İlgili kavramlar**

#### “Örnek 2: Yönetilen MQ abonesi” sayfa 855

Yönetilen MQ abonesi, çoğu abone uygulaması için tercih edilen kalıbdır. Bu örnek, kuyrukların, konuların ya da aboneliklerin yönetimsel tanımlamasını *hayır* gerektirir.

#### “Örnek 3: Yönetilmeyen MQ abonesi” sayfa 860

Yönetilmeyen abone, önemli bir abone uygulaması sınıfıdır. Bununla birlikte, yayınların ve yayınların tüketilmesinin *denetimi* ile yayınlama/abone olma avantajlarını birleştirin. Bu örnek, abonelikleri ve kuyrukları birleştirmenin farklı yollarını gösterir.

#### “Yayınlayıcı uygulamaları yazılıyor” sayfa 845

İki örnek üzerinde çalışarak yayınlayıcı uygulamaları yazmaya başlayın. İlki, bir kuyruğa ileti yerleştirmek için bir noktadan noktaya mümkün olduğunca yakın bir şekilde modellenir ve ikinci olarak konular, yayınlayıcı uygulamaları için dinamik olarak daha yaygın bir kalıba yol açmaktadır.

#### *Örnek 2: Yönetilen MQ abonesi*

Yönetilen MQ abonesi, çoğu abone uygulaması için tercih edilen kalıbdır. Bu örnek, kuyrukların, konuların ya da aboneliklerin yönetimsel tanımlamasını *hayır* gerektirir.

Bu en basit yönetilen abone tipik olarak *dayanıklı olmayan* bir abonelik kullanır. Örnek, dayanıklı olmayan bir aboneliğe odaklanır. The subscription lasts only as long as the lifetime of the subscription handle from MQSUB. Any publications that match the topic string during the lifetime of the subscription are sent to the subscription queue (and possibly a retained publication if the flag MQSO\_NEW\_PUBLICATIONS\_ONLY is not set or defaulted, an earlier publication matching the topic string was retained, and the publication was persistent or the queue manager has not terminated, since the publication was created).

Ayrıca, bu kalıpla *dayanıklı* bir abonelik de kullanabilirsiniz. Tipik olarak, yönetilen bir kalıcı abonelik kullanılıyorsa, hata oluşmadan, bir abonelik oluşturmak yerine, güvenilirlik nedenleriyle, aboneden daha uzun süre yaşarın gerçekleştirilmesini sağlar. Yönetilen, yönetilmeyen, dayanıklı ve dayanıklı olmayan aboneliklerle ilişkilendirilmiş farklı yaşam çevrimlerine ilişkin daha fazla bilgi için ilgili konular bölümünü görür.

Sürekli abonelikler genellikle kalıcı yayınlarla ve kalıcı olmayan yayınlarla, kalıcı olmayan aboneliklerle ilişkilendirilir, ancak abonelik dayanıklılığı ile yayın sürekliliği arasında gerekli bir ilişki yoktur. Tüm kalıcılık ve dayanıklılık birleşimleri mümkündür.

Yönetilen kalıcı olmayan vaka dikkate alındığında, kuyruk yöneticisi, kuyruk kapatıldığında temizlenen ve silinen bir abonelik kuyruğu oluşturur. Bu yayınlar, kalıcı olmayan abonelik kapatıldığında kuyruktan kaldırılır.

Bu kod tarafından belirtilen yönetilen dayanıklı olmayan örüntüye ilişkin değerli iç işlev kümeleri aşağıdaki gibidir:

1. İstek üzerine abonelik: abonelik konu dizgisi dinamiktir. Uygulama çalıştırıldığında uygulama tarafından sağlanır.
2. Kendi kendini yöneten kuyruk: Abonelik kuyruğu kendi kendini tanımlıyor ve yönetmekte.
3. Otomatik olarak abonelik yaşam çevrimini yönetme: *dayanıklı olmayan* abonelikleri yalnızca abone uygulamasının süresi için var olur.

- Bir *dayanıklı* yönetilen aboneliği tanımlıyorsa, kalıcı bir abonelik kuyruğu ve yayınların etkin olmamasıyla ilgili yayınların bu abonelik kuyruğunda saklanabilmesine devam eder. Kuyruk yöneticisi, yalnızca uygulama ya da yönetici aboneliği silmeyi seçtikten sonra, kuyruğu siler (ve alınmamış tüm yayınları temizler). The subscription can be deleted using an administrative command, or by closing the subscription with the MQCO\_REMOVE\_SUB option.
  - Consider setting SubExpiry for durable subscriptions so that publications cease to be sent to the queue and the subscriber can consume any remaining publications before removing the subscription and causing the queue manager to delete the queue and any remaining publications on it.
4. Esnek konu dizesi devreye alımı: Abonelik konu yönetimi, yönetimsel olarak tanımlanmış bir konu kullanılarak aboneliğin kök kısmı tanımlanarak basitleştirilmiştir. Daha sonra, konu ağacının kök kısmı uygulamadan gizlenir. Bir uygulamanın kök kısmı gizleyerek, uygulama yanlılıkla başka bir konu ağacı ya da başka bir uygulama tarafından oluşturulan başka bir konu ağacına çakılan bir konu ağacı oluşturmadan konuşturulabilir.
5. Denetlenen konular: birinci kısmın bir yönetimsel olarak tanımlanmış bir konu nesnesiyle eşleşmesinin, yayınların konu nesnesinin özniteliklerine göre yönetilmesiyle ilgili bir konu dizgisi kullanılarak yapılır.
- Örneğin, konu dizgisinin birinci kısmı, kümelenmiş bir konu nesnesiyle ilişkilendirilmiş konu dizgisiyle eşleşirse, abonelik, kümenin diğer üyelerinden yayınları alabilir.
  - Yönetimsel olarak tanımlanmış konu nesnelere ve programsal olarak tanımlanmış aboneliklere ilişkin seçici eşleştirme, her ikisinin de avantajlarını birleştirir ve sağlar. Sistem yöneticisi konular için öznitelikler sağlar ve programcı konuların yönetimiyle ilgili kaygılanmadan alt konuları dinamik olarak tanımlar.
  - Bu dize, konu ile ilişkili öznitelikleri sağlayan konu nesnesini eşleştirmek için kullanılan ve sd.Objectname içinde yer alan konu nesnesinin, genellikle bir ve aynı olmak üzere dışarı çıkmasına rağmen, sonuçtaki konu dizilimidir. Bkz. [“Örnek 2: Bir değişken konusuna yayınlayıcı” sayfa 849.](#)

Örneğin aboneliği kalıcı hale getirerek, abonenin aboneliği MQCO\_KEEP\_SUB seçeneğiyle kapattıktan sonra yayınların abonelik kuyruğuna gönderilmeye devam etmesi gerekir. Abone etkin olmadığı zaman kuyruk yayınları almaya devam eder. You can override this behavior by creating the subscription with the MQSO\_PUBLICATIONS\_ON\_REQUEST option and using MQSUBRQ to request the retained publication.

Abonelik, daha sonra MQCO\_RESUME seçeneği ile birlikte abonelik açılarak sürdürülür.

You can use the queue handle, Hobj, returned by MQSUB in a number of ways. Kuyruk tanıtıcısı, örnek olarak, abonelik kuyruğunun adını sorgulamak için kullanılır. Yönetilen kuyruklar, SYSTEM.NDURABLE.MODEL.QUEUE ya da SYSTEM.DURABLE.MODEL.QUEUE varsayılan model kuyrukları kullanılarak açılmıştır. Abonelik ilişkili konu nesnesinin özellikleri olarak, konu temelinde bir konuda kendi dayanıklı ve kalıcı olmayan model kuyruklarınızı belirterek varsayılan değerleri geçersiz kılabilirsiniz.

Model kuyruklarından devralınan özniteliklerden bağımsız olarak, ek abonelik yaratmak için yönetilen bir kuyruk tanıtıcısını yeniden kullanamazsınız. Ayrıca, yönetilen kuyruğu, döndürülen kuyruk adını kullanarak ikinci kez açarak, yönetilen kuyruk için başka bir tanıtıcı elde edebilirsiniz. Kuyruk, dışlayıcı giriş için açılmış gibi işlev görür.

Yönetilmeyen kuyruklar, yönetilen kuyruklardan daha esneklerdir. Örneğin, yönetilmeyen kuyrukları paylaşabilir ya da bir kuyrukte birden çok abonelik tanımlayabilirsiniz. Sonraki örnek, abonelikleri yönetilmeyen bir abonelik kuyruğuyla nasıl birleştirileceğini gösterir.

**Not:** Sıkıştırılmış kodlama biçemi, üretim kullanımına hazır olmamaya yöneliktir.



Sonuçlar Şekil 78 sayfa 858’inde 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 76. Yönetilen MQ aboneliği-bölüm 1: bildirimler ve parametre işleme.

Bu örnekteki bildirimlere ilişkin olarak bazı ek açıklamalar da vardır.

#### **MQHOBJ Hobj = MQHO\_NONE;**

Yayınları almak için, kalıcı olmayan bir yönetilen abonelik kuyruğunu belirttik olarak açamazsınız; ancak, kuyruk yöneticisinin sizin için kuyruğu açtığı anda döndürecek nesne tanıtıcısı için saklama alanı ayırmanız gerekir. Tanıtıcıyı MQHO\_OBJECT' ta başlatmak için önemlidir. Kuyruk yöneticisine, abonelik kuyruğuna bir kuyruk tanıtıcısı döndürmesi gerektiğini gösterir.

#### **MQSD sd = {MQSD\_DEFAULT};**

MQSUB içinde kullanılan yeni abonelik tanımlayıcısı.

#### **MQCHAR48 qName;**

Örneğin, abonelik kuyruğunun bilgisini gerektirmese de, örnek abonelik kuyruğunun adını sorgulasa da, MQINQ bağ tanımı C dilinde biraz garip olabilir, bu nedenle örneğin, çalışmanın yararlı olduğu bu bölümü 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 77. 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 78. MQ abonesi

Bu örnekteki kod hakkında yapılacak ek açıklamalar da vardır.

**strncpy(sd.ObjectName, topicName, MQ\_Q\_NAME\_LENGTH);**

topicName boş değerli ya da boş (*varsayılan değer*)ise, çözümlenen konu dizesini hesaplamak için konu adı kullanılmaz.

**sd.ObjectString.VSPtr = topicString;**

Önceden tanımlanmış bir konu nesnesini kullanmak yerine, bu örnekte programcı, MQSUB ile birleştirilen bir konu nesnesi ve bir konu dizgisi sağlar. Konu dizgisinin bir MQCHARV yapısı olduğunu fark edin.

**sd.ObjectString.VSLength = MQVS\_NULL\_TERMINATED;**

Bir MQCHARV alanının uzunluğunu ayarlamaya alternatif bir seçenek.

**sd.Options = MQSO\_CREATE | MQSO\_MANAGED | MQSO\_NON\_DURABLE | MQSO\_FAIL\_IF QUIESCING;**

Konu dizesini tanımladıktan sonra, sd.Options işaretlerinin en dikkat çekmesine gerek vardır. Birçok seçenek vardır; örneğin, yalnızca en yaygın olarak kullanılan seçenekleri belirtir. Diğer seçenekler varsayılan değerleri kullanır.

1. Abonelik *dayanıklı olmayan*ise, bu, uygulamada açık aboneliğin bir ömür sürmesine sahip, MQSO\_CREATE işaretini ayarladı. You can also set the (*varsayılan*) MQSO\_NON\_DURABLE flag for readability.
2. Complemening MQSO\_CREATE , MQSO\_RESUME. Her iki işaret birlikte ayarlanabilir; kuyruk yöneticisi yeni bir abonelik yaratır ya da var olan bir aboneliğe devam eder; hangisi uygunsa. However, if you do specify MQSO\_RESUME you must also initialize the MQCHARV structure for sd.SubName, even if there is no subscription to resume. Failure to initialize SubName results in a return code of 2440: MQRC\_SUB\_NAME\_ERROR from MQSUB.

**Not:** MQSO\_RESUME is always ignored for a non-durable managed subscription: but specifying it without initializing the MQCHARV structure for sd.SubName does cause the error.

3. Buna ek olarak, aboneliğin nasıl açılacağını etkileyen üçüncü bir işaret de vardır, MQSO\_ALTER. Doğru izinleri göz önüne alındığında, devam ettiren bir aboneliğin özellikleri, MQSUB' ta belirtilen diğer özniteliklerle eşleşecek şekilde değiştirilir.

**Not:** MQSO\_CREATE, MQSO\_RESUME ve MQSO\_ALTER işaretlerinden en az birinin belirtilmesi gerekir. Bkz. Seçenekler (MQUZE). “Örnek 3: Yönetilmeyen MQ abonesi” sayfa 860' ta üç işaretin tümünü kullanmanın örnekleri vardır.

4. Otomatik olarak sizin için aboneliği yönetmek üzere kuyruk yöneticisi için MQSO\_MANAGED ' yi ayarlayın.

**sd.ObjectString.VSLength = MQVS\_NULL\_TERMINATED;**

İsteğe bağlı olarak, boş sonlandırılmış dizgiler için MQCHARV uzunluğunun ayarını 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 döndürmesi için IBM MQ için MQCHARV ResObjectString değerini ayarlayın.

**Not:** resTopicStringBuffer, memset(resTopicStr, 0, sizeof(resTopicStrBuffer))' ta boş değerler (null) olarak kullanıma hazırlandı. Döndürülen konu dizgileri sondaki boş değerle bitmiyor.

**sd.ResObjectString.VSBufSize = sizeof(resTopicStrBuffer)-1;**

sd.ResObjectString ' in arabellek büyüklüğünü, gerçek büyüklüğünden daha az bir değere ayarlayın. Bu, çözümlenen konu dizesinin tüm arabelleği dolduracağı durumlarda, sağlanan boş değerli sonlandırıcının üzerine yazılmasını önler.

**Not:** Konu dizgisi sizeof(resTopicStrBuffer) -1' den uzunsa, hata döndürülmez. Even if VSLength > VSBufSize the length returned in sd.ResObjectString.VSLength is the length of the complete string and not necessarily the length of the returned string. Konu dizgisinin tamamlandığını doğrulamak için sd.ResObjectString.VSLength < sd.ResObjectString.VSBufSize sınamasını test edin.

### **MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);**

MQSUB işlevi bir abonelik yaratır. If it is non-durable you are probably not interested in its name, though you can inspect its status in IBM MQ Explorer. sd . SubName parametresini giriş olarak sağlayabilirsiniz, bu nedenle hangi adı arayabileceğini bilirsiniz; diğer aboneliklerle ad çakışmalarını önlemeniz gerekir.

### **MQCLOSE(Hconn, &Hsub, MQCO\_REMOVE\_SUB, &CompCode, &Reason);**

Hem abonelik, hem de abonelik kuyruğu kapatılıyor. Örnekte, abonelik kapatılır, ancak kuyruk kapatılmaz. Abonelik, kalıcı olmayan bir abonelik olduğunda, bu durumda varsayılan olarak MQCLOSE MQCO\_REMOVE\_SUB seçeneği varsayılan değerdir. MQCO\_KEEP\_SUB kullanımını bir hatadır.

**Not:** the subscription *kuyruk* is not closed by MQSUB, and its handle, Hobj, remains valid until the queue is closed by MQCLOSE or MQDISC. Uygulama zamanından önce sona erdirilirse, kuyruk ve abonelik, uygulama sona erdirildikten sonra kuyruk yöneticisi tarafından temizlenir.

### **İlgili kavramlar**

[“Örnek 1: MQ Publication consumer” sayfa 853](#)

MQ Publication tüketicisi, konuların kendisine abone olmayan bir IBM MQ ileti tükettidir.

[“Örnek 3: Yönetilmeyen MQ abonesi” sayfa 860](#)

Yönetilmeyen abone, önemli bir abone uygulaması sınıfıdır. Bununla birlikte, yayınların ve yayınların tüketilmesinin *denetimi* ile yayınlama/abone olma avantajlarını birleştirin. Bu örnek, abonelikleri ve kuyrukları birleştirmenin farklı yollarını gösterir.

[“Yayıncı uygulamaları yazılıyor” sayfa 845](#)

İki örnek üzerinde çalışarak yayıncı uygulamaları yazmaya başlayın. İlki, bir kuyruğa ileti yerleştirmek için bir noktadan noktaya mümkün olduğunca yakın bir şekilde modellenir ve ikinci olarak konular, yayıncı uygulamaları için dinamik olarak daha yaygın bir kalıba yol açmaktadır.

*Örnek 3: Yönetilmeyen MQ abonesi*

Yönetilmeyen abone, önemli bir abone uygulaması sınıfıdır. Bununla birlikte, yayınların ve yayınların tüketilmesinin *denetimi* ile yayınlama/abone olma avantajlarını birleştirin. Bu örnek, abonelikleri ve kuyrukları birleştirmenin farklı yollarını gösterir.

The unmanaged pattern is more commonly associated with *dayanıklı* subscriptions than *dayanıklı olmayan*. Tipik olarak, yönetilmeyen bir abone tarafından yaratılan bir aboneliğin yaşam çevrimi, abone olan uygulamanın yaşam çevriminden bağımsızdır. Abonelik dayanıklı hale getirilerek, abonelik etkin olmadığına da abonelik alır.

Aynı sonucu elde etmek için dayanıklı *yönetilen* abonelikler oluşturabilirsiniz, ancak bazı uygulamalar, yönetilen abonelikle mümkün olan kuyruklar ve iletiler üzerinde daha fazla esneklik ve denetim gerektirir. Kalıcı olarak yönetilen bir abonelik için, kuyruk yöneticisi, abonelik konularıyla eşleşen yayınlar için kalıcı bir kuyruk yaratır. Abonelik silindiğinde kuyruğun ve ilişkili yayınların silinmesine neden olur.

Genellikle dayanıklı *yönetilen* abonelikler, uygulamanın yaşam çevrimi ve abonelik temelde aynıysa, ancak garanti vermek için çok zorsa kullanılır. Aboneliği kalıcı hale getirerek ve paylaşılan abonelikleri kullanarak, aboneliği paylaşan her uygulama aynı yönetilen kuyruğu açar ve iletileri bundan alır.

A *yönetilen* subscription is one where IBM MQ handles the subscription and does the registering and de-registering for you, whereas, in an *yönetilmeyen* subscription, the application is responsible for specifying the queue where the subscriptions are stored.

Kuyruk yöneticisi, bir abone için kalıcı olarak yönetilen abonelik kuyruğunu örtük olarak açar; böyle bir yolla, kuyruğun işlenmesi olanaklı değildir. Buna ek olarak, her yönetilen kuyruk için birden çok abonelik yaratamazsınız ve kuyrukların adları üzerinde daha az denetime sahip olduğunuz için kuyrukları yönetmek için daha zor bulabilirsiniz. Bu nedenlerden dolayı, *yönetilmeyen* MQ abonesinin, *yönetilen* MQ abonesinden dayanıklı abonelikler gerektiren uygulamalar için daha uygun olup olmadığını göz önünde bulundurun.

[Şekil 81 sayfa 866](#) içindeki kod, yönetilmeyen bir kalıcı abonelik örüntüsünü gösterir. Şekil için, kodun yönetilmeyen, kalıcı olmayan abonelikleri de yaratılmasına neden olur. Bu örnek, aşağıdaki örüntü kategorilerini gösterir:

- İstek üzerine abonelikler: abonelik konu dizgileri dinamikdir. Bunlar, uygulama çalıştırıldığında uygulama tarafından sağlanır.
- Basitleştirilmiş abonelik konu yönetimi: abonelik konusu yönetimi, bir yönetimsel olarak tanımlanmış bir konu kullanılarak abonelik konu dizgisinin kök kısmı tanımlanarak basitleştirilmiştir. Bu, uygulamadaki konu ağacının kök kısmını gizler. Bir abonenin kök kısmını gizleyerek farklı konu ağaçlarına konuşlandırılabilir.
- Esnek abonelik yönetimi: Bir aboneliği yönetimsel olarak tanımlayabilir ya da bir abone programında isteğe bağlı olarak yaratabilirsiniz. Aboneliğin nasıl oluşturulduğunu gösteren bir öznitelik dışında, yönetimsel olarak programlı olarak abonelikler arasında bir fark yoktur. Aboneliklerin dağıtılması için kuyruk yöneticisi tarafından otomatik olarak oluşturulan üçüncü bir abonelik tipi vardır. Tüm abonelikler IBM MQ Gezgini 'nde görüntülenir.
- Kuyruklara sahip aboneliklerin esnek ilişkilendirmesi: Önceden tanımlanmış bir yerel kuyruk, MQSUB işlevi tarafından bir abonelik ile ilişkilendirilir. Abonelikleri kuyruklarla ilişkilendirmek için MQSUB ' yi kullanmanın farklı yolları vardır:
  - Var olan abonelikleri yok olan bir kuyrukla ilişkilendirmeyi ilişkilendirin MQSO\_CREATE + (Hobj from MQOPEN).
  - Bir yeni aboneliğini, var olan aboneliklere sahip bir kuyrukla ilişkilendirin, MQSO\_CREATE + (Hobj from MQOPEN).
  - Var olan bir aboneliğin farklı bir kuyruğa taşınması, MQSO\_ALTER + (Hobj from MQOPEN).
  - Resume an existing subscription associated with an existing queue, MQSO\_RESUME + (Hobj = MQHO\_NONE), or MQSO\_RESUME + (Hobj = from MQOPEN of queue with existing subscription).
  - By combining MQSO\_CREATE | MQSO\_RESUME | MQSO\_ALTER in different combinations, you can cater for different input states of the subscription and the queue without having to code multiple versions of MQSUB with different sd.Options values.
  - Alternatively, by coding a specific choice of MQSO\_CREATE | MQSO\_RESUME | MQSO\_ALTER the queue manager returns an error ( Çizelge 127 sayfa 862 ) if the states of the subscription and queue provided as input to MQSUB are inconsistent with the value of sd.Options. Şekil 87 sayfa 869 shows the results of issuing MQSUB for Subscription X with different individual settings of the sd.Options flag, and passing it three different object handles.

Bu farklı tür hatalara aşına olmak için, Şekil 80 sayfa 865 içindeki örnek program için farklı girişleri keşfedin. Çizelgede listelenen vakalara dahil olmayan RC = 2440 ortak bir hata, bir abonelik adı hatasıdır. MQSO\_RESUME ya da MQSO\_ALTER ile boş değerli ya da geçersiz bir abonelik adı geçirilerek bu neden kaynaklanır.

- Çoklu işlem: Yayınların okuması için birçok tüketicinin arasında paylaşımına sahip olabilirsiniz. Yayınlar, abonelik konusuyla ilişkili tek kuyruğa gider. Consumers have a choice of opening the queue directly using MQOPEN or resuming the subscription using MQSUB.
- Abonelik yoğunluğu: Aynı kuyruğun birden çok aboneliği yaratılabilir. Çakışan aboneliklere yol açabileceği ve aynı yayını birden çok kez alan bu yetenek için dikkatli olun. MQSO\_GROUP\_SUB seçeneği, çakışan aboneliklerin neden olduğu yinelenen yayınları ortadan kaldırır.
- abone ve tüketici ayrımı: örneklerde resimli üç tüketici modeli ile bir diğer model de tüketiciyi aboneden ayırmak. Yönetilmeyen MQ Abonesi 'nin bir varyasyonu, aynı programda MQOPEN ve MQSUB , yayınlara abone olan bir program ve başka bir program bunları tüketir. Örneğin, abone bir yayınlama/abone olma kümesinin bir parçası olabilir ve kuyruk yöneticisi kümesi dışında bir kuyruk yöneticisine bağlı tüketici olabilir. Tüketici, abonelik kuyruğunu uzak kuyruk tanımlaması olarak tanımlayarak standart dağıtılmış kuyruklama yoluyla yayınlar alır.

Özellikle de bu seçeneklerin birleşimlerini kullanarak kodunuzu basitleştirmeyi planlıyorsanız, MQSO\_CREATE | MQSO\_RESUME | MQSO\_ALTER ' un davranışını anlamak önemlidir. Study the table Çizelge 127 sayfa 862 that shows the results of passing different queue handles to MQSUB, and the results of running the example program shown in Şekil 82 sayfa 867 to Şekil 87 sayfa 869.

Tabloyu oluşturmak için kullanılan senaryoda bir abonelik X ve iki kuyruk, A ve B bulunur. `sd.SubName` abonelik adı parametresi X olarak ayarlandı, Akuyruğuna eklenmiş bir aboneliğin adı. B kuyruğu, bu kuyruğa bağlı bir aboneliğe sahip değil.

In Çizelge 127 sayfa 862, `MQSUB` is passed subscription X and the queue handle to queue A. Abonelik seçenekleri ile ilgili sonuçlar aşağıdaki gibidir:

- Kuyruk tanıtıcısı, zaten X aboneliği olan A kuyruğuna karşılık geldiği için `MQSO_CREATE` başarısız oluyor. Başarılı çağrıya karşılık bu davranışı karşılaştırın. That call succeeds because queue B does not have a subscription to X attached to it.
- `MQSO_RESUME` succeeds because the queue handle corresponds to the queue A which already has a subscription to X. In contrast, the call fails where the subscription X does not exist on queue A.
- `MQSO_ALTER`, abonelik ve kuyruk açılmasına ilişkin olarak `MQSO_RESUME` 'e benzer bir şekilde hareket eder. However if the attributes contained within the subscription descriptor passed to `MQSUB` differ from the attributes of the subscription, `MQSO_RESUME` fails, whereas `MQSO_ALTER` succeeds as long as the program instance has permission to alter the attributes. Bir abonelikte konu dizisini hiçbir zaman değiştiremeyeceğiniz, ancak bir hata döndürmektense, `MQSUB`, abonelik tanımlayıcısındaki konu adı ve konu dizisi değerlerini yoksayar ve var olan abonelikte değerleri kullanır.

Daha sonra, Çizelge 127 sayfa 862 'a bakın; burada `MQSUB`, XX aboneliği geçirilir ve kuyruk B' ye ilişkin kuyruk sapmasını içerir. Abonelik seçenekleri ile ilgili sonuçlar aşağıdaki gibidir:

- `MQSO_CREATE` succeeds and creates subscription X on queue B because this is a new subscription on queue B.
- `MQSO_RESUME` başarısız olur. `MQSUB`, B kuyruğunda X aboneliği olup olmadığını arar ve onu bulmaz, ancak `RC = 2428-subscription X 'in yokdöndürülmesi yerine, RC = 2019-Abonelik kuyruğu kuyruk nesne tanıtıcısı ile eşleşmezdeğerini döndürür.` `MQSO_ALTER` üçüncü seçeneğindeki davranış, bu beklenmeyen hatanın nedenini belirtir. `MQSUB`, kuyruk sapının bir aboneliği olan bir kuyruğu göstermesini bekler. `sd.SubName` içinde belirtilen aboneliğin var olup olmadığını denetlemeden önce bunu denetler.
- `MQSO_ALTER` succeeds, and moves the subscription from queue A to queue B.

A case that is not shown in the table is if the subscription name of the subscription on queue A does not match the subscription name in `sd.SubName`. Bu arama, `RC = 2428-Kuyruk A 'da abonelik X yokile` başarısız olur.

Çizelge 127. Farklı kuyruk tanıtıcıları ve abonelik birleşimleriyle <code>MQSUB</code> 'daki hatalar		
Kuyruk tanıtıcıları	Kuyruk A Abonelik X Kuyruk B Abonelik yok	Kuyruk A Abonelik yok Kuyruk B Abonelik yok
Hobj for Kuyruk A , <code>MQSUB</code> 'ye geçti	<p><b>MQSO_CREATE</b> RC = 2432-Abonelik X, Kuyruk A ' da zaten var</p> <p><b>MQSO_RESUME</b> Kuyruk A üzerindeki X aboneliğini sürdürür</p> <p><b>MQSO_ALTER</b> Kuyruk A 'daki X aboneliğini sürdürür ve izin verilen değişiklikleri yapar</p>	<p><b>MQSO_CREATE</b> Kuyruk A ' da abonelik X 'i yaratır</p> <p><b>MQSO_RESUME</b> RC = 2428-Kuyruk A ' da Abonelik X yok</p> <p><b>MQSO_ALTER</b> RC = 2428-Kuyruk A ' da Abonelik X yok</p>

Çizelge 127. Farklı kuyruk tanıtıcıları ve abonelik birleşimleriyle MQSUB 'daki hatalar (devamı var)

Kuyruk tanıtıcıları	Kuyruk A Abonelik X Kuyruk B Abonelik yok	Kuyruk A Abonelik yok Kuyruk B Abonelik yok
<b>Hobj for Kuyruk B , MQSUB 'ye geçti</b>	<p><b>MQSO_CREATE</b> Kuyruk B 'de yeni abonelik X yaratır</p> <p><b>MQSO_RESUME</b> RC = 2019-Abonelik kuyruğu, kuyruk nesnesi tanıtıcısı ile eşleşmiyor</p> <p><b>MQSO_ALTER</b> X aboneliği X 'i Kuyruk A 'dan Kuyruk B 'ye taşı</p>	<p><b>MQSO_CREATE</b> Kuyruk B 'de yeni abonelik X yaratır</p> <p><b>MQSO_RESUME</b> RC = 2428-Kuyruk B 'de abonelik X yok</p> <p><b>MQSO_ALTER</b> RC = 2428-Kuyruk B 'de abonelik X yok</p>
<b>MQHO_NONE, MQSUB 'a geçti</b>	<p><b>MQSO_CREATE</b> RC = 2019-Hatalı nesne tanıtıcısı: yönetilen abonelik yaratmak ve yönetilen bir kuyruk yaratmak için MQSO_MANAGED işaretini ayarlayın</p> <p><b>MQSO_RESUME</b> Kuyruk A 'daki X aboneliğini sürdürür ve Hobj 'ı Kuyruk A' ya döndürür</p> <p><b>MQSO_ALTER</b> A Kuyruğu A 'da abonelik X 'i sürdürür, Hobj 'ı Kuyruk A' ya döndürür ve izin verilen değişiklikleri yapar</p>	<p><b>MQSO_CREATE</b> RC = 2019-Hatalı nesne tanıtıcısı: yönetilen abonelik yaratmak ve yönetilen bir kuyruk yaratmak için MQSO_MANAGED işaretini ayarlayın</p> <p><b>MQSO_RESUME</b> Dönüş kodu = 2428-X aboneliği yok</p> <p><b>MQSO_ALTER</b> RC = 2019-Hatalı nesne tanıtıcısı: Kuyruk A ya da B yok</p>

**Not:** Sıkıştırılmış kodlama biçemi, üretim kullanımına hazır olmamaya yöneliktir.

```

#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 79. Yönetilmeyen MQ abonesi-kısım 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 80. Yönetilmeyen MQ aboneliği - bölüm 2: parametre işleme.

Bu örnekteki parametre işleme ile ilgili ek açıklamalar aşağıdaki gibidir:

### **switch((argv[5][0]))**

Örnekte varsayılan olarak kullanılan MQSUB seçeneği ayarının geçersiz kılınmanın etkisini sınamak için, 5. parametredeki **A lter** | **C reate** | **R esume** 'e giriş seçeneğiniz vardır. Örnek tarafından kullanılan varsayılan ayar **MQSO\_CREATE** | **MQSO\_RESUME** | **MQSO\_DURABLE**' dir.

**Not:** Setting **MQSO\_ALTER** or **MQSO\_RESUME** without setting **MQSO\_DURABLE** is an error, and **sd.SubName** must be set and refer to a subscription that can be resumed or altered.

### **\*subscriptionQueue = '\0';**

### **sdOptions = sdOptions + MQSO\_MANAGED;**

Varsayılan abonelik kuyruğu, STOCKTICKER değeri **MQSO\_CREATE** olduğu sürece boş değer dizgisiyle değiştirilirse, örnek **MQSO\_MANAGED** işaretini ayarlar ve dinamik bir abonelik kuyruğu oluşturur. **Alter** or **Resume** beşinci değiştirmede ayarlandıysa, örneğin davranışı **subscriptionName** değerine bağlıdır.

```
*subscriptionName = '\0';
```

```
sdOptions = sdOptions - MQSO_DURABLE;
```

Varsayılan abonelik IBMSTOCKPRICESUBise, boş değerli bir dizgiyle değiştirilirse, örnek MQSO\_DURABLE işaretini kaldırır. Diğer parametrelere ilişkin varsayılan değerleri sağlayan örneği çalıştırırsanız, STOCKTICKER 'a gönderilen bir ek geçici abonelik oluşturulur ve yinelenen yayınlar alır. Bir sonraki örneği çalıştırıyorsanız, herhangi bir parametre olmadan, yalnızca bir yayın 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 81. Yönetilmeyen MQ abonesi-bölüm 3: kod gövdesi.

Bu örnekteki kodla ilgili ek açıklamalar aşağıdaki gibidir:

```
if (strlen(subscriptionQueue))
```

If there is no subscription queue name then the example uses MQHO\_NONE as the value of Hobj.

**MQOPEN(...);**

Abonelik kuyruğu açılır ve kuyruk tanıtıcısı Hobjçinde saklanır.

**MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);**

Abonelik, MQOPEN (ya da abonelik kuyruğu adı yoksa MQHO\_NONE ) tarafından geçirilen Hobj kullanılarak açılır. An unmanaged queue can be resumed without explicitly opening it with an MQOPEN.

**MQCLOSE(Hconn, &Hsub, MQCO\_NONE, &CompCode, &Reason);**

Abonelik, abonelik tanıtıcısı kullanılarak kapatılır. Aboneliğin dayanıklı olup olmadığına bağlı olarak, abonelik örtük bir MQCO\_KEEP\_SUB ya da MQCO\_REMOVE\_SUB ile kapatılır. You can close a durable subscription with MQCO\_REMOVE\_SUB, but you cannot close a non-durable subscription with MQCO\_KEEP\_SUB. MQCO\_REMOVE\_SUB işlemi, abonelik kuyruğuna gönderilmekte olan başka yayınların durdurulacağı aboneliği kaldırmaktan başka bir işlem değildir.

**MQCLOSE(Hconn, &Hobj, MQCO\_NONE, &CompCode, &Reason);**

Abonelik yönetilmezse, özel bir işlem yapılmamaktadır. Kuyruk yönetilirse ve abonelik belirttik ya da örtük MQCO\_REMOVE\_SUB ile kapatılırsa, tüm yayınlar bu noktada silinerek kuyruktan silinmektedir.

**gmo.MatchOptions = MQMO\_MATCH\_CORREL\_ID;****memcpy(md.CorrelId, sd.SubCorrelId, MQ\_CORREL\_ID\_LENGTH);**

Alınan iletilerin, aboneliğimiz için geçerli olduğundan emin olun.

Bu örnekten elde edilen sonuçlar, yayınlama/abone olma konularının özelliklerini gösterir:

In [Şekil 82 sayfa 867](#) the example starts by publishing 130 on the NYSE/IBM/PRICE topic.

```
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 82. 130-NYSE/IBM/PRICE 'yi yayınlayın*

Örneğin, örneğe ilişkin [Şekil 83 sayfa 867](#) uygulamasında varsayılan parametreleri kullanarak alıkonan yayın 130 olur. Sağlanan konu nesnesi ve konu dizisi, [Şekil 87 sayfa 869](#) içinde gösterildiği gibi yoksayıdır. Konu nesnesi ve konu dizisi, her zaman bir abonelik nesnesinden alınır, biri sağlandığında ve konu dizisi değişmez. Örneğin, gerçek davranışı MQSO\_CREATE, MQSO\_RESUME ve MQSO\_ALTER' in seçimine ya da bileşimine 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 83. Alıkonan yayını al*

In ([Şekil 84 sayfa 868](#)) hiçbir yayın alınmıyor. çünkü dayanıklı abonelik, alıkonan yayını zaten almış durumda. Bu örnekte, abonelik, kuyruk adı olmadan yalnızca abonelik adı sağlanarak sürdürülür. Kuyruk adı sağlandıysa, önce kuyruk açılacaktır ve tanıtıcı MQSUB' a iletilecektir.

**Not:** The 2038 error from MQINQ is due to the implicit MQOPEN of STOCKTICKER by MQSUB not including the MQOO\_INQUIRE option. Avoid the 2038 return code from MQINQ by opening the queue explicitly.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE IBMSTOCKPRICESUB / Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(1ter)|
C(reate)|R(esome)
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 84. Aboneliği sürdür

Şekil 85 sayfa 868’inde örnek, hedef olarak STOCKTICKER kullanarak, kalıcı olmayan bir yönetilmeyen abonelik yaratır. Bu yeni bir abonelik olduğu için alıkonan yayını alır.

```
W:\Subscribe3\Debug>solution3 STOCKS IBM/PRICE / STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(1ter)|
C(reate)|R(esome)
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 85. Alıkonan yayını, yönetilmeyen yeni kalıcı olmayan abonelikle al

Şekil 86 sayfa 868’ ta çakışan abonelikleri göstermek için başka bir yayın gönderilir ve alıkonan yayın değiştiriliyor. Daha sonra, yeni bir kalıcı olmayan, yönetilmeyen bir abonelik, abonelik adı sağlamayarak yaratılır. Alıkonan yayın, yeni abonelik için iki kez, yeni abonelik için bir kez ve STOCKTICKER kuyruğunda hala etkin olan kalıcı IBMSTOCKPRICESUB aboneliği için bir kez alınır. Örnek, uygulamanın değil, abonelikleri olan kuyruğun olduğu bir resimdir. Uygulamanın bu çağrısındaki IBMSTOCKPRICESUB aboneliğine başvurmamasına rağmen, uygulama yayını iki kez alır: bir kez yönetimsel olarak oluşturulan sürekli abonelikten, bir kez de uygulamanın kendisi tarafından oluşturulan kalıcı 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(1ter)|
C(reate)|R(esome)
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 86. Çakışma abonelikleri

Şekil 87 sayfa 869’ ta örnek, yeni bir konu dizgisi sağlamanın ve var olan bir aboneliğin değiştirilme aboneliğiyle sonuçlanmadığını gösterir.

1. İlk durumda, Resume , beklediğiniz gibi, var olan aboneliğe devam eder ve değiştirilen konu dizisini 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 "" NASDAQ/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Resume
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values "" "NASDAQ/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 "" NASDAQ/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Alter
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values "" "NASDAQ/IBM/PRICE" "IBMSTOCKPRICESUB" "STOCKTICKER" sd.Options=8201
Completion code 2 and Return code 2510

W:\Subscribe3\Debug>solution3 "" NASDAQ/IBM/PRICE IBMSTOCKPRICESUB STOCKTICKER Create
Optional parameters: topicName, topicString, subscriptionName, subscriptionQueue, A(lter)|C(reate)|R(esume)
Values "" "NASDAQ/IBM/PRICE" "IBMSTOCKPRICESUB" "STOCKTICKER" sd.Options=8202
Completion code 2 and Return code 2432
```

Şekil 87. Abonelik konuları değiştirilemez

## İlgili kavramlar

“Örnek 1: MQ Publication consumer” sayfa 853

MQ Publication tüketicisi, konuların kendisine abone olmayan bir IBM MQ ileti tükettirir.

“Örnek 2: Yönetilen MQ aboneliği” sayfa 855

Yönetilen MQ aboneliği, çoğu abone uygulaması için tercih edilen kalıptır. Bu örnek, kuyrukların, konuların ya da aboneliklerin yönetimsel tanımlamasını *hayır* gerektirir.

“Yayıncı uygulamaları yazılıyor” sayfa 845

İki örnek üzerinde çalışarak yayıncı uygulamaları yazmaya başlayın. İlki, bir kuyruğa ileti yerleştirmek için bir noktadan noktaya mümkün olduğunca yakın bir şekilde modellenir ve ikinci olarak konular, yayıncı uygulamaları için dinamik olarak daha yaygın bir kalıba yol açmaktadır.

## Yaşam çevrimlerini yayınla/abone ol

Yayıncı/abone olma uygulamaları tasarlarlarken konuların, aboneliklerin, abonelerin, yayınların, yayıncıların ve kuyrukların yaşam çevrimlerini göz önünde bulundurun.

Bir nesnenin (abonelik gibi) yaşam çevrimi, yaratılımla başlar ve silme işlemi ile sona erer. Ayrıca geçici askıya alma, üst ve alt konular olması, süre bitimi ve silinme gibi diğer durumları ve geçeceği değişiklikleri de içerebilir.

Geleneksel olarak, kuyruklar gibi IBM MQ nesnelere yönetimsel olarak ya da Programlar Komut Biçimi (PCF) kullanılarak yönetimsel programlar tarafından yaratılır. Yayıncı/abone olma, MQSUB ve MQCLOSE API fiillerinin, yalnızca kuyrukları yaratmayan ve silmeyen, ancak tüketilmeyen iletileri temizleyen ve yönetimsel olarak oluşturulan konu nesnelere iletilerle programlar olarak ya da yönetimsel olarak konu dizgileri arasında ilişkilendirmeler içeren yönetilen abonelikler kavramına sahip olan abonelikleri oluşturmak ve silmek için farklı bir şekilde yayınlanabilir/abone olunması farklıdır.

Bu işlevsel zenginlik, geniş bir yayıncı/abone olma gereksinimleri yelpazesi için hizmet sağlar ve aynı zamanda yayıncı/abone olma uygulamasının bazı ortak kalıplarını tasarlamayı basitleştirir. Yönetilen abonelikler, örneğin, yalnızca bu aboneliği yaratan program kadar uzun bir süre için amaçlanan bir aboneliğin hem programlamasını hem de yönetimini basitleştirir. Yönetilmeyen abonelikler, abone olmak ve yayınları tüketmek arasında gevşek bir bağlantının olduğu programlamayı basitleştirir. Merkezi olarak oluşturulan abonelikler, örüntünün merkezi bir denetim modeli (örneğin, uçuş bilgilerini otomatik geçitlere göndermesi gibi) merkezi bir denetim modeline dayalı olarak, tüketicilere yönlendirici yayın trafiğinden biri olduğu durumlarda kullanışlıdır, ancak geçit personeli bir kapıya bir uçuş numarası girerek, uçuş için yolcular kayıtlarına abone olmak üzere programlı olarak oluşturulmuş abonelikler kullanılmıştır.

bu son örnekte yönetilen bir dayanıklı abonelik uygun olabilir: yönetilen, abonelikler çok sık oluşturulmaya başlandığından ve geçit kapandığında ve abonelik programlı olarak kaldırıldığında net bir uç noktası olması, bir nedenle veya başka bir nedenle aşağı giden geçit abone programı nedeniyle bir yolcu kaydını kaybetmemek için dayanıklı bir uç nokta olmalıdır.<sup>8</sup> yolcu kayıtlarının geçitten yayınlanmasını başlatmak için, olası bir tasarım, geçit numarası kullanılarak hem yolcu kayıtlarına abone olmak için geçit uygulaması için hem de geçit numarasını kullanarak geçit açma olayını yayıncı olacaktır. yayıncı, yolcu kayıtlarını yayıncı olarak geçit açma olayına cevap veriyor. daha sonra faturalama gibi diğer ilgili taraflara da gidilebilecek olan uçuş kayıtlarını, uçuşun yerini ve müşteri hizmetlerine, kapı numarasının yolcuların cep telefonlarına metin bildirimlerine (mesaj) bildirerek yanıt veriyor.

<sup>8</sup> Yayıncı, diğer olası arızaları önlemek için yolcu kayıtlarını kalıcı iletiler olarak göndermelidir, elbette.

Merkezi olarak yönetilen abonelik, her bir geçit için önceden tanımlanmış bir kuyruk kullanarak, dayanıklı, yönetilmeyen bir model, yönlendirme yolcu listelerini kapiya yönlendirebilir.

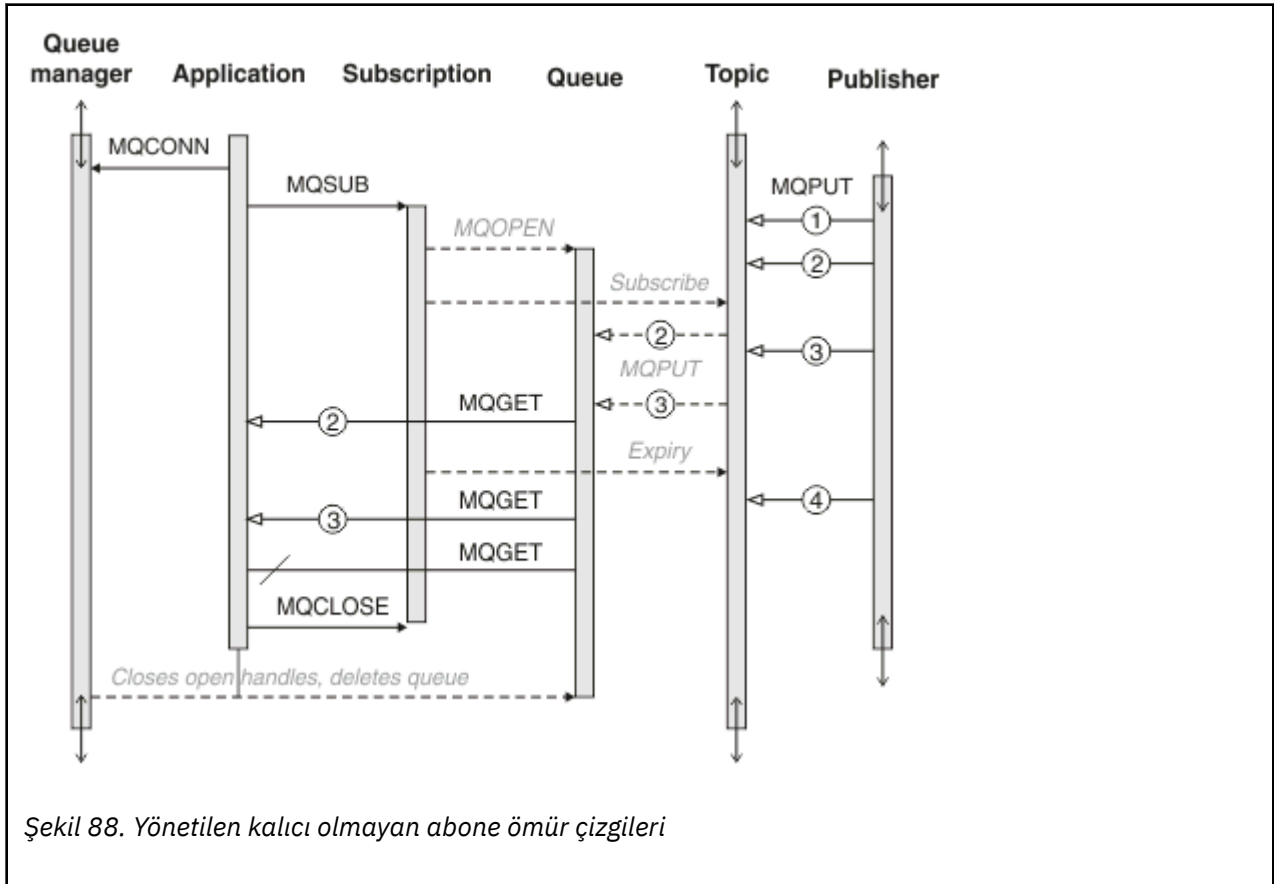
Sonraki yayınlama/abone olma yaşam çevrimlerine ilişkin aşağıdaki üç örnek, kalıcı olmayan, yönetilen dayanıklı ve yönetilmeyen dayanıklı abonelerin abonelikler, konular, kuyruklar, yayıncılar ve kuyruk yöneticisi ile nasıl etkileşimde bulunabileceğini ve sorumlulukların yönetim ile abone programları arasında nasıl bölünebileceğini gösterir.

### Yönetilen kalıcı olmayan abone

Şekil 88 sayfa 870 , yönetilen bir kalıcı olmayan abonelik yaratan bir uygulama, abonelikte belirtilen konuya yayınlanan iki ileti alma ve sonlandırma sırasında bir uygulama gösterir. Noktalı oklu italik gri yazı tipiyle etiketlenen etkileşimler örtük olur.

Notların bir kısmı var.

1. Uygulama, zaten iki kez yayınlanmış bir konuda abonelik yaratır. Abone ilk yayını aldığında, alıkonan yayın olan *saniye* yayını alır.
2. Kuyruk yöneticisi, geçici bir abonelik kuyruğu yaratmanın yanı sıra, konu için bir abonelik yaratmanın yanı sıra, bir geçici abonelik kuyruğu da yaratır
3. Aboneliğin süre bitimi var. Aboneliğin süresi dolduğunda, bu aboneliğe ilgili yayınların gönderilmesi sona ermez, ancak abone aboneliğin sona ermesinden önce yayınlanan iletileri almaya devam eder. Yayın süre bitimi, abonelik süre bitiminden etkilenmez.
4. Dördüncü yayın abonelik kuyruğuna konmaz ve sonuç olarak son MQGET yayını geri döndürmez.
5. Abone aboneliğini kapattısa da, kuyrukla ya da kuyruk yöneticisiyle olan bağlantısını kapatmıyor.
6. Uygulama sona erdikten kısa bir süre sonra kuyruk yöneticisi temizliyor. Abonelik yönetiliyor ve kalıcı olmayan bir abonelik kuyruğu olduğundan, abonelik kuyruğu silinir.



## Yönetilen dayanıklı abone

Yönetilen dayanıklı abone, önceki örneği bir adım daha ileriye götürür ve abone olunan uygulamanın sona erdirilmesinin ve yeniden başlatılıp başlatılabilmemesinin bir yönetilen aboneliğini gösterir.

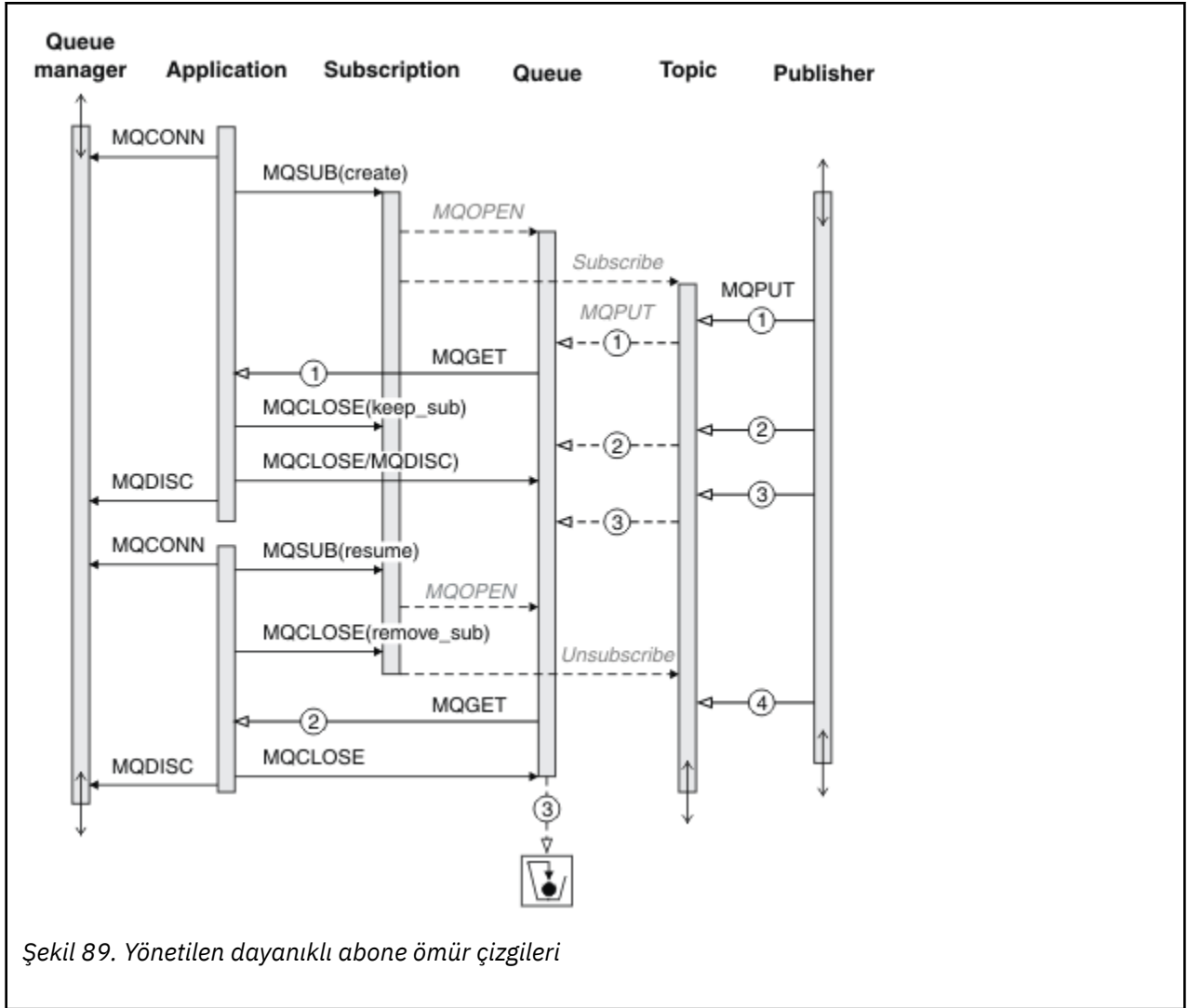
Notlara dikkat etmek için yeni noktalar var.

1. Bu örnekte, son olarak, yayın konusu abonelikte tanımlanmadan önce var olmamıştı.
2. The first time the subscriber terminates, it closes the subscription with the option MQCO\_KEEP\_SUB. Yönetilen bir kalıcı aboneliği örtük olarak kapamak için varsayılan davranış budur.
3. Abone aboneliğe devam ettiğinde, abonelik kuyruğu yeniden açılır.
4. Yeniden açılmadan önce kuyruğa yerleştirilen yeni yayın 2, abonelik kaldırıldıktan sonra da MQGET' e kullanılabilir.

Abonelik dayanıklı olsa da, abonenin gönderdiği tüm iletiler, yalnızca *her ikisi* abonelik dayanıklı ve iletiler kalıcı olduğunda aboneye güvenir. Message persistence depends on the setting of the Persistent field in the MQMD of the message sent by the publisher. Bir abonenin bu konuda bir kontrolü yok.

5. MQCO\_REMOVE\_SUB işaretiyle aboneliği kapatmak, aboneliği kaldırır ve abonelik kuyruğuna yerleştirilen diğer yayınların durdurulmasına neden olur. When the subscription queue is closed, then the queue manager removes the unread publication 3, and then deletes the queue. İşlem, aboneliğin yönetimsel olarak silinmesine eşdeğerdir.

**Not:** Do not delete the queue manually, or issue MQCLOSE with the option MQCO\_DELETE, or MQCO\_PURGE\_DELETE. Yönetilen aboneliğin görünür gerçekleştirme ayrıntıları, desteklenen IBM MQ arabiriminin bir parçası değildir. Kuyruk yöneticisi yönetimi, tam denetimi olmadığı sürece aboneliği güvenilir bir şekilde yönetemez.



### Yönetilmeyen dayanıklı abone

Üçüncü örneğe bir yönetici eklenir: yönetilmeyen dayanıklı abone. Yöneticinin bir yayınlama/abone olma uygulaması ile nasıl etkileşimde bulunabileceğini göstermek için iyi bir örnektir.

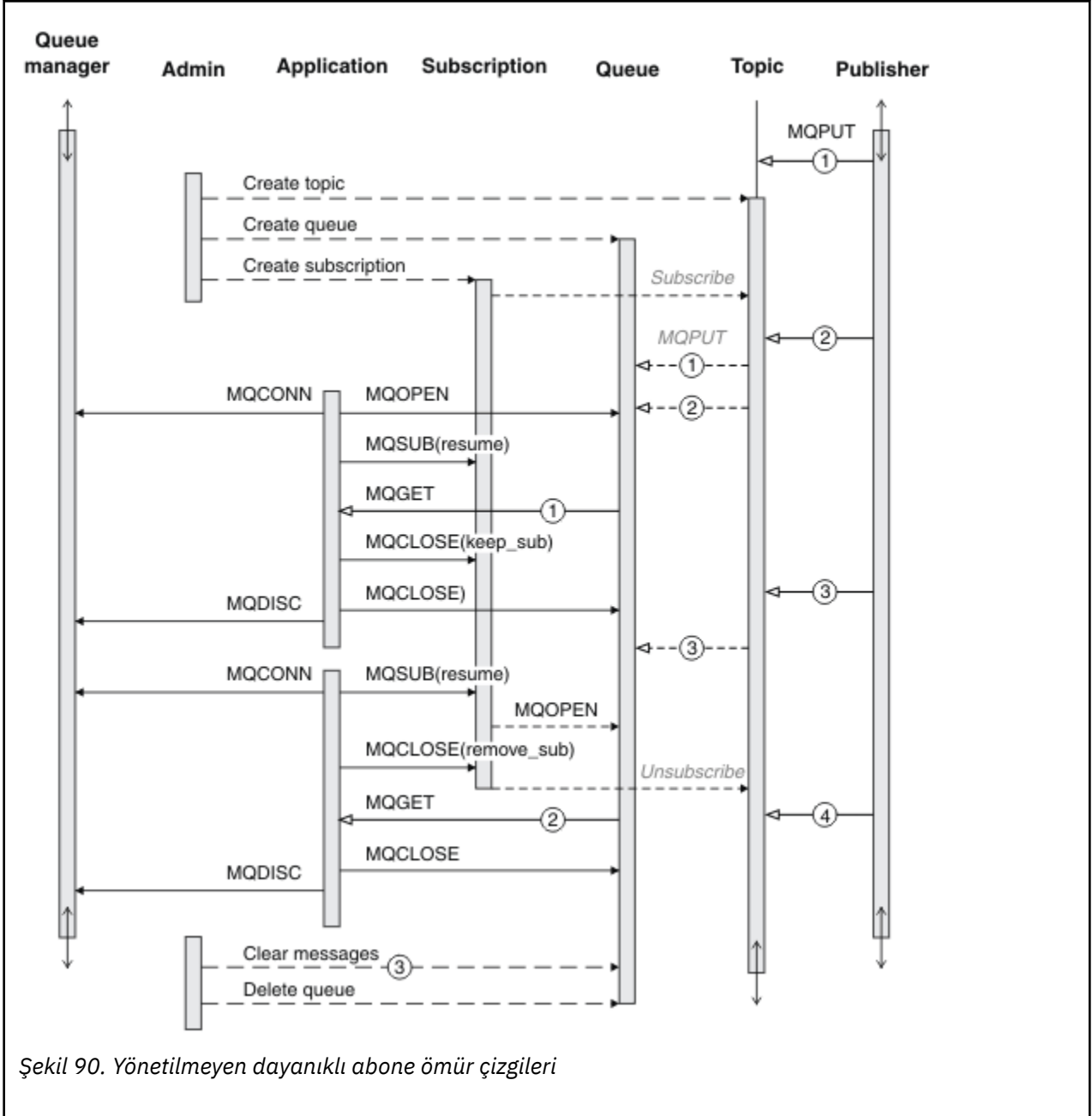
Notların olduğu noktalar listelenir.

1. Yayıncı bir iletiyi ( 1), daha sonra abonelik için kullanılan konu nesnesiyle ilişkilendirilecek bir konuya koyar. Konu nesnesi, genel arama karakterleri kullanılarak yayınlanan konuyla eşleşen bir konu dizisini tanımlar.
2. Konu, korunan bir yayınına sahip.
3. Yönetici bir konu nesnesi, bir kuyruk ve bir abonelik yaratır. Konu nesnesi ve kuyruğun abonelikten önce tanımlanması gerekir.
4. Uygulama, aboneliğe ilişkin kuyruğu açar ve MQSUB ' un kuyruğun tanıtıcısını geçirmesini sağlar. It could, alternatively, simply open the subscription, passing it the queue handle MQHO\_NONE. Converse doğru değil, yalnızca kuyruk tanıtıcısını abonelik adı olmadan geçirerek bir aboneliği sürdürülemez; bir kuyruk birden çok aboneliğine sahip olabilir.
5. Uygulama, aboneliği ilk kez açmış olsa da, MQSO\_RESUME seçeneğini kullanarak aboneliği açar. Bir yönetimsel olarak oluşturulan bir aboneliğin sürdürülmesi.
6. Abone alıkonan yayını alır, 1. Publication 2, although published before any publications were received by the subscriber, was published after the subscription started, and is the second publication on the subscription queue.



**Not:** Alıkonan yayın kalıcı bir ileti olarak yayınlanmadıysa, kuyruk yöneticisi yeniden başlatıldıktan sonra bu yayın kaybedilir.

7. Bu örnekte, abonelik dayanıklıdır. Bir programın yönetilmeyen, kalıcı olmayan bir abonelik yaratması mümkündür; bu, yöneticinin yapabildiği bir şey olmadığı açık olmalıdır.
8. The effect of the option MQCO\_REMOVE\_SUB on closing the subscription is to remove the subscription just as if the administrator had deleted it. Bu, kuyruğa gönderilen tüm yayınları durdurur; ancak, kuyruk kapatıldığında bile, önceden kuyruksa olan yayınları etkilemez; *yönetilen* kalıcı bir aboneliğin tersine.
9. Yönetici daha sonra geri kalan iletiyi ( 3 ) siler ve kuyruğu siler.



Yönetilmeyen bir abonelik için normal bir kalıp, yönetici tarafından gerçekleştirilecek kuyruk ve abonelik ev bakımı içindir. Tipik olarak, bir yönetilen abonenin davranışını taklit etmeyi ve uygulama kodunda programsal olarak kuyruklar ve abonelikler toplamayı denemez. Yönetim mantığını yazmaya gerek duyuyorsanız, yönetilen bir kalıp kullanarak aynı sonuçları elde edemediğinizi sorgunuz. Sıkı bir şekilde senkronize, tamamen güvenilir bir yönetim kodu yazmak kolay değildir. İletilerin, aboneliklerin ve kuyrukların durumundan bağımsız olarak silinebilmesi için, daha sonra el ile ya da otomatik bir yönetim programı kullanarak daha sonra toparlanabilirsiniz.

## **İleti özelliklerini yayınla/abone ol**

Birçok ileti özelliği IBM MQ yayınlama/abone olma mesajlarıyla ilgilidir.

### **PubAccountingSimgesi**

Bu, bu abonelikte eşleşen tüm yayın iletilerinin İleti Tanımlayıcısının (MQMD) AccountingToken alanında yer alacak değerdir. AccountingToken , iletinin kimlik bağlamının bir parçasıdır. İleti bağlamına ilişkin daha fazla bilgi için bkz. “İleti bağlamı” sayfa 44. MQMD ' deki AccountingToken alanı hakkında daha fazla bilgi için bkz. [AccountingToken](#).

### **PubApplIdentityData**

Bu, bu abonelikte eşleşen tüm yayın iletilerinin (MQMD) ApplIdentityVeri alanında yer alacak değerdir. ApplIdentityVerileri, iletinin kimlik bağlamının bir parçasıdır. İleti bağlamına ilişkin daha fazla bilgi için bkz. “İleti bağlamı” sayfa 44. MQMD ' de ApplIdentityVeri alanıyla ilgili daha fazla bilgi için bkz. [ApplIdentityData](#).

MQSO\_SET\_IDENTITY\_CONTEXT seçeneği belirtilmediyse, bu abonelik için yayınlanan her iletide ayarlanacak olan ApplIdentityVerisi, varsayılan bağlam bilgileri olarak boşluklara sahip olur.

MQSO\_SET\_IDENTITY\_CONTEXT seçeneği belirtilirse, kullanıcı tarafından PubApplIdentityData oluşturulmakta ve bu alan, bu abonelik için her yayında ayarlanacak ApplIdentityVerilerini içeren bir giriş alanıdır.

### **PubPriority**

Bu değer, bu abonelikte eşleşen tüm yayın iletilerinin İleti Tanımlayıcısının (MQMD) Öncelik alanında olacak değer. MQMD ' deki Öncelik alanıyla ilgili daha fazla bilgi için [Priority\(Öncelik\)](#) başlıklı konuya bakın.

Değer sıfırdan büyük ya da sıfıra eşit olmalıdır; sıfır, en düşük önceliğe sahip olmalıdır. Aşağıdaki özel değerler de kullanılabilir:

- MQPRI\_PRIORITY\_AS\_Q\_DEF-Bir abonelik kuyruğu, MQSUB çağrısındaki Hobj alanında sağlandığında ve yönetilen bir tanıtıcı değilse, iletinin önceliği bu kuyruğun DefPriority özniteliğinden alınır. Belirlenen kuyruk bir küme kuyruğuna ya da kuyruk-adı çözünürlük yolunda birden çok tanımlama varsa, bu öncelik MQMD ' deki Öncelik için açıklandığı gibi, yayın iletisi kuyruğa konduğunda öncelik belirlenir. MQSUB çağrısı yönetilen bir tanıtıcı kullanıyorsa, ileti için öncelik, abone olunan konuyla ilişkili model kuyruğunun DefPriority özniteliğinden alınır.
- MQPRI\_PRIORITY\_AS\_PUBLICID-İletiye ilişkin öncelik, özgün yayının önceliğidir. Bu, bu alanın ilk değeridir.

### **SubCorrelTanıtıcısı**



**Uyarı:** Bir ilinti tanıtıcısı yalnızca, bir sıradüzeninde değil, yayınlama/abone olma kümesindeki kuyruk yöneticileri arasında geçirilebilir.

Bu abonelikte eşleşmesi için gönderilen tüm yayınlar, ileti tanımlayıcısında bu ilinti tanıtıcısını içerir. Birden çok abonelik, yayınlarını almak için aynı kuyruğu kullanıyorsa, MQGET by correlation ID ' nin kullanılması yalnızca belirli bir abonelik için yayınların elde edilebilmesini sağlar. Bu ilinti tanıtıcısı kuyruk yöneticisi ya da kullanıcı tarafından yaratılabilir.

MQSO\_SET\_COREL\_ID seçeneği belirtilmediyse, ilinti tanıtıcısı kuyruk yöneticisi tarafından oluşturulur ve bu alan, bu abonelik için yayınlanan her iletide ayarlanacak ilinti tanıtıcısını içeren bir çıkış alanıdır.

MQSO\_SET\_COREL\_ID seçeneği belirtilirse, kullanıcı tarafından ilinti tanıtıcısı oluşturulmakta 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 tanımlayıcısı, iletinin özgün tanıtıcısıyla yaratılan ilinti tanımlayıcısıdır.

MQSO\_GROUP\_SUB seçeneği belirtilirse ve belirtilen ilinti tanıtıcısı, aynı kuyruğu ve çakışan bir konu dizisini kullanan var olan gruplanmış bir abonelikte aynıysa, yayının bir kopyasıyla yalnızca gruptaki en önemli abonelik sağlanır.

## SubUserVerileri

Bu, abonelik kullanıcı verileridir. Bu alandaki aboneliğe ilişkin sağlanan veriler, bu aboneliğe gönderilen her yayının MQSubUserVeri iletisi özelliği olarak içerilir.

## Yayın özellikleri

Çizelge 128 sayfa 875 , bir yayın iletisiyle birlikte sağlanan yayın özelliklerini listeler.

You can access these properties directly from the **MQRFH2** folder, or retrieve them using MQINQMP. MQINQMP , sorgulanacak özelliğin adı olarak özellik adını ya da **MQRFH2** adını kabul eder.

Çizelge 128. Yayın özellikleri			
Özellik adı	MQRFH2 adı	Tip	Tanım
MQTopicString	mmps.Top	MQTYPE_STRING	Konu dizesi
MQSubUserVerileri	mmps.Sud	MQTYPE_STRING	Abone kullanıcı verileri
MQIsRetained	mmps.Ret	MQTYPE_BOOLEAN	Alıkonan yayın
MQPubOptions	mmps.Pub	MQTYPE_INT32	Yayın seçenekleri
MQPubLevel	mmps.Pbl	MQTYPE_INT32	Yayın düzeyi
MQPubTime	mmpse.Pts	MQTYPE_STRING	Yayın zamanı
MQPubSeqNum	mmpse.Seq	MQTYPE_INT32	Yayın sıra numarası
MQPubStrIntData	mmpse.Sid	MQTYPE_STRING	Yayıncı tarafından eklenen String/Integer verileri
MQPubFormat	mmpse.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 tabi olarak).

Olağan durumda ileti sıralaması, her abonenin belirli bir kuyruk yöneticisinden belirli bir konuya, belirli bir yayıncıdan yayımlandığı sırayla yayımlandığı sırayla ileti aldığı anlamına gelir.

Ancak, tüm IBM MQ iletileriyle olduğu gibi, iletiler, zaman zaman, siparişin dışında teslim edilme olasıdır. Bu durum aşağıdaki durumlarda gerçekleşebilir:

- Ağdaki bir bağlantı aşağı inerse ve sonraki iletiler başka bir bağlantı boyunca yeniden yönlendirilirse
- Bir kuyruk geçici olarak tam olarak dolduysa ya da geçici olarak engellenirse, bir ileti çıkmaz bir kuyruğa konursa ve bu nedenle gecikirse, sonraki iletiler düz olarak geçer.
- Sistem yöneticisi, yayıncılar ve aboneler çalışmaya devam ederken bir kuyruk yöneticisini silerse, kuyruğa alınan iletilerin, durdurulacak ileti kuyruğuna ve aboneliklere konmasına neden olur.

Bu şartlar ortaya çıkmazsa, yayınlar her zaman sırayla teslim edilir.

**Not:** Publish/Subscribe ile gruplanmış ya da bölümlenmiş iletiler kullanmak olanaklı değildir.

## Yayınların ele geçirmesi

Bir yayını durdurabilir, değiştirebilir ve daha sonra başka bir aboneye ulaşmadan yeniden yayınlayabilirsiniz.

Aşağıdaki eylemlerden birini yapmak için bir yayının bir aboneye ulaşmadan önce kesişmesini isteyebilirsiniz:

- İletiyi ek bilgi ekle
- İletiyi engelle
- İletiyi dönüştür

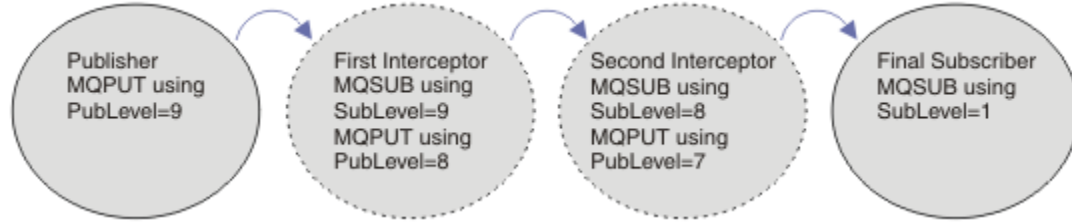
Her iletide aynı işlemi gerçekleştirebilir ya da işlemi, aboneliğe, iletiye ya da ileti üstbilgisine bağlı olarak değişiklik yapabilirsiniz.

### İlgili başvurular

[MQ\\_PUBLISH\\_EXIT-Yayınlama çıkışı](#)

#### Abonelik düzeyleri

Bir yayının son abonelerine ulaşmadan önce kesişmesini engellemeye ilişkin abonelik düzeyini ayarlayın. Abone olan bir abone daha yüksek abonelik düzeyine abone olur ve daha düşük bir yayın düzeyinde yeniden yayınlar. Son abonelere teslim edilmeden önce, bir yayında ileti işleme işlemi gerçekleştirmek için bir araya gelen aboneler zinciri oluşturun.



Şekil 91. Engellenen abonelerin sırası

Bir yayını durdurmak için **MQSD** SubLevel özneliğini kullanın. Bir ileti algılandıktan sonra, **MQPMO** PubLevel özneliğini değiştirerek, bu ileti dönüştürülebilir ve daha sonra daha düşük bir yayın düzeyinde yeniden yayınlanabilir. Daha sonra, ileti son abonelere gider ya da daha düşük bir abonelik düzeyinde ara abone tarafından yeniden durdurulur.

Kesişme aboneleri genellikle, yeniden yayınlamadan önce bir iletiyi dönüştürür. Bir dizi kesişme aboneleri bir ileti akışı oluşturur. Diğer bir seçenek olarak, kesişen yayını yeniden yayınlamayabilirsiniz: Alt abonelik düzeylerindeki aboneler iletiyi almazlardı.

Diğer abonelerden önce, yayıncının yayınları aldığından emin olun. Engelleyici abonelik düzeyini diğer abonelere göre ayarlayın. Varsayılan olarak, abonelerin bir SubLevel ' si ( 1) vardır. En yüksek değer 9' dir. Bir yayının en az en yüksek SubLevel en az bir PubLevel (Ortak Düzey) ile başlaması gerekir. Başlangıçta 9' un varsayılan PubLevel ile yayınlayın.

- Bir konuyla ilgili bir aboneye sahipseniz, SubLevel ' i 9 olarak ayarlayın.
- Bir konuyla ilgili birden çok kesme uygulaması için, her ardışık işlem aboneleri için alt SubLevel ayarlayın.
- You can implement a maximum of 8 intercepting applications, with subscription levels from 9 down to 2 inclusive. İletinin son alıcısının bir SubLevel ( 1) vardır.

Yayının PubLevel ' a eşit ya da daha düşük en yüksek abonelik düzeyine sahip kesici, önce yayını alır. Belirli bir abonelik düzeyinde bir konu için tek bir araya girme aboneleri yapılandırın. Belirli bir abonelik düzeyinde birden çok abonelerin olması, yayının birden çok kopyasında abone olunan uygulamaların son kümesine gönderilmektedir.

A subscriber with a SubLevel of 0 is used as a catchall. Bu ileti, son abonelerin iletiyi almaması durumunda yayını alır. A subscriber with SubLevel of 0 might be used to monitor the publications that no other subscribers received.

## Bir araya gelen aboneyi programlamak

Çizelge 129 sayfa 877’inde açıklanan abonelik seçeneklerini kullanın.

Çizelge 129. Abonelerin kesişmesine ilişkin abonelik seçenekleri	
Abonelik seçeneği	Notlar
MQSO_SET_CORREL_ID ve SubCorrelId , MQCI_NONEolarak ayarlanır	Kesişen yayının CorrelId (CorrelId) yayını özgün yayınlı aynı tutun. <b>Not:</b> Bir yayının bir sıradüzeninde ilinti tanıtıcısını geçemezsiniz. Alan, kuyruk yöneticisi tarafından kullanılır.
PubPriority , MQPRI_PRIORITY_AS_PUBLISHEDolarak ayarlanır	Engellenen yayının önceliğini, özgün yayınlı aynı tutun.

The options in Çizelge 129 sayfa 877 must be used by all the intercepting subscribers. Sonuçta, ilinti tanıtıcısı ve ileti önceliği, özgün yayınlayıcı ayarından değiştirilmez.

Kesişme abonesi yayını işlediğinde, iletiyi, kendi aboneliğinin SubLevel değerinden daha düşük bir PubLevel ile aynı konuya yeniden yayınlar. If the intercepting subscriber set a SubLevel of 9, it republishes the message with a PubLevel of 8.

İletiyi doğru bir şekilde yeniden yayınlamak için, özgün yayından birkaç bilgi parçası gereklidir. Reuse the same **MQMD** as in the original message and set MQPMO\_PASS\_ALL\_CONTEXT to ensure all information in the **MQMD** is passed on to the next subscriber. Copy the values from the message properties shown in Çizelge 130 sayfa 877 into the corresponding fields of the republished message. Kesişme abonesi bu değerleri değiştirebilir. **MQPMO**’ e ek değerler eklemek için OR işlecini kullanın. Koyma iletisi seçeneklerini birleştirmek içinSeçenekler alanını kullanın.

Yönetilen yayın kuyruğunu kullanmak yerine, yayın kuyruğunu açık bir şekilde açmanız gerekir. You cannot set MQSO\_SET\_CORREL\_ID for a managed queue. You also cannot set MQOO\_SAVE\_ALL\_CONTEXT on a managed queue. “Örnekler” sayfa 878’ ta listelenen kod parçalarına bakın.

Çizelge 130. Yeniden yayınlanan iletiler içinMQPUT değerleri	
İletiyi MQPUT kullanarak yeniden yayınlama	Yayın iletisinde bilgi
<b>MQOD</b> . ObjectString	İleti Özelliği MQTopicString
<b>MQPMO</b> . Options	İleti Özelliği MQPubOptions

Son abonede, abonelik seçeneklerini farklı bir şekilde ayarlama seçeneği vardır. Örneğin, yayınlama önceliğini belirttik olarak MQPRI\_PRIORITY\_AS\_PUBLISHEDyerine de ayarlayabilirsiniz. Son abonenin ayarları, yalnızca zincirdeki son kesişen aboneden yayını etkiler.

## Alıkonan yayınlar

Alıkonan bir yayın, özgün put-message seçeneklerini yeniden yayınlanan iletiye kopyalayarak, araya girdikten sonra korunmalıdır.

MQPMO\_TUT seçeneği, yayınlayıcı tarafından ayarlanır. Her bir araya gelen abonenin, MQPubOptions i, yeniden yayınlanan iletinin put-message (put-message) seçeneklerine Çizelge 130 sayfa 877’inde gösterildiği şekilde aktarması gerekir. Put-message seçeneklerinin kopyalanması, özgün yayınlayıcıya göre, yayının korunulup tutulmayacağı gibi seçenekleri korur.

Bir yayın, kesişme abone olan abonelerin geçişini bitirdiğinde ve son abonelere teslim edildiğinde, son abonelere teslim edilir. Yeni aboneler, SubLevel 1’ da, alıkonan yayını istiyor, daha fazla bir araya getirilmeden teslim alıyor. 1 değerinden büyük bir SubLevel aboneler, alıkonan yayın gönderilmez. Sonuç olarak, alıkonan yayın, abonelerin ikinci kez kesişme zincirine göre değiştirilmez.

## Örnekler

Örnekler, bir araya gelen aboneyi oluşturmak için birleştirilebilen kod parçalarıdır. Kod, üretim kalitesinden çok kısa bir süre olacak şekilde yazılır.

Şekil 92 sayfa 878 içindeki önışlemci yönergelerinde, MQINQMP MQI çağrısı için gerekli olan yayın iletilerinden alınacak iki özellik tanımlanır.

```
#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 92. Ön işlemci yönergeleri

Şekil 93 sayfa 878 , kod parçalarında kullanılan bildirimleri listeler. Vurgulanan terimler dışında, bildirimler bir IBM MQ uygulaması için standarttır.

Vurgulanan put ve Al seçenekleri, tüm bağlamı geçirmek için başlatılır. Vurgulanan MQTOPICSTRING ve MQPUBOPTIONS , ön işlemci yönergelerinde tanımlı olan özellik adlarına ilişkin MQCHARV initializer'larıdır. Adlar MQINQMP' a iletilir.

```
int main(int argc, char **argv) {
    MQLONG Reason = MQRC_NONE;
    MQLONG CompCode = MQCC_OK;
    MQHCONN Hcon = MQHC_UNUSABLE_HCONN;
    MQCHAR QMName[49] = "";
    MQCMHO CrtMsgH0pts = {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 InqProp0pts = {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 93. Bildirimler

Bildirimlerde kolayca gerçekleştirilmeyen kullanıma hazırlama işlemleri Şekil 94 sayfa 879 içinde gösterilir. Vurgulanan değerler açıklama gerektiriyor.

#### SYSTEM.NDURABLE.MODEL.QUEUE

In this example, instead of using MQSUB to open a managed non-durable subscription, the model queue, SYSTEM.NDURABLE.MODEL.QUEUE, is used to create a temporary dynamic queue. Tanıtıcısı MQSUB' a iletilir. Kuyruk doğrudan açılarak, tüm ileti bağlamını kaydedebilir ve abonelik seçeneğini (MQSO\_SET\_CORREL\_ID) ayarlamasını.

#### MQGMO\_CURRENT\_VERSION

IBM MQ yapılarının büyük kısmının yürürlükteki sürümünü kullanmak önemlidir. gmo.MsgHandle gibi alanlar yalnızca denetim yapılarının en son sürümünde kullanılabilir.

#### MQGMO\_PROPERTIES\_IN\_HANDLE

Özgün yayında konu dizgisi ve koyma iletisi seçenekleri, ileti özelliklerini kullanarak araya giriş abonesi tarafından alınmak üzere ayarlanır. Diğer bir seçenek, iletide MQRFH2 yapısının doğrudan okunmasını sağlar.

#### MQSO\_SET\_CORREL\_ID

Birlikte MQSO\_SET\_CORREL\_ID ile birlikte kullanılması,

```
memcpy(sd.SubCorrelId, MQCI_NONE, sizeof(sd.SubCorrelId));
```

Bu seçeneklerin etkisi ilinti tanıtıcısından geçmektedir. Özgün yayıncı tarafından ayarlanan ilinti tanıtıcısı, yayınlama aboneliği tarafından alınan yayının ilinti tanıtıcısı alanına yerleştirilir. Her bir araya gelen abone aynı ilinti tanıtıcısında geçiyor. Son abonede, aynı ilinti tanıtıcısını alma seçeneği vardır.

**Not:** Yayınlama bir yayınlama/abone olma sıradüzeninden geçirilirse, ilinti tanıtıcısı hiçbir zaman alıkonmaz.

#### MQPRI\_PRIORITY\_AS\_PUBLISHED

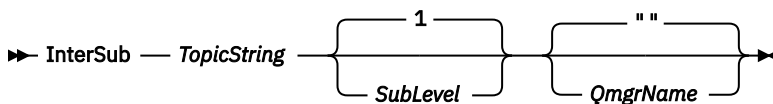
Yayın, yayımlandığı şekliyle aynı ileti önceliğine sahip yayın kuyruğuna yerleştirilir.

```
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 94. Kullanıma Hazırlama

Şekil 95 sayfa 880 , komut satırı parametrelerini okuyacak, kullanıma hazırlama işlemini tamamlanacak ve kesişme aboneliğini oluşturabilmek için kod parçasını gösterir.

Programı komutla çalıştırın.



Hata işlenmesini mümkün olduğunca açık hale getirmek için, her bir MQI çağrısından neden kodu farklı bir dizi ögesinde saklanır. Her çağrıdan sonra tamamlanma kodu test edilir ve bu değer MQCC\_FAIL ise, denetim do { } while (0) kod öbeğinden çıkar.

İki kayda değer kod satırı,

**pmo.PubLevel = sd.SubLevel - 1;**

Yeniden yayınlanan iletinin yayın düzeyini, kesilecek abonenin abonelik düzeyinden bir daha küçük bir düzeye ayarlar.

**gmo.MsgHandle = Hmsg;**

İleti özelliklerini döndürmek için MQGET 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 95. Yayınların kesişme hazırlığı yapılıyor*

The main code fragment, [Şekil 96 sayfa 881](#), gets messages from the publication queue. İleti özelliklerini sorgular ve konu dizesini kullanarak iletileri yeniden yayınlar ve özgün **MQPMO**. seçenek yayınlarının özellikleri.

Bu örnekte, yayınında herhangi bir dönüşüm gerçekleştirilmez. Yeniden yayınlanan yayının konu dizgisi, her zaman, araya abone olunan abonenin abone olduğu konu dizgisiyle eşleşir. Kesişme abonesi, aynı yayın kuyruğuna gönderilen birden çok aboneliğin durdurulmasından sorumlu olursa, farklı aboneliklerle eşleşen yayınları ayırt etmek için konu dizesini sorgulamak gerekebilir.

MQINQMP çağrıları vurgulanır. Konu dizesi ve yayını, ileti seçenekleri özelliklerini doğrudan çıkış kontrol yapılarına yazılıp yazılır. putOD.ObjectString MQCHARV uzunluk alanını belirttik bir değerle boş olarak sonlandırılmış dizgiye değiştirmenin tek nedeni, dizginin çıkışını yapmak için printf kullanılmasıdır.



```

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 96. Yayını engelle ve yeniden yayınlama

Son kod parçası Şekil 97 sayfa 881’de 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);
}

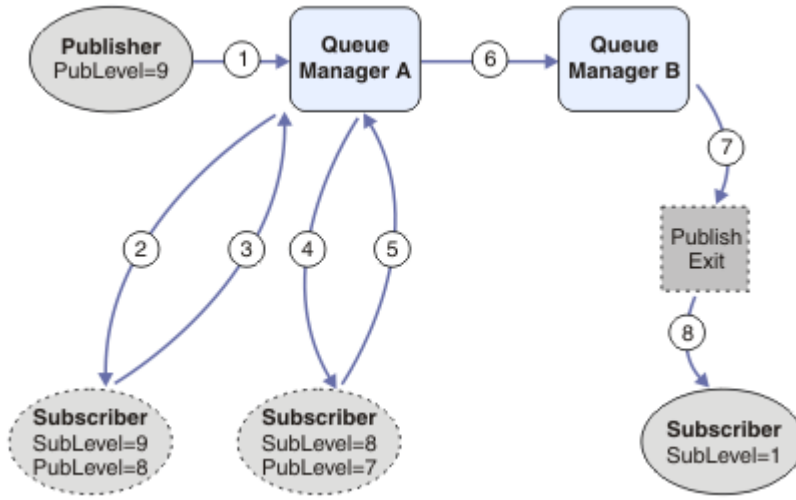
```

Şekil 97. Tamamlanma

*Yayınları ve dağıtım yayınlama/abone olma*

Bir dağıtılmış yayınlama/abone olma topolojisinde araya girdiğinizde ya da yayınlama çıkışlarını konuşlandırdığınızda basit bir örüntüye uyun. Yayıncılarla aynı kuyruk yöneticilerindeki araya girmekte olan aboneleri konuşlandırın ve son aboneler olarak aynı kuyruk yöneticilerindeki çıkışları yayınlayın.

Şekil 98 sayfa 882 , bir yayınlama abone olma kümesine bağlı iki kuyruk yöneticisini gösterir. A publisher creates a publication to a cluster topic at publication level 9. Numaralandırılmış oklar, yayınlama adımların sırasını, abonelere küme konusuna akıştıkça gösterir. The publication is intercepted by the subscriber with Alt düzey 9 and republished with Publevel 8. Alt düzey 8’ da bir abone tarafından yeniden yakalanır. Abone, Publevel 7 konumunda yeniden yayınlar. Kuyruk yöneticisi tarafından sağlanan yetkili sunucu aboneleri, yayını kuyruk yöneticisi B ' ye iletir; burada son aboneye ek olarak bir Yayınlama çıkışı konuşlandırılmıştır. The publication is processed by the Publish exit before it is finally received by the final subscriber at Alt düzey 1. Araya gelen aboneler ve yayınlama çıkışı kırık çerçevelerle gösterilir.



Şekil 98. Bir kümedeki başlangıç ve yayınlama çıkışıdır

Yalın örüntünün amacı, aynı yayını almak için bir yayın alan her abone içindir. Bu yayın, abonenin bağlı olduğu yerden bağımsız olarak aynı dönüşümler sırasının üzerinden geçer. Yayıncıların ya da son abonelerin bağlı olduğu yere bağlı olarak, dönüşümlerin sırasının değişmesini önlemeniz gerekebilir. Makul bir kural dışı durum, yayını son olarak her bir aboneye teslim etmek için uyarlamak olacaktır. Yayınlama çıkışı kullanın.

Dağıtılmış bir yayınlama/abone olma topolojisinde, kesişen abonelerin ve yayınlama çıkışlarının konuşlandırılacağı yeri dikkatli bir şekilde değerlendirmelisiniz. Bu düz örüntü, aboneleri yayıncılarla aynı kuyruk yöneticisine yerleştirir ve son abonelerle aynı kuyruk yöneticilerine yayınlar yayınlayın.

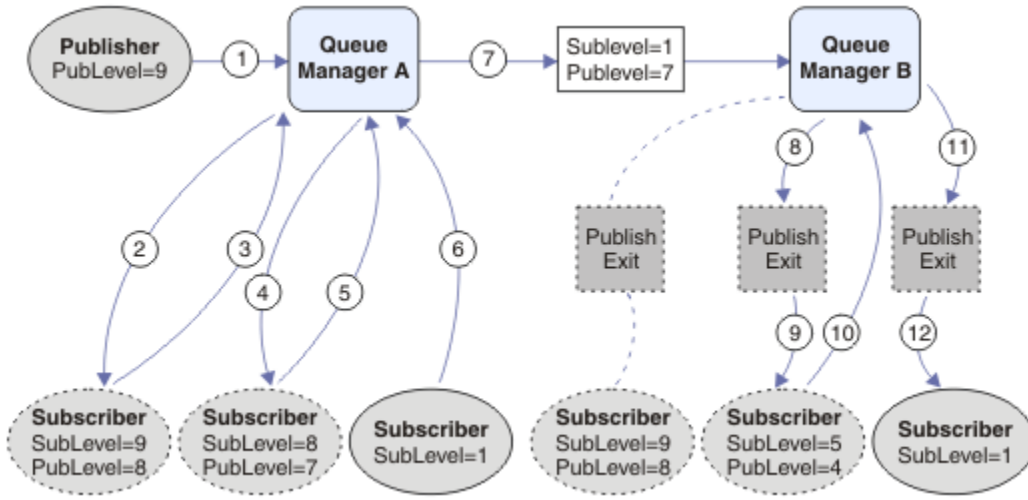
## Düzenek karşıtı

Şekil 99 sayfa 883 shows how matters can go awry, if you do not follow a simple pattern.

Konuşlandırmayı karmaşık hale getirebilmek için, kuyruk yöneticisine bir son abone eklenir ve iki ek kesici aboneleri kuyruk yöneticisi B ' ye eklenir.

The publication is forwarded to queue manager B at PubLevel 7, where it is intercepted by a subscriber at SubLevel 5 before being consumed by the final subscriber at SubLevel 1. Yayınlama çıkışı, hem algılayıcı tüketiciye, hem de kuyruk yöneticisi B ' deki son tüketiciye iletilmeden önce yayını ele geçirmektedir. Bu yayın, Yayınlama çıkışı tarafından işlenmeden kuyruk yöneticisi A ' nın son abonelerine ulaşır.

Bir yayınlama/abone olma topolojisinde, yetkili sunucu aboneleri SubLevel 1'a abone olur ve son arama aboneleri tarafından ayarlanan PubLevel 1 ' ı iletir. In Şekil 99 sayfa 883, the result is that the publication is not intercepted by the subscriber using SubLevel 9 at queue manager B.



Şekil 99. Kesişen abonelerin karmaşık devreye alınması

### Yayın seçenekleri

İletilerin yayınlanma şeklini denetleyen çeşitli seçenekler vardır.

### Yanıtlama yanıtı-abonelerden bilgi almak

If you do not want subscribers to be able to reply to publications they receive, it is possible to withhold information in the ReplyToQ and ReplyToQmgr fields of the MQMD by using the MQPMO\_SUPPRESS\_REPLYTO put-message option. Bu seçenek kullanılırsa, kuyruk yöneticisi bu bilgileri herhangi bir aboneye iletmeden önce yayın aldığı MQMD 'den kaldırır.

Bu seçenek, bir ReplyToQ ' u gerektiren bir rapor seçeneğiyle birlikte kullanılamaz; bu, çağrıya çağrılan MQRC\_MISSING\_REPLY\_TO\_Q ile başarısız olma girişiminde bulunursa kullanılamaz.

### Yayın düzeyi

Yayın düzeylerinin kullanılması, hangi abonelerin yayını alacağını denetlemeye ilişkin bir yöntemdir. Yayın düzeyi, yayınlı hedeflenen abonelik düzeyini belirtir. Yalnızca en yüksek abonelik düzeyine sahip olan abonelikler, yayının yayın düzeyinden daha az ya da bu yayın düzeyiyle aynı olan abonelikler için geçerli olan abonelikleri alır. Bu değer, sıfır ile dokuz aralığında olmalıdır; sıfır, en düşük yayın düzeyidir. Bu alanın ilk değeri 9 'tır. Yayın ve abonelik düzeylerinin kullanılarından biri de yayını engelle.

### Yayının herhangi bir aboneye teslim edilmediği denetleniyor

Bir yayının abonelere teslim edilmediğini denetlemek için, MQPUT çağrısına sahip MQPMO\_WARN\_IF\_NO\_XX\_ENCODE\_CASE\_ONE subs\_matched put-message seçeneğini kullanın. Put işlemi tarafından MQCC\_UYARI ve bir MQRC\_NO\_ALTCHATCHED bir neden kodu döndürülürse, yayın herhangi bir aboneliğe teslim edilemedi. Koyma işleminde MQPMO\_RETAIN seçeneği belirtilirse, ileti alıkonur ve daha sonra tanımlanmış eşleşen abonelikte teslim edilir. Dağıtılmış bir yayınlama/abone olma sisteminde, MQRC\_NO\_ALTS\_MATCHED neden kodu, yalnızca kuyruk yöneticisinde konu için kayıtlı yetkili sunucu aboneliği yoksa döndürülür.

### Abonelik seçenekleri

İleti aboneliklerinin nasıl işleneceğini denetleyen birkaç seçenek vardır.

### İleti kalıcılığı

Kuyruk yöneticileri, yayıncı tarafından ayarlanan abonelere ilettikleri yayınların sürekliliğini korurlar. Yayınlayıcı, kalıcılığı aşağıdaki seçeneklerden biri olacak şekilde ayarlar:

0

Kalıcı olmayan

1

Kalıcı

2

Kuyruk/konu tanımlaması olarak kalıcılık

Yayınla/abone olma için yayınlayıcı, konu nesnesini ve **topicString** çözümlenen bir konu nesnesiyle çözülebilir. Yayınlayıcı, kuyruk/konu tanımlaması olarak kalıcılık belirtiyorsa, çözümlenen konu nesnesindeki varsayılan kalıcılık yayınlama için belirlenir.

## Alıkonan yayınlar

Alıkonan yayınlar alındığında denetim yapmak için aboneler iki abonelik seçeneği kullanabilir:

### Yalnızca istek üzerine yayınla, MQSO\_PUBLICATIONS\_ON\_REQUEST

Bir abonenin yayınlarını aldığı anda denetime sahip olmasını istiyorsanız, MQSO\_PUBLICATIONS\_ON\_REQUEST abonelik seçeneğini kullanabilirsiniz. Bir abone daha sonra MQSUBRQ çağrısını kullanarak (özgün MQSUB çağrısından döndürülen Hsub tanıtıcısını belirterek) bir konunun korunan yayınına gönderildiğini istemek için yayınları aldığı anda denetleyebilirler. MQSO\_PUBLICATIONS\_ON\_REQUEST abonelik seçeneğini kullanan aboneler, alıkonmamış yayınlar almaz.

MQSO\_PUBLICATIONS\_ON\_REQUEST değerini belirlerseniz, herhangi bir yayını almak için MQSUBRQ 'yı kullanmanız gerekir. MQSO\_PUBLICATIONS\_ON\_REQUEST 'i kullanmayacaksa, iletiler yayımlandığı gibi iletilir.

Bir abone, MQSUBRQ çağrısını kullanıyorsa ve abonelikte genel arama karakterleri kullanıyorsa, abonelik, bir konu ağacındaki birden çok konu ya da düğüm ile eşleşebilir; tüm tutulan iletiler (varsa) aboneye gönderilecektir.

Bu seçenek, bir kuyruk yöneticisi, abone uygulaması çalışmasa bile, kullanıcı tarafından bir aboneye yayın göndermeye devam edeceği için, kalıcı aboneliklerde kullanıldığında özellikle yararlı olabilir. Bu, abone kuyruğunda iletilerin birikmesine neden olabilir. Bu oluşturma, abonenin MQSO\_PUBLICATIONS\_ON\_REQUEST seçeneğini kullanarak kaydolması önenebilir. Diğer bir seçenek olarak, istenmeyen iletilerin bir oluşturmalarını önlemek için uygulamanıza uygun olması durumunda, dayanıklı olmayan abonelikler de kullanabilirsiniz.

Bir abonelik dayanıklıysa ve bir yayınlayıcı alıkonacaksa, abone uygulaması, yeniden başlatma işleminden sonra durum bilgilerini yenilemek için MQSUBRQ çağrısını kullanabilir. Daha sonra, abonenin MQSUBRQ çağrısını kullanarak durumunu düzenli olarak yenilemesi gerekir.

Bu seçeneği kullanarak MQSUB çağrısının sonucu olarak hiçbir yayın gönderilmez. Bağlantı kesildiğinde sürdürülen kalıcı abonelik, özgün abonelik bu seçeneği kullanacak şekilde yapılandırıldıysa, MQSO\_PUBLICATIONS\_ON\_REQUEST seçeneğini kullanacaktır.

### Yalnızca yeni yayınlar, MQSO\_NEW\_PATICATIONS\_ONLY

Bir konuyla ilgili alıkonan bir yayın varsa, yayından sonra abonelik yapan aboneler bu yayının bir kopyasını alır. Bir abone, yapılmakta olan abonelikten daha önce yapılmış bir yayını almak istemezse, abone MQSO\_NEW\_XX\_ENCODE\_CASE\_ONE publications\_only abonelik seçeneğini kullanabilir.

## Abonelikleri gruplama

Yayınları almak için bir kuyruk ayarladıysanız ve aynı kuyruğa yayınların beslenmesi için bir dizi çakışan abonelikler varsa, abonelikleri gruplamanızı göz önünde bulundurun. Bu durum, [Çakışan abonelikleri](#) içindeki örneğe benzer.

Bir konuya abone olduğunuzda, MQSO\_GROUP\_SUB seçeneğini ayarlayarak yinelenen yayınlar almaktan kaçınabilirsiniz. Sonuçta, gruptaki birden çok abonelik bir yayının konusuyla eşleştiğinde, yayının kuyruğa konmasından yalnızca bir abonelik sorumlu olur. Yayın konusu ile eşleşen diğer abonelikler yok sayılır.

Yayını kuyruğa yerleştirmekten sorumlu olan abonelik, herhangi bir genel arama karakteri ile karşılaşmadan önce en uzun eşleşen konu dizisine sahip olduğu temel alınarak seçilir. Bu, en yakın

eşleşen abonelik olarak düşünülebilir. Özellikleri, MQSO\_NOT\_OWN\_PUBS özelliğine sahip olup olmadığı da içinde olmak üzere, yayına yansıtılır. Bu durumda, eşleşen diğer abonelikler MQSO\_NOT\_OWN\_PUBS özelliğine sahip olmasalar da, kuyruğa yayın gönderilmez.

Yinelenen yayınları ortadan kaldırmak için tüm aboneliklerinizi tek bir gruba yerleştiremezsiniz. Gruplanmış abonelikler aşağıdaki koşulları yerine getirmelidir:

1. Aboneliklerin hiçbiri yönetilmedi.
2. Bir abonelik grubu, yayınları aynı kuyruğa teslim eder.
3. Her abonelik aynı abonelik düzeyinde olmalıdır.
4. Gruptaki her abonelik için yayın iletisi, aynı ilinti tanıtıcısına sahiptir.

Her abonelik sonucunun aynı ilinti tanıtıcısına sahip bir yayın iletisinde olmasını sağlamak için, MQSO\_SET\_COREL\_ID ' u yayında kendi ilinti tanımlayıcınız yaratmak için ayarlayın ve her abonelikte **SubCorrelId** alanında aynı değeri ayarlayın. **SubCorrelId** değerini MQCI\_NONE değerine ayarlamayın.

Ek bilgi için [./com.ibm.mq.ref.dev.doc/q100080\\_.dita#q100080\\_/mqso\\_group\\_sub](http://com.ibm.mq.ref.dev.doc/q100080_.dita#q100080_/mqso_group_sub) konusuna bakın.

## Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

Bunlar, bir kuyruk yöneticisinin bir nesneyi işlemesine neden olur. Her IBM MQ nesnesi tipinin öznitelikleri, [Nesnelerin özniteliklerialanında](#) ayrıntılı olarak açıklanmıştır.

Bazı öznitelikler, nesne tanımlandığında ayarlanır ve yalnızca IBM MQ komutları kullanılarak değiştirilebilir; bu tür bir öznitelige örnek olarak, bir kuyruğa konulan iletiler için varsayılan öncelik verilebilir. Diğer öznitelikler kuyruk yöneticisinin çalışmasından etkilenir ve zaman içinde değişebilir; örnek, kuyruğun yürürlükteki derinliğini gösterir.

MQINQ çağrısını kullanarak çoğu öznitelğin yürürlükteki değerlerini sorgulayabilirsiniz. MQI, bazı kuyruk özniteliklerini değiştirebileceğiniz bir MQSET çağrısı da sağlar. Diğer herhangi bir nesne tipinin özniteliklerini değiştirmek için MQI çağrılarını kullanamazsınız. Bunun yerine aşağıdaki kaynaklardan birini kullanmalısınız:

- **ULW** MQSC komutları içinde anlatılan MQSC olanağı.
- **IBM i** IBM için CL komutları başvurusu içinde açıklanan CHGMQMx CL komutları ya da MQSC olanağı.
- **z/OS** ALTER işleci komutları ya da REPLACE seçeneğiyle DEFINE komutları MQSC komutları içinde açıklanmıştır.

**Not:** Bu belgede, nesnelerin özniteliklerinin adları, bu belgede MQINQ ve MQSET çağrılarıyla kullandığınız biçimlerde gösterilir. Öznitelikleri tanımlamak, değiştirmek ya da görüntülemek için IBM MQ komutlarını kullandığınızda, konu bağlantılarındaki komutların açıklamalarında gösterilen anahtar sözcükleri kullanarak öznitelikleri tanımlamanız gerekir.

Hem MQINQ hem de MQSET çağrıları, tanınabilmek için seçicilerin dizilerini kullanır sorgulamak ya da ayarlamak istediğiniz öznitelikler. Birlikte çalışabileceğiniz her öznitelik için bir seçici vardır. Seçici adının bir öneki var, öznitelğin niteliği tarafından belirlenir:

Çizelge 131. Seçici adlarına ilişkin önekler	
Önek	Tanım
MQCA_	Bu seçiciler, karakter verileri (örneğin, bir kuyruğun adı) içeren özniteliklere başvurur.

Çizelge 131. Seçici adlarına ilişkin örnekler (devamı var)	
Önek	Tanım
MQIA_	Bu seçiciler sayısal değerleri (örneğin, <i>CurrentQueueDepth</i> , kuyruklardaki ileti sayısı) ya da sabit bir değeri (queuegibi, kuyruk yöneticisinin eşitleme noktalarını destekleyip desteklemediği <i>SyncPoint</i> gibi) içeren özniteliklere başvurur.

MQINQ ya da MQSET çağrılarını kullanmadan önce, uygulamanızın kuyruk yöneticisine bağlı olması ve özniteliklerin belirlenmesi ya da aranması için nesneyi açmak üzere MQOPEN çağrısını kullanmanız gerekir. Bu işlemler “Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773 ve “Nesnelerin açılması ve kapatılması” sayfa 781’ de anlatılır.

Nesne özniteliklerinin sorulması ve ayarlanmasıyla ilgili daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- “Bir nesnenin özniteliklerinin sorulmasına neden oluyor” sayfa 886
- “MQINQ çağrısının başarısız olduğu bazı durumlar” sayfa 887
- “Kuyruk özniteliklerinin ayarlanması” sayfa 888

### İlgili kavramlar

“Message Queue Interface-Genel Bakış” sayfa 760

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 781

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

“İletileri Kuyruğa Koyma” sayfa 791

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

“Kuyruktan İleti Alınması” sayfa 806

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabılır alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

“Starting IBM MQ applications using triggers” sayfa 899

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

“MQI ve kümelerle çalışma” sayfa 918

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

“Uygulamaların IBM MQ for z/OSüzerinde kullanılması ve yazılması” sayfa 922

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

“IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları” sayfa 65

This information helps you to write IMS applications using IBM MQ.

### **Bir nesnenin özniteliklerinin sorulmasına neden oluyor**

Herhangi bir IBM MQtürünün özniteliklerini sorgulamak için MQINQ çağrısını kullanın.

Bu aramaya giriş olarak, aşağıdaki bilgileri sağlamanız gerekir:

- Bağlantı tanıtıcısı.

- Bir nesne tanıtıcısı.
- Seçicilerin sayısı.
- Her bir seçici, MQCA\_ \* ya da MQIA\_ \* biçiminde olacak şekilde, öznitelik seçicileri dizisi. Her seçici, sorgulamak istediğiniz bir değeri olan bir özniteliği temsil eder ve her bir seçici, nesne tutamacının temsil ettiği nesne tipi için geçerli olmalıdır. Seçicileri herhangi bir siparişte belirtebilirsiniz.
- Sormakta olduğunuz tamsayı özniteliklerin sayısı. Tamsayı öznitelikleri sorulmayacaksa, sıfır değerini belirtin.
- The length of the character attributes buffer in *CharAttrLength*. Bu, her bir karakter özniteliği dizilimini tutmak için gereken uzunluklar toplamını en az bir toplamın olması gerekir. Karakter öznitelikleri sorulmayacaksa, sıfır değerini belirtin.

MQINQ ' un çıkışı şöyledir:

- Diziye kopyalanan bir tamsayı öznitelik değerleri kümesi. Değer sayısı *IntAttrCount* tarafından belirlenir. *IntAttrCount* ya da *SelectorCount* sıfırsa, bu parametre kullanılmaz.
- Karakter özniteliklerinin döndürüldüğü arabellek. Arabellek uzunluğu **CharAttrLength** parametresiyle verilir. *CharAttrLength* ya da *SelectorCount* sıfırsa, bu parametre kullanılmaz.
- Bir tamamlanma kodu. Tamamlanma kodu bir uyarı veriyorsa, arama yalnızca kısmen tamamlandı demektir. Bu durumda neden kodunu inceleyin.
- Bir neden kodu. Üç kısmi tamamlama durumu vardır:
  - Seçici kuyruk tipi için geçerli değil
  - Tamsayı öznitelikleri için yeterli alan yok
  - Karakter öznitelikleri için yeterli alan yok

Bu durumların birden fazlası ortaya çıkarsa, geçerli olan ilk değer döndürülür.

Çıkış ya da sorgu için bir kuyruk açsanız ve bu, yerel olmayan bir küme kuyruğuna çözümlerse, yalnızca kuyruk adı, kuyruk tipi ve ortak öznitelikleri sorgulayabilirsiniz. MQOO\_BIND\_ON\_Açık kullanıldıysa, ortak özniteliklerin değerleri seçilen kuyruklardır. MQOO\_BIND\_NOT\_FIXED ya da MQOO\_BIND\_ON\_GROUP kullanıldıysa ya da MQOO\_BIND\_AS\_Q\_DEF kullanılmadıysa ve **DefBind** kuyruk özniteliği MQBND\_BIND\_NOT\_FIXED ise, değerler olası küme kuyruklarından oluşan rasgele bir değerindir. Ek bilgi için “MQOPEN ve kümeler” sayfa 919 ve MQOPEN başlıklı konuya bakın.

**Not:** Çağrı tarafından döndürülen değerler, seçilen özniteliklerin anlık görüntüleridir. Programınız döndürülen değerlerde işlem yapmadan önce öznitelikler değişebilir.

MQINQ' da MQINQ çağrısının bir açıklaması var.

### **MQINQ çağrısının başarısız olduğu bazı durumlar**

Özniteliklerini sorgulamak için bir diğer adı açsanız, diğer ad kuyruğunun özniteliklerini (başka bir kuyruğa erişmek için kullanılan IBM MQ nesnesi), temel kuyruğun özniteliklerini değil olarak geri döndürmeniz gerekir.

Ancak, diğer adın çözdüğü temel kuyruğun tanımlaması kuyruk yöneticisi tarafından da açılır ve başka bir program, MQREAD ve MQINQ çağrılarınız arasındaki aralıktaki temel kuyruk kullanımını değiştirirse, MQINQ çağrısının başarısız olur ve MQRC\_OBJECT\_CHANGED neden kodunu döndürür. Diğer ad kuyruğu nesnesinin öznitelikleri değiştirilirse arama da başarısız olur.

Benzer bir şekilde, uzak bir kuyruğun özniteliklerini sorgulamak üzere açtığınızda, yalnızca uzak kuyruğa ilişkin yerel tanımlamanın özniteliklerini geri döndürmeniz gerekir.

Araştırdığınız kuyruk öznitelikleri tipi için geçerli olmayan bir ya da daha çok seçici belirlerseniz, MQINQ çağrısı bir uyarıyla tamamlanır ve çıkışı aşağıdaki gibi ayarlar:

- Tamsayı öznitelikleri için, ilgili *IntAttrs* öğeleri MQIAV\_NOT\_UYGULANABİLİR olarak ayarlanır.
- Karakter öznitelikleri için, *CharAttrs* dizgisinin karşılık gelen bölümleri yıldız işaretleri olarak ayarlanır.

Sormakta olduğunuz nesne özniteliklerinin tipi için geçerli olmayan bir ya da daha çok seçici belirlerseniz, MQINQ çağrısı başarısız olur ve MQRC\_SELECTOR\_ERROR neden kodunu döndürür.

Bir model kuyruğuna bakmak için MQINQ ' yi çağırabilirsiniz; MQSC olanağını ya da altyapınızda var olan komutları kullanın.

### **Kuyruk özniteliklerinin ayarlanması**

MQSET çağrısını kullanarak kuyruk özniteliklerinin nasıl ayarlanacak bilgilerini öğrenmek için bu bilgileri kullanın.

MQSET çağrısını 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 çağrıyla aynı parametrelere sahip. Ancak, MQSET için, tamamlanma kodu ve neden kodu dışındaki tüm değiştirgeler giriş değiştirgeleridir. Kısmi tamamlama durumları yoktur.

**Not:** Yerel olarak tanımlanmış kuyruklar dışındaki IBM MQ nesnelere özniteliklerini ayarlamak için MQI ' yı kullanamazsınız.

MQSET çağrısıyla ilgili daha fazla ayrıntı için bkz. [MQSET](#).

### **İş birimlerinin kesinleştirilmesi ve yedeklenmesi**

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabılır alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

Bu konuda aşağıdaki terimler kullanılmıştır:

- Kesinleştir
- İptal Et
- Syncpoint koordinasyonu
- Syncpoint
- İş birimi
- Tek aşamalı kesinleştirme
- İki aşamalı kesinleştirme

Bu işlem işleme terimlerini alıyorsanız, [“IBM MQ uygulamalarındaki eşitleme noktası konuları” sayfa 890' a atlayabilirsiniz.](#)

#### **Kesinleştir ve geri al**

Bir program, bir iş birimi içindeki bir kuyruğa ileti yerleştirdiğinde, bu ileti yalnızca program iş birimini kesinleştirdiğinde diğer programlar tarafından görülebilir hale getirilmektedir. Bir iş birimini kesinleştirmek için, veri bütünlüğünü korumak için tüm güncellemelerin başarılı olması gerekir. Program bir hata saptarsa ve koyma işleminin kalıcı olmamaya karar verirse, iş birimini geri alabilirler. Bir program bir geri alma işlemi gerçekleştirdiğinde, IBM MQ bu iş birimi tarafından kuyruğa konulan iletileri kaldırarak kuyruğu geri yükler. Programın kesinleştirme ve geri çıkış işlemlerini gerçekleştirmesi, programın çalışmakta olduğu ortama bağlıdır.

Benzer bir şekilde, program bir iş birimi içindeki bir kuyruktan ileti aldığı anda, program iş birimini kesinleştirenceye kadar bu ileti kuyruktan kalır, ancak diğer programlar tarafından alınamayacak ileti görüntülenemez. Program, iş birimini kesinleştirdiğinde, ileti kuyruktan kalıcı olarak silinir. Program iş birimini yedeklerse, IBM MQ , iletilerin diğer programlar tarafından alınabilmesini sağlanarak, kuyruğu geri yükler.



## Syncpoint koordinasyonu, uyumluluk noktası, çalışma birimi

*Syncpoint eşgüdümü* , iş birimlerinin veri bütünlüğü ile kesinleştirileceği ya da yedekleneceği süreçtir.

Değişikliklerin kesinleştirme ya da geri alınması, en basit durumda, bir işlemin sonunda alınır. Ancak, bir uygulamanın veri değişikliklerini bir işlem içindeki diğer mantıksal noktalarda eşitlemesi daha kullanışlı olabilir. Bu mantıksal noktalar *syncpoints* (ya da *eşzamanlama noktaları* ) olarak adlandırılır. ve iki syncpointe arasında bir dizi güncelleme işleme süresi *iş birimi*olarak adlandırılır. Birden çok MQGET çağrısı ve MQPUT çağrıları tek bir iş biriminin bir parçası olabilir.

Bir iş birimi içindeki ileti sayısı üst sınırı, ALTER QMGR komutunun MAXUMSGS özniteliğinde denetlenebilir.

### Tek aşamalı kesinleştirme




Bir *tek aşamalı kesinleştirme* işlemi, bir programın, değişikliklerini diğer kaynak yöneticileriyle koordine etmeden bir kuyrukta güncelleyebileceği bir işlemdir.

### İki aşamalı kesinleştirme

*İki aşamalı kesinleştirme* işlemi, bir programın IBM MQ kuyruklarına yaptığı güncellemelerin diğer kaynaklarla (örneğin, Db2 denetimi altındaki veritabanları) eşgüdümlü olarak eşgüdümlü olarak gerçekleştirilebileceği bir işlemdir. Bu tür bir sürecin altında, tüm kaynaklarda yapılan güncellemeler kesinleştirilir ya da birlikte yedeklenir.

İş birimlerinin işlenmesine yardımcı olmak için IBM MQ , **BackoutCount** özniteliğini sağlar. Bu, iş birimi içindeki bir iletinin geriletileceği her defasında artırılır. If the message repeatedly causes the unit of work to abnormally end, the value of the *BackoutCount* finally exceeds that of the *BackoutThreshold*. Bu değer, kuyruk tanımlandığında ayarlanır. Bu durumda uygulama, iletiyi çalışma biriminden kaldırabilir ve *BackoutRequeueQName* ' ta tanımlandığı gibi başka bir kuyruğa yerleştirebilir. İleti taşındığında, iş birimi kesinleştirilebilir.

İş birimlerinin kesinleştirilmesi ve yedeklenmesi hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- “IBM MQ uygulamalarındaki eşitleme noktası konuları” sayfa 890
-  “IBM MQ for z/OS uygulamalarındaki eşitleme noktaları” sayfa 891
-  “Syncpoints in CICS for IBM i applications” sayfa 893
- “IBM MQ for Multiplatforms içindeki eşitleme noktaları” sayfa 894
-  “IBM i dış eşitleme noktası yöneticisine arabirimler” sayfa 898

### İlgili kavramlar

“Message Queue Interface-Genel Bakış” sayfa 760

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

“Nesnelerin açılması ve kapatılması” sayfa 781

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

“İletileri Kuyruğa Koyma” sayfa 791

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

“Kuyruktan İleti Alınması” sayfa 806

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

“Starting IBM MQ applications using triggers” sayfa 899

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

“MQI ve kümelerle çalışma” sayfa 918

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OS üzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

[“IBM MQ for z/OS üzerinde IMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

### **IBM MQ uygulamalarındaki eşitleme noktası konuları**

IBM MQ uygulamalarındaki eşitleme noktalarını kullanmaya ilişkin bilgi edinmek için bu bilgileri kullanın.

İki aşamalı kesinleştirme aşağıdaki ortamlar tarafından desteklenir:

- **AIX** IBM MQ for AIX
- **IBM i** IBM MQ for IBM i
- **Linux** IBM MQ for Linux
- **Solaris** IBM MQ for Solaris
- **Windows** IBM MQ for Windows
- **z/OS** z/OS için CICS Transaction Server
- **z/OS** TXSeries
- **z/OS** IMS/ESA
- **z/OS** RRS ile z/OS toplu işi
- X/Open XA arabirimini kullanan diğer dış koordinatörler

Tek aşamalı kesinleştirme aşağıdaki ortamlar tarafından desteklenir:

- **IBM i** IBM MQ for IBM i
- **UNIX** IBM MQ Değiştirildiği tarih/saat: UNIX
- **Windows** IBM MQ for Windows
- **z/OS** z/OS toplu

Dış arabirimler hakkında daha fazla bilgi için [“Çoklu Platformlar üzerindeki Dış Syncpoint Yöneticilerine Arabirimler” sayfa 896](#) başlıklı konuya ve Open Group tarafından yayınlanan XA belgeleri *CAE Specification Distributed Transaction Processing: The XA Specification* başlıklı konuya bakın. Hareket yöneticileri (CICS, IMS, Encina ve Tuxedo gibi), diğer kurtarılabilir kaynaklarla koordine edilen iki aşamalı kesinleştirmeye katılabilirler. Bu, IBM MQ tarafından sağlanan kuyruğa alma işlevlerinin, hareket yöneticisi tarafından yönetilen bir iş birimi kapsamında getirilebilir.

Samples shipped with IBM MQ show IBM MQ coordinating XA-compliant databases. Bu örneklerle ilgili daha fazla bilgi için bkz. [“IBM MQ örnek yordamsal programlarının kullanılması” sayfa 1092.](#)

IBM MQ uygulamanıza, çağrıyı eşitleme noktası denetimi kapsamına almak isteyip istemediğiniz çağrılan her put ve get (alma) çağrısını belirleyebilirsiniz. Bir put işleminin syncpoint denetimi altında çalışması için, MQPUT ' u çağırdığınızda, MQPMO yapısının *Options* alanında MQPMO\_SYNCPOINT değerini kullanın. Alma işlemi için, MQGMO yapısının *Options* alanında MQGMO\_SYNCPOINT değerini kullanın. Bir seçeneği belirtir olarak seçmezseniz, varsayılan işlem altyapıya bağlıdır:

- **Multi** Syncpoint denetimi varsayılan değeri NO. (NO).
- **z/OS** Syncpoint denetimi varsayılan değeri YES (YES) olur.

When an MQPUT1 call is issued with MQPMO\_SYNCPOINT, the default behavior changes, so that the put operation is completed asynchronously. Bu, MQOD ve MQMD yapılarındaki bazı alanlara dayanan, ancak şimdi tanımsız değerler içeren bazı uygulamaların davranışlarında değişikliğe neden olabilir. Bir uygulama, koyma işleminin zamanuyumlu olarak gerçekleştirilmesini ve tüm uygun alan değerlerinin tamamlandığından emin olmak için MQPMO\_SYNC\_RESPONSE değerini belirtebilir.

Uygulamanız bir MQPUT ya da MQGET altında bir MQPUT ya da MQGET yanıtına yanıt olarak bir MQRC\_BACKED\_OUT neden kodu aldığında, uygulamanın olağan durumda MQBACK kullanarak yürürlükteki hareketi geri alması ve uygunsa, işlemin tamamını yeniden deneyin. Uygulama bir MQCMIT ya da MQDISC çağrısına yanıt olarak MQRC\_BACKED\_OUT alırsa, MQBACK 'ı çağırmasına gerek yoktur.

Bir MQGET çağrısı her yedeklendiğinde, etkilenen iletinin MQMD yapısındaki *BackoutCount* alanı artırılır. Yüksek *BackoutCount* , sürekli olarak yedeklenen bir iletiyi belirtir. Bu, araştırmanız gereken bu iletiyle ilgili bir sorun olduğunu gösterebilir. *BackoutCount* ile ilgili ayrıntılar için bkz. [BackoutCount](#) .

RRS ile z/OS toplu işi dışında, kesinleştirilmemiş istekler varken bir program MQDISC çağrısını yayınlarsa, örtük bir eşitleme noktası oluşur. Program olağan dışı sona ererse, örtük bir geri alma gerçekleşir.

**z/OS** z/OS' ta, program ilk olarak MQDISC çağrılmadan sona ererse, örtük bir eşitleme noktası da oluşur. program, MQ ' e bağlı TCB olağan şekilde sona ererse, olağan durumda sona ermiş olarak kabul edilir. z/OS UNIX System Services and Language Environment (LE) altında çalışırken, olağandışı sonlar ya da sinyaller için varsayılan koşul işleme çağrılır. LE koşulu işleyicileri, hata koşulunu ve TCB ' yi olağan bir şekilde sona erdirir. Bu koşullar altında MQ iş birimini kesinleştirir. Daha fazla bilgi için bkz. [Dil Ortamı Koşulu İşleme Tanıtımı](#).

**z/OS** IBM MQ for z/OS programları için, geriletme ortaya çıkarsa (bir *MQGET-error-backout* döngüden kaçınmak için) bir iletinin yedeklenmemesi gerektiğini belirtmek için MQGMO\_MARK\_SKIP\_BACKUT seçeneğini kullanabilirsiniz. Bu seçeneğin kullanılmasıyla ilgili bilgi için bkz. [“Geri alma işlemi atlanıyor” sayfa 835](#).

Kuyruk özniteliklerinde (MQSET çağrısıyla ya da komutlarla) yapılan değişiklikler, iş birimlerinin kesinleştirilmesiyle ya da yedeklenmesiyle etkilenmez.

### **z/OS IBM MQ for z/OS uygulamalarındaki eşitleme noktaları**

Bu konuda, hareket yöneticisinde eşitleme noktalarının nasıl kullanılacağı açıklanmaktadır ( CICS ve IMS ) ve toplu iş uygulamaları.

#### **z/OS CICS Transaction Server for z/OS uygulamaları içindeki eşitleme noktaları**

Bir CICS uygulamasında EXEC CICS SYNCPOINT komutunu kullanarak bir uyumluluk noktası oluşturmanızı sağlar.

Önceki eşitleme noktasına yapılan tüm değişiklikleri geri almak için EXEC CICS SYNCPOINT ROLLBACK komutunu kullanabilirsiniz. Daha fazla bilgi için *CICS Application Programming Reference* adlı kılavuza bakın.

Diğer kurtarılabılır kaynaklar iş biriminde yer aldıysa, kuyruk yöneticisi ( CICS syncpoint manager ile birlikte) iki aşamalı kesinleştirme protokolünde yer alır; tersi durumda, kuyruk yöneticisi tek aşamalı kesinleştirme işlemi gerçekleştirir.

Bir CICS uygulaması MQDISC çağrısını yayınlarsa, örtük eşitleme noktası alınmaz. Uygulama olağan şekilde kapatılırsa, açık kuyruklar kapatılır ve örtük bir kesinleştirme gerçekleşir. Uygulama olağan dışı bir şekilde kapatılırsa, tüm açık kuyruklar kapatılır ve örtük bir geri alma gerçekleşir.

#### **z/OS IMS uygulamalarındaki eşitleme noktaları**

Bir IMS uygulamasında, IOPCB ve CHKP (denetim noktası) için GU (benzersiz al) gibi IMS çağrılarını kullanarak bir uyumluluk noktası oluşturun.

Önceki denetim noktasından bu yana yapılan tüm değişiklikleri geri almak için, IMS ROLB (rollback) çağrısını kullanabilirsiniz. Daha fazla bilgi için IMS belgelerine bakın.

Kuyruk yöneticisi ( IMS syncpoint manager ile birlikte), başka kurtarılabılır kaynaklar da iş biriminde yer alırsa, iki aşamalı kesinleştirme protokolünde yer alır.

Tüm açık tutamaçlar, bir eşitleme noktasında IMS bağdaştırıcısı tarafından kapatılır (toplu ya da iletilmeyen bir BMP ortamı dışında). Bunun nedeni, başka bir kullanıcının sonraki iş birimini başlatabileceği ve MQCONN, MQCONNX ve MQOPEN çağrıları yapılırken, MQPUT ya da MQGET çağrıları yapılmadığında IBM MQ güvenlik denetimi gerçekleştirilmektedir.

Ancak, bir WFI (WFI) ya da sözde Giriş (PWFI) ortamında, IMS , bir sonraki ileti gelene kadar ya da uygulamaya bir QC durum kodu döndürülünceye kadar tutamaçları kapatmasını IBM MQ ' e bildirmez. Uygulama IMS bölgesinde bekliyorsa ve bu tutamaçların herhangi biri tetiklenen kuyruklara aitse, kuyruklar açık olduğu için tetikleme gerçekleşmez. Bir WFI ya da PWFI ortamında çalışan bu neden için, sonraki ileti için GU 'yu IOPCB' ye yapmadan önce, kuyruk tanıtıcılarını belirttik olarak MQCLOSE işlemi gerçekleştirmelidir.

Bir IMS uygulaması (bir BMP ya da MPP) MQDISC çağrısını yayınlarsa, açık kuyruklar kapatılır, ancak örtük eşitleme noktası alınmaz. Uygulama olağan şekilde kapatılırsa, açık kuyruklar kapatılır ve örtük bir kesinleştirme gerçekleşir. Uygulama olağan dışı bir şekilde kapatılırsa, tüm açık kuyruklar kapatılır ve örtük bir geri alma gerçekleşir.

### z/OS toplu iş uygulamalarındaki eşitleme noktaları

Toplu iş uygulamaları için, IBM MQ syncpoint yönetim çağrıları kullanabilirsiniz: MQCMIT ve MQBACK. Önceki sürümlerle uyumluluk için, CSQBCMT ve CSQBBAK, eşanlamlılar olarak kullanılabilir.

**Not:** If you need to commit or back out updates to resources managed by different resource managers, such as IBM MQ and Db2, within a single unit of work you can use RRS. Daha fazla bilgi için bkz. [“Hareket yönetimi ve kurtarılabılır kaynak yöneticisi hizmetleri” sayfa 892.](#)

## **MQCMIT çağrısını kullanarak değişiklikler kesinleştiriliyor**

Giriş olarak, MQCONN ya da MQCONNX çağrısının döndürdüğü bağlantı tanıtıcısını (*Hconn*) sağlamanız gerekir.

MQCMIT ' in çıkışı bir tamamlanma kodu ve neden kodudur. Eşitleme noktası tamamlandıysa, ancak kuyruk yöneticisi, bir önceki eşitleme noktasından bu yana put ve get işlemlerini yedeklediyseniz, arama bir uyarıyla tamamlanır.

MQCMIT çağrısının başarıyla tamamlanması, kuyruk yöneticisine uygulamanın bir uyumluluk noktasına ulaştığını ve önceki eşitleme noktasından bu yana yapılan tüm işlemlerin kalıcı hale getirilip getirilmemesinin tamamlandığını gösterir.

Hata yanıtlarının tümü MQCMIT ' in tamamlanmadığı anlamına gelmez. Örneğin, uygulama MQRC\_CONNECTION\_BROKEN ' ı alabilir.

[MQCMIT](#) içinde MQCMIT çağrısının bir açıklaması var.

## **MQBACK çağrısını kullanarak değişikliklerin yedeklenmesi**

Giriş olarak bir bağlantı tanıtıcısı sağlamanız gerekir (*Hconn*). MQCONN ya da MQCONNX çağrısının döndürdüğü tanıtıcıyı kullanın.

MQBACK ' tan çıkış, bir tamamlanma kodu ve neden kodudur.

Çıkış, kuyruk yöneticisine uygulamanın bir uyumluluk noktasına ulaştığını ve son eşitleme noktasından bu yana yapılmış olan tüm alıkonaların ve alıkonaların geriletildiğinden emin olduğunu gösterir.

There is a description of the MQBACK call in [MQBACK](#).

## **Hareket yönetimi ve kurtarılabılır kaynak yöneticisi hizmetleri**

Hareket yönetimi ve kurtarılabılır kaynak yöneticisi hizmetleri (RRS), katılımcı kaynak yöneticilerine iki fazlı syncpoint desteği sağlamak için kullanılan bir z/OS tesidir.

Bir uygulama, IBM MQ ve Db2 gibi çeşitli z/OS kaynak yöneticileri tarafından yönetilen kurtarılabılır kaynakları güncelleyebilir ve sonra bu güncellemeleri tek bir iş birimi olarak kesinleştirebilir ya da yedekleyebilirler. RRS, olağan yürütme sırasında gerekli birim çalışma durumu günlük kaydını sağlar, syncpoint işlemini düzenler ve altsistem yeniden başlatma işlemi sırasında uygun iş birimi durumu bilgilerini sağlar.

IBM MQ for z/OS RRS katılımcısı desteği, toplu iş, TSO ve Db2 saklanmış yordam ortamlarındaki IBM MQ uygulamalarının hem IBM MQ hem de IBM MQ dışı kaynakları güncellemesini sağlar (örneğin, Db2) Tek bir mantıksal iş birimi içinde. RRS katılımcısı desteğine ilişkin bilgi için bkz. [z/OS MVS Programming: Resource Recovery](#).

IBM MQ uygulamanız, MQCMIT ve MQBACK ya da RRS çağrılarını, SRrCMT ve SRRBACK değerini kullanabilir. Ek bilgi için "[RRS toplu iş bağdaştırıcısı](#)" sayfa 925 başlıklı konuya bakın.

### RRS kullanılabilirliği

RRS z/OS sisteminizde etkin değilse, RRS sınırlı kod öbeği (CSQBRSTB ya da CSQBRSI) ile bağlantılı bir programdan yayınlanan herhangi bir IBM MQ çağrısı MQRC\_ENVIRONMENT\_ERROR değerini döndürür.

### Db2 saklı yordamlar

RRS ile Db2 saklanmış yordamları kullanıyorsanız, aşağıdaki bilgileri dikkate alın:

- RRS 'yi kullanan Db2 saklanmış yordamları, iş yükü yöneticisi (WLM-Managed) tarafından yönetilmelidir.
- Db2 tarafından yönetilen bir saklanmış yordam IBM MQ çağrılarını içeriyorsa ve bu yordam RRS sınırlı kod öbeği (CSQBRSTB ya da CSQBRSI) ile bağlantılıysa, MQCONN ya da MQCONNX çağrısı MQRC\_ENVIRONMENT\_ERROR değerini döndürür.
- WLM tarafından yönetilen bir saklanmış yordam IBM MQ çağrılarını içeriyorsa ve RRS olmayan bir sınırlı kod öbeğiyle bağlantılıysa, saklanmış yordam adres alanı başlatıldığından bu yana yürütülen ilk IBM MQ çağrısı olmadıkça, MQCONN ya da MQCONNX çağrısı MQRC\_ENVIRONMENT\_ERROR değerini döndürür.
- If your Db2 stored procedure contains IBM MQ calls and is linked with a non-RRS stub, IBM MQ resources updated in that stored procedure are not committed until the stored procedure address space ends, or until a subsequent stored procedure does an MQCMIT (using an IBM MQ Batch/TSO stub).
- Aynı saklanmış yordamın birden çok kopyası aynı adres alanında koştuzamanlı olarak yürütülebilir. Db2 'in saklanmış yordamının tek bir kopyasını kullanmasını istiyorsanız, programınızın yeniden girişli bir şekilde kodlandığından emin olun. Ters durumda, programınızdaki herhangi bir IBM MQ çağrısı için MQRC\_HCONN\_ERROR alabilirsiniz.
- WLM tarafından yönetilen bir Db2 saklanmış yordamında MQCMIT ya da MQBACK kodunu kodlamayın.
- Tüm programları Dil Ortamı (LE) içinde çalıştırmak üzere tasarlayın.

### IBM i **Syncpoints in CICS for IBM i applications**

IBM MQ for IBM i participates in CICS for IBM i units of work. You can use the MQI within a CICS for IBM i application to put and get messages inside the current unit of work.

You can use the EXEC CICS SYNCPOINT command to establish a syncpoint that includes the IBM MQ for IBM i operations. Önceki eşitleme noktasına kadar olan tüm değişiklikleri geri almak için, EXEC CICS SYNCPOINT ROLLBACK komutunu kullanabilirsiniz.

If you use MQPUT, MQPUT1, or MQGET with the MQPMO\_SYNCPOINT, or MQGMO\_SYNCPOINT, option set in a CICS for IBM i application, you cannot log off CICS for IBM i until IBM MQ for IBM i has removed its registration as an API commitment resource. Kuyruk yöneticisinden bağlantıyı kesmeden önce beklemedeki bir put ya da get işlemini kesinleştirir ya da geri al. Bu, IBM için CICS oturumunu kapatmanızı sağlar.

## Multi

### IBM MQ for Multiplatforms içindeki eşitleme noktaları

Syncpoint desteği iki tip iş birimi üzerinde çalışır: yerel ve küresel.

*yerel* iş birimi, güncellenen tek kaynakların IBM MQ kuyruk yöneticilerinden biri olduğu bir iş birimidir. Burada syncpoint eşgüdümü, kuyruk yöneticisinin tek aşamalı kesinleştirme yordamı kullanılarak sağlanıyor.

*genel* iş birimi, veritabanları gibi diğer kaynak yöneticilerine ait kaynakların da güncellendiği bir iş birimidir. IBM MQ , bu tür birimleri koordine edebilir. Ayrıca, bir dış kesinleştirme denetleyicisi tarafından da eşgüdümlü olarak kullanılabilir. Örneğin:

- Başka bir hareket yöneticisi
- **IBM i** IBM i kesinleştirme denetleyicisi

Tam bütünlük için, iki aşamalı kesinleştirme yordamı kullanın. İki aşamalı kesinleştirme, XA uyumlu hareket yöneticileri ve veritabanları tarafından sağlanabilir. Örneğin:

- TXSeries
- UDB
- **IBM i** IBM i kesinleştirme denetleyicisi

## ULW

IBM MQ ürünleri, iki aşamalı kesinleştirme süreci kullanarak genel iş birimlerini koordine edebilir.

## IBM i

IBM MQ for IBM i , bir WebSphere Application Server ortamında genel çalışma birimleri için bir kaynak yöneticisi işlevi yapabilir, ancak hareket yöneticisi olarak işlev göremezler.

## Örtük eşitleme noktası

### V9.1.0

Kalıcı iletiler yerleştirilirken IBM MQ , kalıcı iletileri eşitleme noktası altına koymak için eniyilenir. Bu uygulamalar syncpoint 'i kullanırsa, aynı kuyruğa kalıcı iletiler koyarak birden çok uygulama daha iyi performans göstermesini sağlar. Bunun nedeni, kalıcı iletileri koymak için Syncpoint kullanılırsa, kuyruk için daha az çekişme olması gerekir.

**ImplSyncOpenOutput** , uygulamalar Syncpoint 'in dışında kalıcı iletiler koyduğunda, örtük bir eşitleme noktası ekler. Bu, uygulamaların örtülü eşitleme noktasının farkında olmadan performans artışı sağlar.

Örtük eşitleme noktası, kuyruğa birden çok uygulama yerleştirirken, kuyruk için çekişmeyi azaltması nedeniyle, yalnızca bir başarımlı artışı sağlar. Bu nedenle, **ImplSyncOpenOutput** , bir örtük eşitleme noktası eklenmeden önce çıkış için kuyruk açma işlemi için gereken minimum uygulama sayısını belirtir. Varsayılan değer 2 'dir. Bu, **ImplSyncOpenOutput** belirtmezseniz, örtük eşitleme noktası yalnızca birden çok uygulama kuyruğa konursa ekleneceğini belirtir.

Ek bilgi için [Değiştirgelerin ayarlanması](#) başlıklı konuya bakın.

### Çoklu Platformlar üzerinde Yerel İş Birimleri

Yalnızca kuyruk yöneticisini içeren iş birimleri *yerel* iş birimleri olarak adlandırılır. Syncpoint eşgüdümü, tek aşamalı kesinleştirme işlemi kullanılarak kuyruk yöneticisinin kendisi (iç koordinasyon) tarafından sağlanır.

Yerel bir iş birimi başlatmak için, uygulama MQGET, MQPUT ya da MQPUT1 istekleri uygun syncpoint seçeneğini belirtmektedir. İş birimi MQCMIT kullanılarak kesinleştirilir ya da MQBACK kullanılarak geriye işlendi. Ancak, iş birimi, uygulama ile kuyruk yöneticisi arasındaki bağlantı kesildiğinde, kasıtlı olarak ya da yanlışlıkla kesildiğinde de sona erer.

If an application disconnects (MQDISC) from a queue manager while a global unit of work coordinated by IBM MQ is still active, an attempt is made to commit the unit of work. Ancak, uygulama kesilmeden sona erdirilirse, uygulama olağandışı sona ermiş olduğu için, iş birimi geri alınır.

## Çoklu Platformlar üzerinde Genel Birim Birimleri

Diğer kaynak yöneticilerine ait kaynaklara ilişkin güncellemeleri de içermeniz gerektiğinde genel iş birimlerini kullanın.

Burada, eşgüdümleme, kuyruk yöneticisinin iç ya da dış dışıyla birlikte olabilir:

## İç eşitleme noktası eşgüdümü

**Genel iş birimlerinin kuyruk yöneticisi koordinasyonu IBM MQ for IBM i ya da IBM MQ for z/OS tarafından desteklenmiyor. Bir IBM MQ MQI client ortamında desteklenmez..**

Here, IBM MQ does the coordination. Genel bir iş birimi başlatmak için, uygulama MQSTART çağrısını yayınlar.

MQBEGIN çağrısına giriş olarak, MQCONN ya da MQCONNX çağrısının döndürdüğü bağlantı tanıtıcısını (*Hconn*) sağlamanız gerekir. Bu tanıtıcı, IBM MQ kuyruk yöneticisiyle olan bağlantıyı gösterir.

Uygulama, uygun syncpoint seçeneğini belirterek MQGET, MQPUT ya da MQPUT1 isteklerine ilişkin sorunları yayınlar. Bu, yerel kaynakları, diğer kaynak yöneticilerine ait kaynakları ya da her ikisini de güncelleştiren genel bir iş birimi başlatmak için MQBEGIN komutunu kullanabilirsiniz. Diğer kaynak yöneticilerine ait kaynaklarda yapılan güncellemeler, o kaynak yöneticisinin API 'si kullanılarak yapılır. Ancak, diğer kuyruk yöneticilerine ait olan kuyrukları güncellemek için MQI ' yi kullanamazsınız. Daha fazla iş birimi (yerel ya da genel) başlatmadan önce MQCMIT ya da MQBACK komutunu verin.

Genel iş birimi MQCMIT kullanılarak kesinleştirilir; bu, iş biriminde yer alan tüm kaynak yöneticilerine ilişkin iki aşamalı kesinleştirmeyi başlatır. Kaynak yöneticilerinin (örneğin, Db2, Oracleve Sybasegibi XA uyumlu veritabanı yöneticileri) ilk olarak kesinleştirmeye hazırlanması istendiği iki aşamalı kesinleştirme işlemi kullanılır. Sadece hepsi hazırlanırsa, kesinleştirilmek isteniyorsa. Herhangi bir kaynak yöneticisi söz veremeyeceğine işaret ederse, bunun yerine her biri dışarı çıkmanız istenir. Diğer bir seçenek olarak, tüm kaynak yöneticilerine ilişkin güncellemeleri geri almak için MQBACK ' ı kullanabilirsiniz.

Genel bir iş birimi hala etkin durumdayken bir uygulama bağlantısı kesilirse (MQDISC), iş birimi kesinleştirilir. Ancak, uygulama kesilmeden sona erdirilirse, uygulama olağandışı sona ermiş olduğu için, iş birimi geri alınır.

MQBEGIN çıkışı bir tamamlanma kodu ve neden kodudur.

Genel bir iş birimi başlatmak için MQBEGIN 'i kullandığınızda, kuyruk yöneticisiyle yapılandırılmış tüm dış kaynak yöneticileri içerilir. Ancak, arama bir iş birimi başlatır, ancak aşağıdaki durumlarda bir uyarıyla tamamlanır:

- Katılımcı kaynak yöneticisi yok (yani, kuyruk yöneticisiyle hiçbir kaynak yöneticisi yapılandırılmadı).

ya da

- Bir ya da daha çok kaynak yöneticisi kullanılamıyor.

Bu durumlarda, iş biriminin yalnızca iş birimi başlatıldığında kullanılabilir olan kaynak yöneticilerine yapılan güncellemeleri içermesi gerekir.

Kaynak yöneticilerinden biri güncellemelerini kesinleştiremezse, tüm kaynak yöneticilerine güncellemelerinin geri işlenmesi bildirilir ve MQCMIT bir uyarıyla tamamlanır. Olağan dışı durumlarda (genellikle, işletmen müdahalesi) bir MQCMIT çağrısı, bazı kaynak yöneticileri güncelleştirmelerini kesinleştirirse, ancak diğerleri geri gönderirse başarısız olabilir; işin *karma* bir sonuçla tamamlandığı varsayılır. Bu tür oluşumların, kuyruk yöneticisinin hata günlüğünde tanısı konur, böylece çözüm işlemi giderilebilir.

Genel bir iş birimi MQCMIT, ilgili tüm kaynak yöneticileri güncellemelerini kesinleştirirse başarılı olur.

MQBEGIN çağrısının açıklaması için bkz. [MQBEGIN](#).

## Dış eşitleme noktası eşgüdümü

Bu durum, IBM MQ dışında bir Syncpoint eşgüdümcüsü seçildiyse oluşur; örneğin, CICS, Encina ya da Tuxedo.

Bu durumda, UNIX and Linux sistemlerinde IBM MQ ve IBM MQ for Windows , eşitleme noktası eşgüdümçüyle birlikte çalışma biriminin sonucuna olan ilgilerini kaydeder; böylece, kesinleştirilmemiş alma ya da koyma işlemlerini zorunlu olarak kesinleştirebilecekler ya da geri alabilirler. Dış eşitleme noktası eşgüdümçüsü, bir ya da iki aşamalı kesinleştirme protokollerinin sağlanıp sağlanmadığını belirler.

Bir dış eşgüdümçü kullandığınızda, MQCMIT, MQBACK ve MQBEGIN komutu yayınlanamaz. Bu işlemlere yapılan çağrılar, MQRC\_ENVIRONMENT\_ERROR neden koduyla başarısız olur.

Dışarıdan eşgüdümlü bir çalışma biriminin başlatıldığı yol, syncpoint eşgüdümçüsü tarafından sağlanan programlama arabirimine bağlıdır. Belirtik bir çağrı gerekli olabilir. Belirtik bir çağrı gerekiyorsa ve bir iş birimi başlatılmadığında MQPMO\_SYNCPOINT seçeneğini belirten bir MQPUT çağrısı yayınladığınızda, MQRC\_SYNCPOINT\_NOT\_AVAM tamamlanma kodu döndürülür.

İş biriminin kapsamı, syncpoint eşgüdümçüsü tarafından belirlenir. Uygulama ile kuyruk yöneticisi arasındaki bağlantının durumu, iş biriminin durumu değil, bir uygulama sorunu olan MQI çağrılarının başarısını ya da başarısızlığı etkiler. Örneğin, bir uygulama, etkin bir iş birimi sırasında bir kuyruk yöneticisine bağlantı kesip yeniden bağlayabilir ve aynı iş birimi içinde daha fazla MQGET ve MQPUT işlemleri gerçekleştirebilirler. Bu, beklemedeki bir bağlantı kesme işlemi olarak bilinir.

You can use IBM MQ API calls in CICS programs, whether you choose to use the XA abilities of CICS. XA kullanmazsanız, kuyruklardan ve kuyruklardan ileti alımları CICS atomik iş birimleri içinde yönetilmez. Bu yöntemi seçmenin bir nedeni, iş biriminin genel tutarlılığın sizin için önemli olmamasıdır.

İş birimlerinin bütünlüğü sizin için önemliyse, XA ' yı kullanmanız gerekir. XA ' yı kullandığınızda, CICS , iş birimi içindeki tüm kaynakların birlikte güncellendiğinden emin olmak için iki aşamalı bir kesinleştirme protokolü kullanır.

İşlem desteğinin ayarlanmasıyla ilgili daha fazla bilgi için bkz. [Transactional support scenarios](#) ve ayrıca TXSeries CICS belgeleri, örneğin, *TXSeries for Multiplatforms CICS Administration Guide for Open Systems*.

**V 9.1.0** **Multi** Çoklu Platformlar üzerindeki örtük Syncpoint

Örtük eşitleme noktası desteği, kalıcı iletinin uyumluluk noktasının dışına yerleştirilmesini sağlar.

Kalıcı iletiler yerleştirilirken IBM MQ , kalıcı iletileri eşitleme noktası altına koymak için eniyilenir. Eşzamanlı iletileri aynı kuyruğa eşzamanlı olarak koyan birden çok uygulama, bu uygulamalar syncpoint 'i kullanırsa, genellikle daha iyi performans sağlar. This is because the locking strategy of IBM MQ is more efficient if syncpoint is used when putting persistent messages.

The **ImplSyncOpenOutput** parameter in the qm . ini file, controls whether an implicit syncpoint can be added when applications put persistent messages outside of syncpoint. Bu, örtük eşitleme noktasının farkında olan uygulamalar olmadan, bir performans artışı sağlayabilir.

Örtülü eşitleme noktası, kilit çekişmesini azalttığından, kuyruğa koşut zamanlı olarak birden çok uygulama yerleştirildiğinde performans artımı sağlar. **ImplSyncOpenOutput** , örtülü bir eşitleme noktası eklenmeden önce çıkış için kuyruk açık olan uygulama sayısı alt sınırını belirtir. Varsayılan değer 2' dir. Bu, **ImplSyncOpenOutput** seçeneğini belirttik olarak belirtmezseniz, örtük eşitleme noktası yalnızca birden çok uygulama kuyruğa konursa eklenir.

Örtük bir eşitleme noktası eklerseniz, istatistikler bunu yansıtır ve **runmqsc display conn**' dan bir işlem çıkışı görebilirsiniz.

Örtük bir syncpoint eklenmesini istemiyorsanız, **ImplSyncOpenOutput=OFF** seçeneğini ayarlayın.

Ek bilgi için [Değiştirelerin ayarlanması](#) başlıklı konuya bakın.

*Çoklu Platformlar üzerindeki Dış Syncpoint Yöneticilerine Arabirimler*

IBM MQ for Multiplatforms , X/Open XA arabirimini kullanan dış eşitleme noktası yöneticilerine göre işlemlerin koordinasyonunu destekler.

Bazı XA hareket yöneticileri (TXSeries), her XA kaynak yöneticisinin adını sağladığından emin olun. Bu, XA anahtarı yapısındaki name adlı dizedir.



- **ULW** UNIX, Linux, and Windows üzerindeki IBM MQ kaynak yöneticisi, MQSeries\_XA\_RMI olarak adlandırılıyor.
- **IBM i** IBM için, kaynak yöneticisi adı MQSeries XA RMI 'tır.

XA arabirimleriyle ilgili ek ayrıntılar için, Open Group tarafından yayınlanan XA belgelerine *CAE Specification Distributed Transaction Processing: The XA Specification* bakın.

Bir XA yapılışındaki IBM MQ for Multiplatforms , XA kaynak yöneticisinin rolünü yerine getirir. Bir XA syncpoint eşgüdümçüsü bir XA kaynak yöneticisi kümesini yönetebilir ve kesinleştirme ya da geri alma işlemini her iki kaynak yöneticisinde de uyumlulaştırır. Bu, statik olarak kayıtlı bir kaynak yöneticisi için bu şekilde çalışır:

1. Bir uygulama, bir hareket başlatmak istediğini eşitleme noktası eşgüdümçü'ine bildirir.
2. Syncpoint eşgüdümçüsü, yürürlükteki işlemi bildirmek için tanıdığı kaynak yöneticilerine çağrı gönderir.
3. Uygulama sorunları, yürürlükteki işlemle ilişkilendirilmiş kaynak yöneticileri tarafından yönetilen kaynakları güncellemek için çağrılar.
4. Uygulama, syncpoint koordinatörünün hareketi kesinleştirmesini ya da geri yüklemesini ister.
5. syncpoint eşgüdümçüsü, her kaynak yöneticisine, işlemi istenen şekilde tamamlamak için iki aşamalı kesinleştirme protokolleri kullanarak çağırır.

XA belirtimi, her kaynak yöneticisinin XA Anahtarı adı verilen bir yapı sağlamasını gerektirir. Bu yapı, kaynak yöneticisinin yeteneklerini ve Syncpoint eşgüdümçüsü tarafından çağrılacak işlevleri bildirir.

Bu yapının iki sürümü vardır:

Çizelge 132. XA Anahtarı Sürümleri	
S\u00fcr\u00fcm	Tanım
MQRMIASwitch	Durağan XA kaynak yönetimi
MQRMIASwitchDynamic	Devingen XA kaynak yönetimi

Bu yapıyı içeren kitaplıkların listesi için [IBM MQ XA anahtar yapısı](#) başlıklı konuya bakın.

Bunları bir XA syncpoint eşgüdümçüyle bağlamak için kullanılması gereken yöntem, eşgüdümçüyle tanımlanır; bu eşgüdümçü tarafından sağlanan belgelere bakın ve XA syncpoint eşgüdümçünüz ile işbirliği yapmak için IBM MQ ' in nasıl etkinleştirileceğini saptayın.

Syncpoint eşgüdümçüsü tarafından herhangi bir *xa\_open* çağrısında geçirilen *xa\_info* yapısı, denetlenmek üzere olan kuyruk yöneticisinin adı olabilir. Bu, MQCONN ya da MQCONNX 'e geçirilen kuyruk yöneticisi adıyla aynı formu alır ve varsayılan kuyruk yöneticisi kullanılacaksa boş bırakılabilir. Ancak, TPM ve AXLIB iki parametre parametresini kullanabilirsiniz.

TPM, hareket yöneticisi adını IBM MQ olarak belirtmenizi sağlar; örneğin, CICS. AXLIB, XA AX giriş noktalarının bulunduğu hareket yöneticisinde gerçek kitaplık adını belirtmenizi sağlar.

Bu deęiřtirgelerden birini ya da varsayılan olmayan bir kuyruk yöneticisini kullanırsanız, QMNAME parametresini kullanarak kuyruk yöneticisi adını belirtmeniz gerekir. Daha fazla bilgi için bkz. [xa\\_open](#) dizisinin CHANNEL, TRPTYPE, CONNAME ve QMNAME parametreleri.

## Kısıtlamalar

1. Paylaşılan bir Hconn ( "Shared (thread independent) connections with MQCONNX" sayfa 778' ta açıklandığı gibi) ile genel iş birimlerinin kullanılmasına izin verilmez.
2. **IBM i** IBM MQ for IBM i , XA kaynak yöneticilerinin dinamik kaydını desteklemez.  
Desteklenen tek hareket yöneticisi 'WebSphere Application Server' dir.
3. **Windows** Windows sistemlerinde, XA anahtarında bildirilen tüm işlevler \_cdecl işlevleri olarak bildirilir.

4. Bir dış eşitleme noktası eşgüdümçüsü, aynı anda yalnızca bir kuyruk yöneticisini yönetebilir. Bunun nedeni, eşgüdümçünün her kuyruk yöneticisinde etkili bir bağlantı olduğundan ve bu nedenle bir kerede tek bir bağlantıya izin verildiğine ilişkin kuralın da söz konusu olduğu içindir.

**Not:** Not: Bir JEE sunucusunda çalışan bir JMS istemci uygulaması (CLIENT JEE uygulaması) bu kısıtlamaya sahip değildir; bu nedenle, tek bir JEE sunucu yönetimli bir hareket, aynı hareket içinde birden çok kuyruk yöneticisini koordine edebilir. Ancak, bağ tanımları kipinde çalışan bir JMS sunucusu uygulaması, bir kerede yalnızca bir bağlantıya izin verilen kuralın konusuna tabi olmaya devam eder.

5. Syncpoint eşgüdümçüsü kullanılarak çalıştırılan tüm uygulamalar, yalnızca bu kuyruk yöneticisine etkin bir şekilde bağlandıkları için, eşgüdümçünün yönettiği kuyruk yöneticisine bağlanabilir. Bir bağlantı tanıtıcısı almak için MQCONN ya da MQCONNX yayınlmalıdır ve çıkış işleminden önce MQDISC yayınlmalıdır. Diğer bir seçenek olarak, bunlar TXSeries CICS için UE014015 çıkışını kullanabilir.

## IBM i IBM i dış eşitleme noktası yöneticisine arabirimler

IBM MQ for IBM i, yerel IBM i kesinleştirme denetimini dış eşitleme noktası eşgüdümçüsü olarak kullanabilir.

İş parçacığı bağımsız (paylaşılan) bağlantılara, kesinleştirme denetimi ile izin verilmez. IBM i' in kesinleştirme denetimi yetenekleriyle ilgili daha fazla bilgi için *IBM i Programming: Backup and Recovery Guide, SC21-8079* adlı yayına bakın.

IBM i kesinleştirme denetimi olanaklarını başlatmak için STRCMTCTL sistem komutunu kullanın. Kesinleştirme denetimini sona erdirmek için ENDCMTCTL sistem komutunu kullanın.

**Not:** Varsayılan değer olan *Kesinleştirme tanımlaması kapsamı* \*ACTGRP değeridir. This must be defined as \*JOB for IBM MQ for IBM i. Örneğin:

```
STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
```

IBM MQ for IBM i, yalnızca IBM MQ kaynakları için güncellemeler içeren yerel iş birimlerini de gerçekleştirebilir. Uygulama MQPUT, MQPUT1 ya da MQGET olduğunda, MQPMO\_SYNCPOINT ya da MQGMO\_SYNCPOINT ya da MQBEGIN belirtildiğinde, her uygulamada yerel iş birimleri ve IBM i tarafından eşgüdümlü çalışma genel birimlerine katılım arasında seçim yapılır. Bu tür bir çağrı yayınlandığında kesinleştirme denetimi etkin değilse, IBM MQ yerel bir iş birimi başlatır ve IBM MQ ' e bu bağlantı için tüm iş birimleri de, kesinleştirme denetimi başlatılıp başlatılmadığından bağımsız olarak yerel iş birimlerini de kullanır. Yerel iş birimi kesinleştirmek için MQCMIT ' yi kullanın. Yerel bir iş birimini yedeklemek için, MQBACK ' ı kullanın. CL komutu COMMIT gibi IBM i kesinleştirme ve geri alma çağrıları, IBM MQ yerel iş birimleri üzerinde herhangi bir etkiye sahip değildir.

If you want to use IBM MQ for IBM i with native IBM i commitment control as an external syncpoint coordinator, ensure that any job with commitment control is active and that you are using IBM MQ in a single-threaded job. Kesinleştirme denetiminin başlatıldığı çok iş parçacıklı bir işte MQPMO\_SYNCPOINT ya da MQGMO\_SYNCPOINT belirtilerek MQPUT ögesini ya da MQGET ögesini çağırırsanız, çağırma işlemi MQRC\_SYNCPOINT\_NOT\_AVAM neden kodlarıyla başarısız olur.

Çok iş parçacıklı bir işte yerel iş birimleri ve MQCMIT ve MQBACK çağrıları kullanmak mümkündür.

MQPUT, MQPUT1 ya da MQGET ' i çağırırsanız, kesinleştirme denetimini başlattıktan sonra MQPMO\_SYNCPOINT ya da MQGMO\_SYNCPOINT belirtilerek, IBM MQ for IBM i kendisini kesinleştirme tanımına bir API kesinleştirme kaynağı olarak ekler. Bu, genellikle bir işteki ilk çağrıdır. Belirli bir kesinleştirme tanımı altında kayıtlı herhangi bir API kesinleştirme kaynağı varsa, bu tanım için kesinleştirme denetimini sona erdiremezsiniz.

IBM MQ for IBM i, yürürlükteki iş biriminde bekleyen bir MQI işlemi yoksa, kuyruk yöneticisiyle bağlantınızı kestiğinizde API kesinleştirme kaynağı olarak kaydını kaldırır.

Yürürlükteki iş biriminde bekleyen MQPUT, MQPUT1 ya da MQGET işlemleri varken kuyruk yöneticisinden bağlantıyı kesiyorsanız, IBM MQ for IBM i, bir sonraki kesinleştirme ya da geriye işleme işleminin bildirmesi için bir API kesinleştirme kaynağı olarak kayıtlı kalır. Bir sonraki eşitleme noktasına ulaşıldığında IBM MQ for IBM i, değişiklikleri gerektiği şekilde kesinleştirir veya geri alır. Bir uygulama,

etkin bir iş birimi sırasında bir kuyruk yöneticisine bağlantı kesebilir ve yeniden bağlanabilir ve aynı iş birimi içinde daha fazla MQGET ve MQPUT işlemleri gerçekleştirebilir (bu, beklemedeki bir kopuktur).

Söz konusu kesinleştirme tanımı için bir ENDCMTCTL sistem komutu yayınlamaya çalışırsanız, bekleyen değişikliklerin etkin olduğunu belirten CPF8355 iletisi yayınlanır. Bu ileti, iş sona erdiğinde iş günlüğünde de görüntülenir. Bunu önlemek için, beklemedeki tüm IBM MQ for IBM i işlemlerini kesinleştirin ya da geriye işlet ve kuyruk yöneticisinden bağlantıyı kesin. Bu nedenle, ENDCMTCTL öncesi COMMIT ya da ROLLBACK komutlarının kullanılması, son kesinleştirme denetiminin başarıyla tamamlanmasına olanak sağlar.

When you use IBM i commitment control as an external syncpoint coordinator, you cannot issue MQCMIT, MQBACK, and MQBEGIN calls. Bu işlemlere yapılan çağrılar, MQRC\_ENVIRONMENT\_ERROR neden koduyla başarısız olur.

Geri almak ya da geri almak (yani, geri dönmek) için, iş biriminiz, kesinleştirme denetimini destekleyen programlama dillerinden birini kullanın. Örneğin:

- CL komutları: COMMIT ve ROLLBAC
- ILE C Programming İşlevleri: \_Rcommit ve \_Rollback
- ILE RPG: COMMIT ve ROLBK
- COBOL/400: COMMIT ve ROLLBAC

When you use IBM i commitment control as an external syncpoint coordinator with IBM MQ for IBM i, IBM i performs a two-phase commit protocol in which IBM MQ participates. Her iş birimi iki aşamada kesinleştirildiğinden, kuyruk yöneticisi ilk aşamada kesinleştirme oyu verdikten sonra ikinci aşama için kullanılamaz duruma gelebilir. Örneğin, kuyruk yöneticisinin iç işleri sona erdirildiyse, bu durum oluşabilir. Bu durumda, kesinleştirmeyi gerçekleştiren iş günlüğü, kesinleştirme ya da geriye işleme işleminin başarısız olduğunu gösteren CPF835F iletisini içerir. Bu iletiden önceki iletiler sorunun nedenini, kesinleştirme ya da geridönüş işlemi sırasında oluşup oluşmadığını ve başarısız iş birimi için mantıksal iş birimi tanıtıcısının (LUWID) ortaya çıktığını belirtir.

Sorun, hazırlanmış bir iş biriminin kesinleştirme ya da geridönüş işlemi sırasında IBM MQ API kesinleştirme kaynağının hatasından kaynaklandıysa, işlemi tamamlamak ve hareketin bütünlüğünü geri yüklemek için WRKMQMTRN komutunu kullanabilirsiniz. Komut, kesinleştirmek ve geri dönmek için iş biriminin LUWID ' yi bilmenizi gerektirir.

## Starting IBM MQ applications using triggers

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

Kuyruklara hizmet veren bazı IBM MQ uygulamaları sürekli olarak çalışır, bu nedenle kuyruklara gelen iletileri almak için her zaman kullanılabilir olur. Ancak, kuyruklara gelen ileti sayısı tahmin edilemez olduğunda bunu istemeyebilirsiniz. Bu durumda, alınacak herhangi bir ileti olmadığında da, uygulamalar sistem kaynaklarını tüketebilirler.

IBM MQ , alınabilmekte kullanılabilir iletiler olduğunda, uygulamanın otomatik olarak başlatılmasına olanak sağlayan bir olanak sağlar. Bu olanak, *tetikleme* olarak bilinir.

Kanalların tetiklenmesine ilişkin bilgi edinmek için [Tetikleme kanalları](#) başlıklı konuya bakın.

## Tetikleme nedir?

Kuyruk yöneticisi, *tetikleme olayları* ' u oluşturan bazı koşulları tanımlar.

Bir kuyruk için tetikleme etkinleştirilirse ve bir tetikleme olayı ortaya çıkarsa, kuyruk yöneticisi *tetikleyici ileti* adlı bir kuyruğa *başlatma kuyruğu* gönderir. Başlatma kuyruğunda tetikleme iletisinin varlığı, bir tetikleme olayının ortaya çıktığını gösterir.

Kuyruk yöneticisi tarafından oluşturulan tetikleme iletileri kalıcı değil. Bu, günlüğe kaydetme işlemi azaltır (performansı iyileştirmeye) ve yeniden başlatma sırasında yinelemeleri en aza indirerek yeniden başlatma süresini iyileştirir.

Başlatma kuyruğunu işleyen program *tetikleme-izleyici uygulaması* olarak adlandırılır ve tetikleme iletilerinde bulunan bilgilere dayalı olarak tetikleme iletilerini okuyup uygun işlemi yapmak için bu işlevi görmeniz gerekir. Genellikle bu işlem, tetikleme iletilerini oluşturan kuyruğu işlemek için başka bir uygulamayı başlatmaya başlamanız gerekir. Kuyruk yöneticisinin bakış açısından, tetikleme izleme uygulamasıyla ilgili özel bir şey yoktur; bir kuyruktan ileti okuyan başka bir uygulamadır (başlangıç kuyruğu).

Bir kuyruk için tetikleme etkinleştirilirse, bu tetikle ilişkilendirilmiş bir *süreç tanımlaması nesnesi* yaratabilirsiniz. Bu nesne, tetikleme olayına neden olan iletiyi işleyen uygulamaya ilişkin bilgileri içerir. Süreç tanımlaması nesnesi yaratıldıysa, kuyruk yöneticisi bu bilgileri alır ve tetikleme izleme uygulaması tarafından kullanılmak üzere tetikleyici iletilerine yerleştirir. Bir kuyrukla ilişkilendirilmiş süreç tanımlamasının adı, *ProcessName* yerel kuyruk özneliği 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 tanımlamanıza gerek yoktur. Bunun yerine iletim kuyruğu tanımlaması kullanılır.

Tetikleme, UNIX, Linux, and Windows üzerinde çalışan IBM MQ istemcileri tarafından desteklenir. İstemci ortamında çalışan bir uygulama, istemci kitaplıklarıyla bağlantı verdiğinizde, tam IBM MQ ortamında çalışan bir uygulama ile aynıdır. Ancak, tetikleme izleme programı ve başlatılacak uygulamanın her ikisi de aynı ortamda olmalıdır.

Tetikleme şunları içerir:

### **Uygulama kuyruğu**

*Uygulama kuyruğu* , tetikleme tetikleyicisi olduğunda ve koşullar karşılandığında tetikleme iletilerinin yazılmasını gerektiren bir yerel kuyruklardır.

### **Süreç tanımlaması**

Uygulama kuyruğunda, uygulama kuyruğundan ileti alacak uygulamanın ayrıntılarını içeren bir *süreç tanımlaması nesnesi* ilişkilendirilebilir. (Özneliklerin listesi için [Süreç tanımlamalarına ilişkin öznelikler](#) konusuna bakın.)

**Bir tetikleyicinin bir kanalı başlatmasını istiyorsanız, bir süreç tanımlaması nesnesi tanımlamanıza gerek olmadığını unutmayın.**

### **İletim kuyruğu**

**Tetikleyicinin bir kanal başlatmasını istiyorsanız, iletim kuyruğuna gereksinim duyarsınız.**

Linux dışındaki bir altyapıda iletim kuyruğu için, iletim kuyruğunun *TriggerData* özneliği başlatılacak kanalın adını belirtebilir. Bu işlem, tetikleme kanallarına ilişkin süreç tanımlamasını değiştirebilir, ancak yalnızca bir süreç tanımlaması yaratılmadığında kullanılır.

### **Tetikleme olayı**

*Tetikleme olayı* , kuyruk yöneticisi tarafından tetikleme iletilerinin oluşturulmasına neden olan bir olaydır. Bu genellikle bir uygulama kuyruğuna gelen bir ilettir, ancak diğer zamanlarda bu ileti de oluşabilir. Örneğin, bkz. [“Tetikleme olayına ilişkin koşullar”](#) sayfa 906.

IBM MQ has a range of options to allow you to control the conditions that cause a trigger event (see [“Tetikleme olaylarını denetleme”](#) sayfa 910 ).

### **Tetikleme iletileri**


Kuyruk yöneticisi, bir tetikleme olayını tanıdığı anda bir *tetikleyici ileti* yaratır. Başlatılacak uygulamayla ilgili tetikleyici ileti bilgilerine kopyalanır. Bu bilgiler, uygulama kuyruğundan ve uygulama kuyruğuyla ilişkili süreç tanımlaması nesnesinden gelir.

Tetikleme iletileri sabit bir biçime sahiptir (bkz. [“Tetikleyici İletilerinin Biçimi”](#) sayfa 917 ).

### **Başlatma kuyruğu**

*Kullanıma hazırlama kuyruğu* , kuyruk yöneticisinin tetikleme iletilerini yerleştirdiği yerel bir kuyruğdur. Bir başlatma kuyruğunun diğer ad kuyruğu ya da model kuyruğu olamayacağı unutulmalıdır.

Kuyruk yöneticisi birden çok kullanıma hazırlama kuyruğuna sahip olabilir ve her biri bir ya da daha çok uygulama kuyruklarıyla ilişkilendirilir.

 Kuyruk paylaşım grubundaki kuyruk yöneticileri tarafından erişilebilen bir yerel kuyruk olan paylaşılan kuyruk, IBM MQ for z/OS üzerinde bir başlatma kuyruğu olabilir.

## Tetikleyici İzleme Programı

*Tetikleme izleme programı*, bir ya da daha çok kullanıma hazırlama kuyruğuna hizmet veren sürekli çalışan bir programdır. Bir tetikleme iletisi bir başlatma kuyruğuna ulaştığında, tetikleme izleme programı iletisi alır. Tetikleyici izleyicisi, tetikleme iletisinde yer alan bilgileri kullanır. Uygulama kuyruğuna gelen iletileri almak için, uygulama kuyruğunun adını içeren tetikleyici ileti üstbilgisinde yer alan bilgileri aktarmak için bir komut verir.

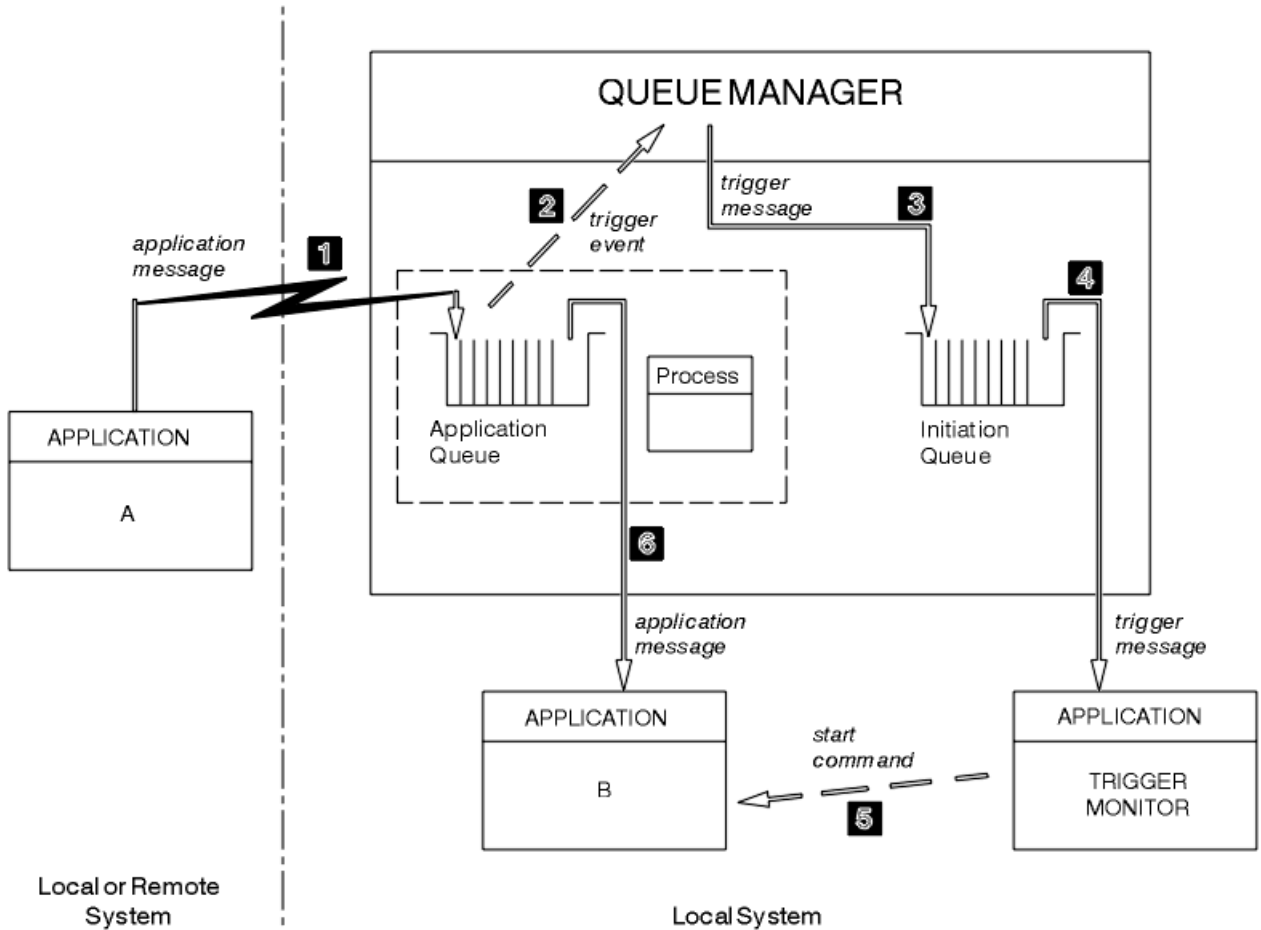
Tüm altyapılarda, kanal başlatıcı olarak bilinen özel bir tetikleme izleme programı, kanalların başlatılmasından sorumlu olur.

**z/OS** z/OS üzerinde, kanal başlatıcı genellikle manuel olarak başlatılır ya da kuyruk yöneticisi, kuyruk yöneticisi başlatma JCL 'de CSQINP2 'yi değiştirerek otomatik olarak başlatılabilir.

**Multi** Multiplatforms üzerinde, kanal başlatıcısı, kuyruk yöneticisi başlatıldığında otomatik olarak başlatılır ya da **runmqchi** komutuyla el ile başlatılabilir.

Daha fazla bilgi için bkz “Tetikleme izleyicileri tarafından kullanıma hazırlama kuyruğu işleme” sayfa 913.

Tetikleyicinin nasıl çalıştığını anlamak için, FIRST (MQTT\_FIRST) tetikleme tipini gösteren bir örnek olan Şekil 100 sayfa 901' i göz önünde bulundurun.



Şekil 100. Uygulama akışı ve tetikleme iletileri

Şekil 100 sayfa 901 içinde olayların sırası şöyledir:

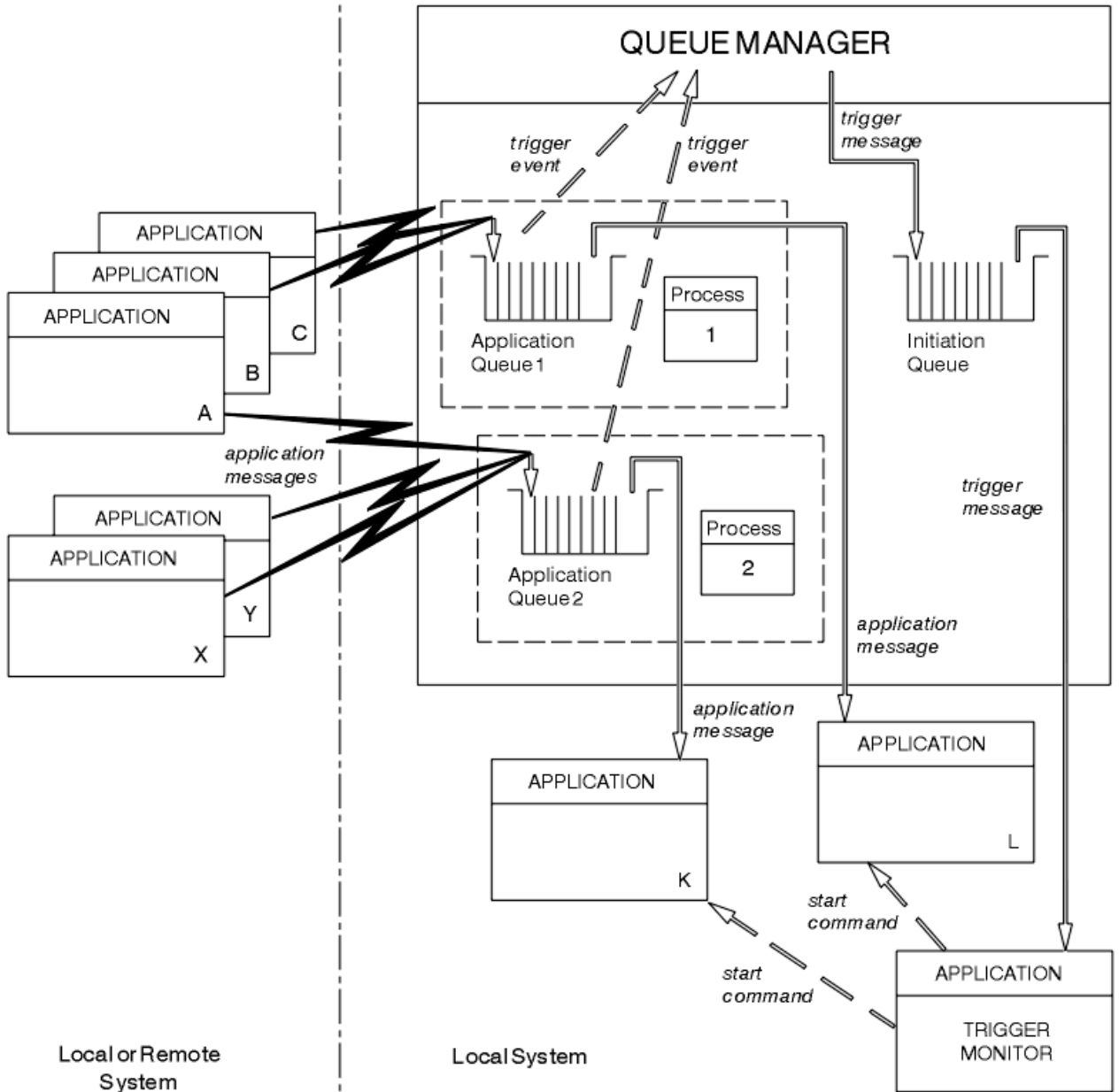
1. Yerel ya da kuyruk yöneticisinde uzak olabilen Uygulama A, bir iletisi uygulama kuyruğuna yerleştirir. Bu kuyruk giriş için açık olan bir uygulama yok. Ancak, bu olgu yalnızca FIRST VE DEPTH tipini tetiklemek için geçerlidir.

2. Kuyruk yöneticisi, bir tetikleme olayı oluşturmak için sahip olduğu koşulların yerine getirilip karşılanmıyorsa, bunları görmek için denetler. Bunlar, bir tetikleme olayı oluşturulur. Tetikleme iletisi yaratılırken, ilişkili süreç tanımlaması nesnesi içinde tutulan bilgiler kullanılır.
3. Kuyruk yöneticisi bir tetikleme iletisi yaratır ve bu iletiyi bu uygulama kuyrukla ilişkilendirilmiş başlatma kuyruğuna koyar, ancak yalnızca bir uygulama (tetikleme izleme programı) giriş için açık kullanıma açma kuyruğu varsa, bu ileti kuyruğunu açar.
4. Tetikleyici izleyicisi, tetikleme iletisini başlatma kuyruğundan alır.
5. Tetikleme izleme programı, B uygulamasını (sunucu uygulaması) başlatmak için bir komut verir.
6. B uygulaması, uygulama kuyruğunu açar ve iletiyi alır.

**Not:**

1. Uygulama kuyruğu giriş için, herhangi bir program tarafından açılırsa ve FIRST ya da DEPTH için tetikleme tetiklemesi varsa, kuyruk zaten sunulmakta olduğu için tetikleme olayı oluşmaz.
2. Başlangıç kuyruğu giriş için açılmamışsa, kuyruk yöneticisi tetikleyici ileti üretmez; bir uygulama giriş için başlatma kuyruğunu açıncaya kadar bekler.
3. Kanallar için tetikleme kullanırken, FIRST ya da DEPTH tetikleyici tipini kullanın.
4. Tetiklenen uygulamalar, tetikleme izleyicisini başlatan kullanıcının kullanıcı kimliği ve grubu altında, CICS kullanıcısı ya da kuyruk yöneticisini başlatan kullanıcı tarafından tetiklenir.

Şu ana kadar, tetikleme içindeki kuyruklar arasındaki ilişki sadece tek bir temele dayandı. Şekil 101 sayfa 903 seçeneğini dikkate alın.



Şekil 101. Tetikleme içindeki kuyrukların ilişkisi

Uygulama kuyruğunda, iletiyi işleyecek uygulamanın ayrıntılarını içeren bir süreç tanımlaması nesnesi ilişkilendirilir. Kuyruk yöneticisi bilgileri tetikleme iletisine yerleştirir, bu nedenle yalnızca bir başlatma kuyruğu gereklidir. Tetikleme izleme programı bu bilgileri tetikleme iletisinden çıkarır ve her bir uygulama kuyruğunda iletiyle başa çıkmak için ilgili uygulamayı başlatır.

Bir kanalın başlangıcını tetiklemek istiyorsanız, bir süreç tanımlaması nesnesi tanımlamanıza gerek olmadığını unutmayın. İletim kuyruğu tanımlaması, tetiklenecek kanalı saptayabilir.

Use the following links to find out more about starting IBM MQ applications using triggers:

- [“Tetikleme için önkoşullar” sayfa 904](#)
- [“Tetikleme olayına ilişkin koşullar” sayfa 906](#)
- [“Tetikleme olaylarını denetleme” sayfa 910](#)
- [“Tetiklenen kuyrukları kullanan bir uygulama tasarlanması” sayfa 912](#)
- [“Tetikleme izleyicileri tarafından kullanıma hazırlama kuyruğu işleme” sayfa 913](#)
- [“Tetikleme İletilerinin Özellikleri” sayfa 916](#)

- [“Tetikleme işe yaramadığında” sayfa 918](#)

## İlgili kavramlar

[“Message Queue Interface-Genel Bakış” sayfa 760](#)

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabılır alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OSüzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

[“IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

## 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 AMQSQNQA */  
CRTMQMQ QNAME('SYSTEM.SAMPLE.INQ') +  
QTYPE(*LCL) REPLACE(*YES) +  
MQMNAME +  
TEXT('queue for AMQSQNQA') +  
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 AMQSQNQA') +  
ENVDATA('JOBPTY(3)') /* Submit parameter */+  
APPID('AMQSQNQA') /* 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
UNIX, Linux, and Windows sistemleri	DEFINE PROCESS ya da SYSTEM.DEFAULT.PROCESS ve ALTER PROCESS komutunu kullanarak değiştirin
 z/OS  z/OS	DEFINE PROCESS kullanın (adım “3” sayfa 905 içindeki örnek koda bakın) ya da işlemleri ve denetim panolarını kullanın.
 IBM i  IBM i	“3” sayfa 905. 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/OS'de, 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
UNIX, Linux, and Windows sistemleri	ALTER QLOCAL (YEREL)
  z/OS	ALTER QLOCAL (YEREL)
  IBM i	CHGMQM

6. Tanımladığınız başlatma kuyruklarına hizmet verecek tetikleyici izleme programlarının  (ya da IBM MQ for IBM i içindeki tetikleyici sunucularının) eşgörünümlerini başlatın. Ek bilgi için bkz. [“Tetikleme izleyicileri tarafından kullanıma hazırlama kuyruğu işleme” sayfa 913](#) .

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 [“Tetikleme olaylarını denetleme” sayfa 910](#) başlıklı konuya bakın.

### **Tetikleme olayına ilişkin koşullar**

Kuyruk yöneticisi, bu konuda ayrıntılı olarak açıklanan koşullar yerine getirildiğinde bir tetikleyici iletisi yaratır.

Bu konuda paylaşılan kuyruklara yapılan başvurular, yalnızca IBM MQ for z/OS üzerinde kullanılabilir olan bir kuyruk paylaşım grubunda paylaşılan kuyruklar anlamına gelir.

Aşağıdaki koşullar kuyruk yöneticisinin bir tetikleyici iletisi yaratmasına neden olur:

1. Bir ileti, bir kuyruğa *yerleştirilir* .
2. İleti, kuyruğun eşik tetikleme önceliğine eşit ya da bu değere eşit bir öncelik içeriyor. Bu öncelik, **TriggerMsgPriority** yerel kuyruk öznitelikinde ayarlanır; sıfır olarak ayarlanmışsa, herhangi bir ileti nitelenmektedir.
3. The number of messages on the queue with priority greater than or equal to *TriggerMsgPriority* was previously, depending on *TriggerType*:
  - Sıfır (tetikleme tipi MQTT\_FIRST için)
  - Herhangi bir sayı (tetikleme tipi MQTT\_EVERY tetikleyicisi için)
  - *TriggerDepth* eksi 1 (MQTT\_DEPTH tetikleme tipi için)

#### **Not:**

- a. Paylaşılmayan yerel kuyruklar için, kuyruk yöneticisi, bir tetikleme olayına ilişkin koşulların var olup olmadığını değerlendirirken hem kesinleştirilmiş hem de kesinleştirilmemiş iletiler sayılır. Sonuç olarak, kuyruktaki iletiler kesinleştirilmediği için, bu uygulamanın alması gereken iletiler olmadığında bir uygulama başlatılmış olabilir. Bu durumda, uygulamanın iletilerin gelmesini beklemesi için uygun bir *WaitInterval* ile bekleme seçeneğini kullanmayı düşünün.
  - b. Yerel paylaşılan kuyruklar için, kuyruk yöneticisi yalnızca kesinleştirilmiş iletileri sayar.
4. Birinci ya da DEPTH tipini tetiklemek için, hiçbir programın ileti kaldırmak için uygulama kuyruğu açılmadı (yani, **OpenInputCount** yerel kuyruk özniteliği sıfır).

#### **Not:**

- a. Paylaşılan kuyruklar için, birden çok kuyruk yöneticisi bir kuyruğa karşı çalışan izleme programlarını tetiklediğinde özel koşullar uygulanır. Bu durumda, bir ya da daha çok kuyruk yöneticisi giriş paylaşılan girişi için açık olursa, diğer kuyruk yöneticilerindeki tetikleme ölçütleri *TriggerType* MQTT\_FIRST ve *TriggerMsgPriority* sıfır olarak değerlendirilir. Tüm kuyruk yöneticileri giriş için kuyruğu kapattığında, tetikleme koşulları kuyruk tanımlamasında belirtilen koşullara geri çevrilir.

Bu koşulun etkilediği bir örnek, uygulama kuyruğu için çalışan bir tetikleme izleme programı olan birden çok kuyruk yöneticisi QM1, QM2 ve QM3 ' dir. Bir ileti, tetikleme için gerekli koşulları yerine getirdiğinde, başlatma kuyruğunda bir tetikleyici iletili oluşturulur. QM1 üzerindeki tetikleyici izleyicisi tetikleme iletilisini alır ve bir uygulamayı tetikler. Tetiklenen uygulama, paylaşılan giriş için uygulama kuyruğunu açar. From this point on the trigger conditions for application queue A are evaluated as *TriggerType* MQTT\_FIRST, and *TriggerMsgPriority* zero on queue managers QM2 and QM3, until QM1 closes the application queue.

- b. Paylaşılan kuyruklar için, bu koşul her kuyruk yöneticisi için geçerli olur. Yani, kuyruk yöneticisinin kuyruk yöneticisi tarafından kuyruğa alma işlemi için queue kuyruk yöneticisinin kuyruğun *OpenInputCount* kuyruk yöneticisi tarafından oluşturulacağı bir kuyruk için sıfır olması gerekir. Ancak, kuyruk paylaşım grubundaki herhangi bir kuyruk yöneticisinde MQOO\_INPUT\_EXCLUSIVE seçeneğini kullanarak kuyruk açıksa, kuyruk paylaşım grubundaki kuyruk yöneticilerinden herhangi biri tarafından bu kuyruk için herhangi bir tetikleme iletili oluşturulmaz.

Tetiklenen uygulama, giriş için kuyruğu açtığı anda, tetikleme koşullarının nasıl değerlendirileceğini değiştirmek için bu değişikliği kullanın. Yalnızca tek bir tetikleme izleyicinin çalıştığı senaryolarda, diğer uygulamalar aynı etkiyi yapabilir, çünkü benzer şekilde giriş için uygulama kuyruğunu açarlar. Uygulama kuyruğunun, tetikleme izleme programı tarafından başlatılan bir uygulama tarafından mı, yoksa başka bir uygulama tarafından mı açıldığı önemli değildir; bu, tetikleme ölçütlerinde değişikliğe neden olan başka bir kuyruk yöneticisinde giriş için kuyruğun açık olduğu gerçektir.

5. IBM MQ for z/OS üzerinde, uygulama kuyruğu **Usage** MQUS\_NORMAL özniteliğinin bulunduğu bir kuyruksa, bunun engellenmemesi için istek alın (yani, **InhibitGet** kuyruk özniteliği MQQA\_GET\_ALLOWALIZD olur). Ayrıca, tetiklenen uygulama kuyruğu, MQUS\_XMITQ **Usage** özniteliğine sahip bir kuyruksa, istek alma işlemi yapılamaz.

6. Aşağıdakilerden birini yapın:

- Kuyruğa ilişkin **ProcessName** yerel kuyruk özniteliği boş değil ve o özniteliğe ilişkin tanımlanan süreç tanımlaması nesnesi yaratıldı ya da
- Kuyruğa ilişkin **ProcessName** yerel kuyruk özniteliği boştur, ancak kuyruk bir iletim kuyruğudur. Süreç tanımlaması isteğe bağlı olduğu için, **TriggerData** özniteliği başlatılacak kanalın adını da içerebilir. Bu durumda, tetikleme iletili aşağıdaki değerleri içeren öznitelikleri içerir:
  - **QName**: kuyruk adı
  - **ProcessName**: boşluklar
  - **TriggerData**: tetikleme verileri
  - **ApplType**: MQAT\_UNKNOWN
  - **ApplId**: boşluklar
  - **EnvData**: boşluklar
  - **UserData**: boşluklar

7. Bir kullanıma hazırlama kuyruğu yaratıldı ve **InitiationQName** yerel kuyruk özniteisinde belirtildi. Ayrıca:

- Alma istekleri, başlatma kuyruğu için engellenmez (yani, **InhibitGet** kuyruk özniteliğinin değeri MQQA\_GET\_ALLOWALIZD değeridir).
- Put isteklerinin kullanıma hazırlama kuyruğu için engellenmemesi gerekir (yani, **InhibitPut** kuyruk özniteliğinin değeri MQQA\_PUT\_ALLOWALIZD olmalıdır).
- Kullanıma hazırlama kuyruğunun **Usage** özniteliğinin değeri MQUS\_NORMAL olmalıdır.

- Dinamik kuyrukların desteklendiği ortamlarda, başlatma kuyruğu, mantıksal olarak silinmiş olarak işaretlenen dinamik bir kuyruk olmamalıdır.
8. Bir tetikleme izleyicinin şu anda iletileri kaldırmak için açma (initiation) kuyruğu açık (yani, **OpenInputCount** yerel kuyruk özniteliğinin sıfırdan büyük olması).
  9. Uygulama kuyruğuna ilişkin tetikleme denetimi (**TriggerControl** yerel kuyruk özniteliği) MQTC\_ON olarak ayarlandı. Bunu yapmak için, kuyruğunuzu tanımladığınızda **trigger** özniteliğini ayarlayın ya da ALTER QLOCAL komutunu kullanın.
  10. Tetikleyici tipi (**TriggerType** yerel kuyruk özniteliği) MQTT\_NONE değil.

Tüm gerekli koşullar karşılanırsa ve tetikleme koşuluna neden olan ileti bir iş biriminin bir parçası olarak konulursa, iş birimi tamamlanıncaya kadar tetikleyici izleme uygulaması tarafından tetikleme iletileri, iş biriminin kesinleştirilip kesinleştirilmediği ya da MQTT\_FIRST ya da MQTT\_DEPTH tetikleme tipi için kullanılabilir duruma gelinceye kadar, tetikleme iletileri, alma işlemi için kullanılabilir duruma gelmeyebilir.

11. Kuyruğa uygun bir ileti, **TriggerType** MQTT\_FIRST ya da MQTT\_DEPTH ve kuyruk için bir ileti yerleştirilir:

- Daha önce boş değil (MQTT\_FIRST), ya da
- **TriggerDepth** ya da daha fazla ileti (MQTT\_DEPTH) vardı

and conditions “2” sayfa 906 through “10” sayfa 908 (excluding “3” sayfa 906) are satisfied, if in the case of MQTT\_FIRST a sufficient interval (**TriggerInterval** queue manager attribute) has elapsed since the last trigger message was written for this queue.

Bu, kuyruktaki tüm iletileri işlemeden önce biten bir kuyruk sunucusu için izin verilmesine olanak sağlar. Tetikleme aralığının amacı, oluşturulan yinelenen tetikleyici ileti sayısını azaltmaktır.

**Not:** Kuyruk yöneticisini durdurup yeniden başladığınızda, **TriggerInterval** süreölçeri sıfırlanır. İki tetikleyici ileti üretmenin mümkün olduğu küçük bir pencere vardır. Bu pencere, kuyruğun tetikleyici özniteliği bir iletiyle aynı zamanda etkinleştirilecek şekilde ayarlandığı ve kuyruk daha önce boş (MQTT\_FIRST) ya da **TriggerDepth** ya da daha fazla ileti (MQTT\_DEPTH) olan bir kuyruk olmadığı zaman var olur.

12. Bir kuyruğa hizmet veren tek uygulama, bir MQCLOSE çağrısı, **TriggerType** MQTT\_FIRST ya da MQTT\_DEPTH için ve en az:

- Bir (MQTT\_FIRST) ya da
- **TriggerDepth** (MQTT\_DEPTH)

messages on the queue of sufficient priority (condition “2” sayfa 906 ), and conditions “6” sayfa 907 through “10” sayfa 908 are also satisfied.

Bu, bir MQGET çağrısını içeren bir kuyruk sunucusuna izin vermek için, kuyruğu boş bulur ve bu nedenle sona erer; ancak, MQGET ile MQCLOSE çağrıları arasındaki aralıkla bir ya da daha fazla ileti gelir.

**Not:**

- a. Uygulama kuyruğuna hizmet veren program tüm iletileri alamazsa, bu durum kapalı bir döngüye neden olabilir. Program kuyruğun her kapanışında, kuyruk yöneticisi tetikleme izleyicinin sunucu programını yeniden başlatmasına neden olan başka bir tetikleyici ileti yaratır.
  - b. Uygulama kuyruğuna hizmet eden program, alma isteğini geri çekerse (ya da program sona erdirilmeden önce), aynı durumda, bu işlem kuyruğun kapatılıp kapatılmadan önce sona erdirilmesinden sonra da aynı olur. Ancak, program alma isteğini yedeklemeden önce kuyruğu kapar ve kuyruk boş değilse, herhangi bir tetikleyici ileti yaratılmaz.
  - c. Böyle bir döngüye engel olmak için, sürekli olarak yedeklenen iletileri saptamak için MQMD 'nin *BackoutCount* alanını kullanın. Daha fazla bilgi için bkz “Yedeklenen iletiler” sayfa 43.
13. Aşağıdaki koşullar, MQSET ya da bir komut kullanılarak karşılanılır:
    - a. • **TriggerControl** , MQTC\_ON olarak değiştirilir ya da

- **TriggerControl** zaten MQTC\_ON ve **TriggerType, TriggerMsgPriority**ya da **TriggerDepth** (ilgiliyse) değeri değiştirilirse, bu değer

ve en azından:

- Bir (MQTT\_FIRST ya da MQTT\_EVERY) ya da
- **TriggerDepth** (MQTT\_DEPTH)

messages on the queue of sufficient priority (condition “2” sayfa 906 ), and conditions “4” sayfa 906 through “10” sayfa 908 (excluding “8” sayfa 908 ) are also satisfied.

Bir uygulamanın ya da işletmenin tetikleme ölçütlerinin değiştirilmesine izin vermesi, bir tetikleyicinin koşullarının ortaya çıkmasına ilişkin koşulların yerine getirilmesine izin verilmemesine neden olur.

- b. Bir kullanıma hazırlama kuyruğunun **InhibitPut** kuyruk özniteliğinin değeri MQQA\_PUT\_INHIBITED değerinden MQQA\_PUT\_ALLOWALIZVE değerine değişir ve en az:

- Bir (MQTT\_FIRST ya da MQTT\_EVERY) ya da
- **TriggerDepth** (MQTT\_DEPTH)

messages of sufficient priority (condition “2” sayfa 906 ) on any of the queues for which this is the initiation queue, and conditions “4” sayfa 906 through “10” sayfa 908 are also satisfied. (Koşulları karşılayan her kuyruk için bir tetikleyici ileti oluşturulur.)

Bu, başlatma kuyruğunda MQQA\_PUT\_INHIBITED koşulu nedeniyle tetikleme iletilerinin oluşturulmamasına izin vermek içindir, ancak bu koşul şimdi değiştirildi.

- c. Bir uygulama kuyruğuna ilişkin **InhibitGet** kuyruk özniteliğinin değeri MQQA\_GET\_INHIBITED değerinden MQQA\_GET\_ALLOWALIZVE değerine değişir ve en az:

- Bir (MQTT\_FIRST ya da MQTT\_EVERY) ya da
- **TriggerDepth** (MQTT\_DEPTH)

messages of sufficient priority (condition “2” sayfa 906 ) on the queue, and conditions “4” sayfa 906 through “10” sayfa 908, excluding “5” sayfa 907, are also satisfied.

Bu, uygulamaların yalnızca uygulama kuyruğundan ileti alabildiği zaman tetiklenebilmesini sağlar.

- d. Tetikleme izleme programı, bir başlatma kuyruğundan giriş için bir MQOPEN çağırısı yayınlar ve en az:

- Bir (MQTT\_FIRST ya da MQTT\_EVERY) ya da
- **TriggerDepth** (MQTT\_DEPTH)

messages of sufficient priority (condition “2” sayfa 906 ) on any of the application queues for which this is the initiation queue, and conditions “4” sayfa 906 through “10” sayfa 908 (excluding “8” sayfa 908 ) are also satisfied, and no other application has the initiation queue open for input (one trigger message is generated for each such queue satisfying the conditions).

Bu, tetikleme izleme programı çalışmazken kuyruklara ulaşan iletilerin ve kuyruk yöneticisinin yeniden başlatılması ve iletilerin (kalıcı olmayan) kaybolması için izin verilmesine olanak tanır.

14. MSGDLVSQ doğru biçimde ayarlanmış. MSGDLVSQ=FIFO ' u ayarlıyorsanız, iletiler ilk olarak ilk giren ilk sırada kuyruğa teslim edilir. İletinin önceliği yok sayılır ve kuyruğun varsayılan önceliği iletiye atanmaktadır. **TriggerMsgPriority** , kuyruğun varsayılan önceliğine göre daha yüksek bir değere ayarlandıysa, hiçbir ileti tetiklenmez. **TriggerMsgPriority** , kuyruğun varsayılan önceliğine eşit ya da bu önceliğe eşit olarak ayarlanırsa, FIRST, EVERY ve DEPTH tipi için tetikleme gerçekleşir. Bu tiplerle ilgili bilgi için, “Tetikleme olaylarını denetleme” sayfa 910 altındaki **TriggerType** alanının açıklamasına bakın.

MSGDLVSQ=PRIORITY değeri ve ileti önceliği *TriggerMsgPriority* alanına eşitse ya da daha büyükse, iletiler yalnızca bir tetikleme olayına doğru sayılır. Bu durumda, FIRST, EVERY ve DEPTH tipi için tetikleyici ortaya çıkar. As an example, if you put 100 messages of lower priority than the **TriggerMsgPriority**, the effective queue depth for triggering purposes is still zero. Daha sonra kuyruğa başka bir ileti koyarsanız, ancak bu kez öncelik **TriggerMsgPriority**' den büyük ya da bu

değere eşit olduğunda, etkin kuyruk derinliği sıfırdan bire yükselir ve **TriggerType** FIRST için koşul karşılanır.

#### Notlar:

1. From step “12” sayfa 908 (where trigger messages are generated as a result of some event other than a message arriving on the application queue), the trigger message is not put as part of a unit of work. Ayrıca, **TriggerType** MQTT\_EVERY ise ve uygulama kuyruğunda bir ya da daha çok ileti varsa, yalnızca bir tetikleyici ileti oluşturulur.
2. If IBM MQ segments a message during MQPUT, a trigger event will not be processed until all the segments have been successfully placed on the queue. Ancak, ileti bölümleri kuyruksa, IBM MQ bunları tetikleme amacıyla tek tek ileti olarak değerlendirir. Örneğin, üç parçaya bölünen tek bir mantıksal ileti, ilk MQPUT ve kesimlendi olduğunda yalnızca bir tetikleyici olayının işlenmesine neden olur. Ancak, üç kesimin her biri kendi tetikleme olaylarının, IBM MQ ağı üzerinden taşındığı şekilde işlenmesine neden olur.
3. IBM MQ for z/OS için, paylaşılan bir kuyruk tetikleme için ayarlandıysa ve paylaşılan kuyruğu barındıran Coupling Facility ile bağlantı kaybolursa, bir tetikleyici olayı oluşturulabilir ve başlatma kuyruğuna bir ileti konabilir. Bu durum, özgün paylaşılan kuyruk ayarına tetikleme için bir ileti konmadığında da oluşabilir. Bu, List Notification Vector içinde belgelendiği gibi, IXLVECTR makrosu tarafından bitlerin aşırı gösterimine neden olur.

### Tetikleme olaylarını denetleme

Tetikleme olaylarını, uygulama kuyruğunuzu tanımlayan bazı öznitelikleri kullanarak denetlemenizi sağlar. Bu bilgiler, tetikleme tiplerinin kullanılmasına ilişkin örnekler de verir: EVERY, FIRST ve DEPTH.

Tetiklemeyi etkinleştirebilir ve devre dışı bırakabilirsiniz; bir tetikleme olayına doğru sayılan iletilerin sayısını ya da önceliğini seçebilirsiniz. Nesnelerin öznitelikler alanında bu özniteliklerin tam açıklaması vardır.

İlgili öznitelikler şunlardır:

#### **TriggerControl**

Bir uygulama kuyruğu için tetiklemeyi etkinleştirmek ve devre dışı bırakmak için bu özniteliği kullanın.

#### **TriggerMsgPriority**

Bir tetikleme olayına doğru sayılması için bir iletinin sahip olması gereken öncelik alt sınırı. Uygulama kuyruğuna *TriggerMsgPriority* değerinden küçük bir öncelik ileti gönderilirse, kuyruk yöneticisi bir tetikleme ileti yaratılıp yaratılmayacağını belirlediğinde iletiyi yoksayar. *TriggerMsgPriority* sıfır olarak ayarlandıysa, tüm iletiler bir tetikleme olayına doğru sayılır.

#### **TriggerType**

NONE (Yok) tetikleme tipine ek olarak (yalnızca *TriggerControl* 'un OFF' ye ayarlanması gibi tetiklemeyi geçersiz kılar), bir kuyruğun olayları tetiklemek üzere duyarlılığını ayarlamak için aşağıdaki tetikleyici tiplerini kullanabilirsiniz:

##### **Her**

Tetikleme olayı, uygulama kuyruğuna her ileti geldiğinde ortaya çıkar. Bir uygulamanın birden çok örneğinin başlatılmış olmasını istiyorsanız, bu tetikleyici tipini kullanın.

##### **Birinci**

Tetikleme olayı, yalnızca uygulama kuyruğunda bulunan ileti sayısı sıfırdan bire değişirse gerçekleşir. Bir hizmet programının bir kuyruğa ilk ileti geldiğinde başlatılmasını istiyorsanız bu tetikleyici tipini kullanın, işlenecek başka ileti kalmayınca kadar devam edin, daha sonra sona erdirin. Kuyruğu boş oluncaya kadar her zaman işlemeniz gerekir. Ayrıca bkz. “ÖNCE tetikleme tipi özel durumu” sayfa 911.

##### **Derinlik**

Tetikleme olayı, yalnızca uygulama kuyruğunda ileti sayısı **TriggerDepth** özniteliğinin değerine ulaştığında oluşur. Bu tetikleme tipinin tipik bir kullanımı, bir istek kümesine verilen tüm yanıtlar alındığında bir program başlatmaya neden olur.

**Derinliğe göre tetikleme:** Kuyruk yöneticisi, derinliği tetikleyerek tetikleme ileti yarattıktan sonra tetiklemeyi ( *TriggerControl* özniteliğini kullanarak) devre dışı bırakır. Uygulamanızın,

bu gerçekleştikten sonra kendisini tetiklemeyi yeniden etkinleştirmesi gerekir (MQSET çağrısını kullanarak).

Tetiklemeyi devre dışı bırakma eylemi syncpoint denetimi altında değildir; bu nedenle, bir iş birimi yedeklenerek tetikleme yeniden etkinleştirilemiyor. Bir program tetikleme olayına neden olan bir put isteğini geri çekerse ya da program sona erdirilirse, MQSET çağrısını ya da ALTER QLOCAL komutunu kullanarak tetiklemeyi yeniden etkinleştirmeniz gerekir.

### **TriggerDepth**

Bir tetikleme olayının derinlemesine tetiklenmesine neden olan bir kuyruğun ileti sayısı.

Bir kuyruk yöneticisinin tetikleme ileti oluşturması için karşılanması gereken koşullar, "[Tetikleme olayına ilişkin koşullar](#)" sayfa 906'inde açıklanmıştır.

## **EVERY tetikleme tipinin kullanımı örneği**

Motor sigortası için istek üreten bir uygulama düşünün. Uygulama, her seferinde aynı yanıtı belirlemek üzere bir dizi sigorta şirketine istek iletileri gönderebilir. Bu yanıt kuyruğunda EVERY tipinde bir tetikleyici tanımlayabilir ve böylece her yanıt geldiğinde yanıt, yanıtı işlemek için sunucunun bir eşgörünümünü tetikleyebilir.

## **FIRST tetikleme tipini kullanma örneği**

Her bir gün işyerine ilişkin ayrıntıları baş ofise iletmek için şube ofisleri olan bir kuruluş düşünün. Bunların hepsi aynı zamanda, çalışma gününün sonunda, ve baş ofisinde tüm şube ofislerinden gelen detayları işleyen bir uygulama var. Baş ofise gelen ilk ileti, bu uygulamayı başlatan bir tetikleme olayına neden olabilir. Bu uygulama, kuyruğunda daha fazla ileti kalmayınca kadar işlemeye devam eder.

## **Tetikleme tipi DEPTH ' in kullanımı örneği**

uçuş rezervasyonunu doğrulamak için tek bir istek oluşturan bir seyahat acente uygulaması düşünün. bir otel odası için rezervasyon onaylamak, bir araba kiralamak ve bazı gezginler çekleri sipariş etmek. Uygulama bu öğeleri dört istek iletilisine ayırabilir ve her birini ayrı bir hedefe gönderilebilir. Yanıtlama kuyruğunda (değeri 4 değerine ayarlanmış bir derinlik), yalnızca dört yanıt geldiğinde yeniden başlatılacak şekilde, yanıtlama kuyruğunda (değere 4 değerine ayarlanmış) bir tetikleyici ayarlanabilir.

Dört yanıtın önce yanıt kuyruğunda başka bir ileti (büyük olasılıkla farklı bir istekse) varırsa, istekte bulunan uygulama erken tetiklenir. Bunu önlemek için, bir isteğe birden çok yanıt toplamak için DERINLIK tetiklemesi kullanılırken, her istek için her zaman yeni bir yanıt kuyruğu kullanın.

## **ÖNCE tetikleme tipi özel durumu**

FIRST tetikleyicisiyle, başka bir ileti geldiğinde uygulama kuyruğunda önceden bir ileti varsa, kuyruk yöneticisi tipik olarak başka bir tetikleyici ileti yaratmaz.

Ancak, kuyruğa hizmet veren uygulama gerçekten kuyruğu açmayabilir (örneğin, uygulama sona erebilir, büyük olasılıkla bir sistem sorunu nedeniyle). Süreç tanımlaması nesnesine yanlış bir uygulama adı konulduysa, kuyruğa hizmet veren uygulama hiçbir iletiyi almayacaktır. Bu durumlarda, uygulama kuyruğuna başka bir ileti gelirse, bu iletiyi (ve kuyrukta bulunan diğer iletileri) işlemek için çalışan bir sunucu yoktur.

Bununla başa çıkmak için, kuyruk yöneticisi aşağıdaki durumlarda daha fazla tetikleme ileti yaratır:

- Uygulama kuyruğuna başka bir ileti gelirse, ancak kuyruk yöneticisi o kuyruk için son tetikleme iletilerini yarattığından bu yana önceden tanımlanmış bir zaman aralığı geçtiyse. Bu zaman aralığı, *TriggerInterval* kuyruk yöneticisi öznelisinde tanımlı. Varsayılan değeri 999 999 milisaniyedir.
- IBM MQ for z/OS' ta, açık bir başlatma kuyruğu adı alan uygulama kuyrukları düzenli olarak taranır. Son tetikleyici iletilerinden bu yana *TRIGINT* milisaniyeler iletiliyse ve kuyruk bir tetikleme olayı için koşulları karşıladığında ve *CURDEPTH* sıfırdan büyükse, bir tetikleyici ileti oluşturulur. Bu süreç, geri durdurma tetiklemesi olarak adlandırılır.

Uygulamanızda kullanılacak tetikleme aralığı için bir değer belirlenirken aşağıdaki noktaları göz önünde bulundurun:

- *TriggerInterval* değerini düşük bir değere ayarladıysanız ve uygulama kuyruğuna hizmet veren bir uygulama yoksa, FIRST tetikleme tipi EVERY tetikleme tipi gibi davranabilir. Bu, iletilerin uygulama kuyruğuna konulmakta olduğu hıza bağlıdır; bu, diğer sistem etkinliklerine bağlı olabilir. Bunun nedeni, tetikleme aralığı çok küçükse, tetikleme tipi FIRST, EVERY değil, her ileti bir uygulama kuyruğuna her ileti konursa başka bir tetikleyici iletilisi oluşturulur. (Tetikleme tipi sıfır olan FIRST ile tetikleme tipi, EVERY tetikleyicisinin eşdeğeridir.)
- On IBM MQ for z/OS if you set *TRIGINT* to a low value, and there is no application serving the trigger type FIRST application queue, backstop triggering will generate a trigger message each time the periodic scan of application queues that name open initiation queues takes place.
- Bir iş birimi yedeklendiyse (bkz. [Tetikleme iletileri ve iş birimleri](#)) ve tetikleme aralığı bir yüksek değere (ya da varsayılan değere) ayarlanmıştır, iş birimi yedeklendiğinde bir tetikleyici iletilisi oluşturulur. Ancak, tetikleme aralığını düşük bir değere ya da sıfır değerine ayarladıysanız (tetikleme tipi FIRST TO DEAD LIKE TRIGGER TYPE EVERY gibi) birçok tetikleyici iletilisi oluşturulabilir. İş birimi yedeklendiyse, tüm tetikleyici iletiler hala kullanılabilir kılınmaya devam eder. Oluşturulan tetikleyici ileti sayısı, tetikleme aralığına bağlıdır. Tetikleme aralığı sıfır olarak ayarlandıysa, ileti sayısı üst sınırı oluşturulur.

### **Tetiklenen kuyrukları kullanan bir uygulama tasarlanması**

Uygulamalarınız için nasıl ayarlanacak, denetleyeceğini ve tetikleme gördünüz. Burada, uygulamanızı tasarlarken dikkate almanız gereken bazı ipuçları bulunur.

### **İletilerin ve çalışma birimlerinin tetiklenmesi**

Bir iş biriminin parçası olmayan tetikleme olayları nedeniyle yaratılan tetikleme iletileri, herhangi bir iş biriminin dışında, başka hiçbir iletiye bağımlı olmadan, başlatma kuyruğuna yerleştirilir ve tetikleme izleme programı tarafından hemen geri alınabilmekte kullanılır.

Bir iş biriminin parçası olan tetikleme olayları nedeniyle yaratılan tetikleme iletileri, iş biriminin kesinleştirildiği ya da yedeklenip yedeklenmediği, UOW çözüldüğünde kullanıma hazırlama kuyruğunda kullanılabilir kılınmaktadır.

Kuyruk yöneticisi bir tetikleme iletilisini bir başlatma kuyruğuna koyamazsa, bu ileti, ölü-mektup (teslim edilemeyen ileti) kuyruğuna konacak.

#### **Not:**

1. Kuyruk yöneticisi, bir tetikleme olayına ilişkin koşulların var olup olmadığını değerlendirirken hem kesinleştirilmiş hem de kesinleştirilmemiş iletiler sayılıyor.

Birinci ya da DERINLIK tipinde tetikleme işlemi ile, istenen koşullar karşılandığında bir tetikleme iletilisi her zaman kullanılabilir olacak şekilde, iş birimi geriletmiş olsa bile, tetikleme iletileri kullanılabilir duruma getirilmektedir. Örneğin, FIRST tetikleyicisi ile tetiklenen bir kuyruğa ilişkin iş birimi içinde bir put isteği düşünün. Bu, kuyruk yöneticisinin tetikleyici iletilisi yaratmasına neden olur. Başka bir iş biriminden başka bir put isteği ortaya çıkarsa, bu durum başka bir tetikleme olayına neden olmaz; çünkü, uygulama kuyruğunda ileti sayısı artık bir tetikleme olayına ilişkin koşulları yerine getirmeyen birinden ikiden ikiye çevirmiştir. Şimdi ilk iş birimi geriletirse, ancak ikincisi kesinleştirilirse, bir tetikleyici iletilisi hala yaratılır.

Ancak bu, tetikleme olayına ilişkin koşullar karşılanmadığında tetikleme iletilerinin bazen yaratıldığını gösterir. tetikleme kullanan uygulamalar her zaman bu durumu ele almak için hazırlanmalıdır. It is recommended that you use the wait option with the MQGET call, setting the *WaitInterval* to a suitable value.

Yaratılan tetikleme iletileri her zaman kullanılabilir kılın, iş biriminin geriletmediği ya da kesinleştirilip kesinleştirilmemiş olması.

2. Yerel paylaşılan kuyruklar için (yani, bir kuyruk paylaşım grubundaki paylaşılan kuyruklar) kuyruk yöneticisi yalnızca kesinleştirilmiş iletileri sayar.



## Tetiklenen kuyruktan ileti alınması

Tetiklemeyi kullanan uygulamaları tasarladığınızda, bir program başlatma tetikleyicisi ile uygulama kuyruğunda kullanılabilir olan diğer iletiler arasında bir gecikme olabileceğini dikkate alınız. Tetikleme olayına neden olan ileti, diğerlerinden önce kesinleştirildiğinde bu durum oluşabilir.

İletilerin gelmesine izin vermek için, MQGET çağrısını kullanırken tetikleme koşullarının belirlendiği bir kuyruktan iletileri kaldırmak için her zaman bekleme seçeneğini kullanın. *WaitInterval* , iletilmekte olan bir ileti arasında en uzun makul süre için yeterli olmalıdır ve bu süre için çağrı kesinleştirilmelidir. İleti uzak bir kuyruk yöneticisinden geldiyse, bu süre aşağıdaki durumdan etkilenir:

- Kesinleştirilmeden önce konacağı iletilerin sayısı
- İletişim bağlantısının hızı ve kullanılabilirliği
- İletilerin büyüklükleri

MQGET çağrısını bekleme seçeneği ile kullanmanız gereken bir durum örneği için, iş birimlerini tanımlarken kullandığımız aynı örneği göz önünde bulundurun. Bu, FIRST tetikleyicisi ile tetiklenen bir kuyruk için çalışma birimi içindeki bir put isteğinde bulunmuyordu. Bu olay kuyruk yöneticisinin bir tetikleyici iletilerini yaratmasına neden olur. Başka bir iş biriminden başka bir put isteği ortaya çıkarsa, bu, uygulama kuyruğunda ileti sayısı sıfırdan bire değişmediği için başka bir tetikleme olayına neden olmaz. Şimdi ilk iş birimi geriletilirse, ancak ikincisi kesinleştirilirse, bir tetikleyici iletilerini hala yaratılır. Bu nedenle, tetikleme iletilerini ilk iş biriminin yedekleneceği sırada yaratılır. İkinci iletilerini kesinleştirilmesinden önce önemli bir gecikme süresi varsa, tetiklenen uygulamanın bunu beklemesi gerekebilir.

DERINLIK tipi tetiklenmesiyle, ilgili tüm iletiler sonunda kesinleştirilse de bir gecikme oluşabilir. **TriggerDepth** kuyruk özniteliğinin 2 değerine sahip olduğunu varsayın. Kuyruğa iki ileti geldiğinde, ikinci ileti bir tetikleme iletilerini yaratılmasına neden olur. Ancak, ikinci ileti kesinleştirilmek üzere ilk iletilerini, tetikleme iletilerini kullanılabilir duruma gelmesi o anda geçerli olur. Tetikleme izleme programı sunucu programını başlatır, ancak program ilk ileti kesinleştirilinceye kadar yalnızca ikinci iletilerini alabilir. Bu nedenle programın, ilk iletilerini kullanıma sunulmasını beklemesi gerekebilir.

Uygulamanızı tasarlayın, böylece bekleme süreniz dolduğunda hiçbir ileti alınamazsa sona erdirilir. Bir ya da daha çok ileti daha sonra ulaşırsa, bunları işlemek için başvurunuzun yeniden denenmesine dikkat edin. Bu yöntem, uygulamaların boşa durmasını ve kaynakların gereksiz yere kullanılmasını önler.

### ***Tetikleme izleyicileri tarafından kullanıma hazırlama kuyruğu işleme***

Bir kuyruk yöneticisine, bir tetikleme izleme programı, kuyruğa hizmet veren diğer uygulamalar gibidir. Ancak, bir tetikleme izleme programı başlatma kuyruklarına hizmet eder.

Tetikleme izleme programı genellikle sürekli olarak çalışan bir programdır. Bir tetikleme iletilerini bir başlatma kuyruğuna ulaştığında, tetikleme izleme programı o iletilerini alır. Bu ileti, uygulama kuyruğunda iletilerini işlemek üzere olan uygulamayı başlatmak için bir komut vermek üzere iletilerini ilgili bilgileri kullanır.

Tetikleme izleme programının, doğru uygulama kuyruğunda doğru işlemleri gerçekleştirebilmesi için, programın başlatıldığı programa yeterli sayıda bilgi geçirmesi gerekir.

Kanal başlatıcı, ileti kanalı araçları için özel bir tetikleyici izleyicisi örneğidir. Ancak bu durumda, FIRST YA DA DEPTH tetikleyicisinden birini kullanmanız gerekir.

### **Windows UNIX UNIX ve Windows sistemlerinde tetikleme izleme programları**

Bu konu, UNIX ve Windows sistemlerinde sağlanan tetikleme izleyicileri hakkındaki bilgileri içerir.

Sunucu ortamı için aşağıdaki tetikleme izleme programları sağlanmıştır:

#### **amqstrg0**

Bu, **runmqtrm** tarafından sağlanan işlevin bir alt kümesini sağlayan örnek bir tetikleme izleyicidir. **amqstrg0** ile ilgili ek bilgi için [“Örnek Programların çoklu Platformlar Üzerinde Kullanılması” sayfa 1093](#) ' e bakın.

## runmqtrm

Bu komutun sözdizimi şöyledir: **runmqtrm** [ -m *QMgrName* ] [ -q *InitQ* ]; burada *QMgrName* , kuyruk yöneticisi ve *InitQ* başlatma kuyruğudur. Varsayılan kuyruk SYSTEM.DEFAULT.INITIATION.QUEUE varsayılan kuyruk yöneticisinde KUYRUK. Uygun tetikleyici iletilerine ilişkin programları çağırır. Bu tetikleyici izleyicisi, varsayılan uygulama tipini destekler.

Tetikleme izleyicinin işletim sistemine geçirilen komut dizilimi aşağıdaki gibi oluşturulur:

1. İlgili PROCESS (süreç) tanımlamasından *AppLId* (yaratıldıysa)
2. Çift tırnak işareti içine alınmış MQTMC2 yapısı
3. İlgili PROCESS (süreç) tanımlamasından *EnvData* (yaratıldıysa)

Burada *AppLId* , komut satırına girilirken çalıştırılacak programın adıdır.

İletilen parametre, MQTMC2 karakter yapısıdır. Sistem komutunun tek bir parametre olarak kabul etmesi için, tam olarak sağlandığı gibi, bu dizeye sahip olan bir komut dizgisi çağrılır.

Tetikleme izleme programı, başlatma kuyruğunda yeni başlatılmış olan uygulamanın tamamlanmasına kadar başka bir ileti olup olmadığını denetleyemez. Uygulamanın yapması gereken çok işlem varsa, tetikleme izleme programı gelen tetikleyici ileti sayısına yetişemeyebilir. İki seçeneğiniz vardır:

- Çalışan daha fazla tetikleyici izleme programı var
- Başlatılan uygulamaları arka planda çalıştır

Çalışmakta olan daha fazla tetikleyiciniz varsa, herhangi bir zamanda çalışabilecek uygulama sayısı üst sınırını denetleyebilirsiniz. If you run applications in the background, there is no restriction imposed by IBM MQ on the number of applications that can run.

To run the started application in the background on Windows systems, within the *AppLId* field, prefix the name of your application with a START command. Örneğin:

```
START ?B AMQSECHA
```

**UNIX** Başlatılan uygulamayı, UNIX üzerinde arka planda çalıştırmak için, PROCESS tanımlamasının *EnvData* sonuna bir & at koyun.

**Not:** **Windows** Bir Windows yolunun, yol adının bir parçası olarak boşluklar varsa, bunlar tırnak işareti içine alınmalıdır ("). tek bir bağımsız değişken olarak ele alındığından emin olmak için. Örneğin, "C:\Program Files\Application Directory\Application.exe".

Aşağıda, dosya adının yolun bir parçası olarak boşluk bulunduğu bir APPLICID dizgisi örneği yer almaktadır:

```
START "" /B "C:\Program Files\Application Directory\Application.exe"
```

Örneğin, Windows START komutunun sözdizimi çift tırnak içine alınmış boş bir dizgi içerir. START, tırnak işaretlerindeki ilk bağımsız değişkenin yeni komutun başlığı olarak işleneceğini belirtir. Windows 'un' title ' bağımsız değişkenine ilişkin uygulama yolunu yanlış yapmadığından emin olmak için, uygulama adından önce komutta çift tırnak işareti içine alınmış bir başlık dizgisi ekleyin.

IBM MQ istemcisi için aşağıdaki tetikleme izleme programları sağlanır:

## runmqtrmc

This is the same as runmqtrm except that it links with the IBM MQ MQI client libraries.

**Windows** **UNIX** *Trigger monitor for CICS*

amqltmc0 tetikleyicisi izleyicisi CICS için sağlanmıştır. It works in the same way as the standard trigger monitor, runmqtrm, but you run it in a different way and it triggers CICS transactions.

Bu konu yalnızca Windows, UNIX ve Linux x86-64 sistemleri için geçerlidir.

Tetikleme izleme programı bir CICS programı olarak sağlanır; bunu 4 karakterlik bir hareket adıyla tanımlayın. Tetikleme izleyicisini başlatmak için 4 karakterlik bir ad girin. Varsayılan kuyruk yöneticisini (qm.ini dosyasında ya da IBM MQ for Windows'de, kayıt defterinde) ve SYSTEM.CICS.INITIATION.QUEUE.

If you want to use a different queue manager or queue, build the trigger monitor MQTMC2 structure: this requires you to write a program using the EXEC CICS START call, because the structure is too long to add as a parameter. Then, pass the MQTMC2 structure as data to the START request for the trigger monitor.

MQTMC2 yapısını kullandığınızda, diğer alanlara gönderme yapmadığı için tetikleyici izleyiciye yalnızca *StrucId*, *Version*, *QName* ve **QMgrName** parametrelerini sağlamanız gerekir.

İletiler başlatma kuyruğundan okunur ve tetikleyici iletisinde APPL\_TYPE ' ın MQAT\_CICS olduğu varsayılarak, EXEC CICS START kullanılarak CICS işlemlerini başlatmak için kullanılır. İleti başlatma kuyruğundan gelen iletilerin okunması, CICS syncpoint denetimi altında gerçekleştirilir.

İletiler, izleme programı başladığında ve durduğunda ve bir hata ortaya çıktığında oluşturulur. Bu iletiler CSMT geçici veri kuyruğuna yollar.

*Çizelge 133. Tetikleme izleyicisinin kullanılabilir sürümleri.*

İki sütuna sahip bir tablo. İlk kolonum, tetikleme izleme programının kullanılabilir sürümlerini ve ikinci kolonda her sürümün hangi platformlarda kullanıldığını gösterir.

S\u00fcr\u00fcm	Kullan
amqltmc0	TXSeries için: <ul style="list-style-type: none"><li><b>AIX</b> AIX</li><li><b>Linux</b> Linux x86-64 sistemleri</li><li><b>Solaris</b> Oracle Solaris Sürüm 5.1</li></ul>
amqltmc4	<b>Windows</b> TXSeries for Windows 5.1
amqltmcc	CICS tetikleme izleyicisinin istemci tarafından bağlanmış sürümü

Başka ortamlar için bir tetikleme izleyicisine gereksinim duyarsanız, kuyruk yöneticisinin başlatma kuyruklarına koyduğu tetikleme iletilerini işleyebilecek bir program yazın. Böyle bir program aşağıdaki eylemleri gerçekleştirmelidir:

1. Bir iletinin başlatma kuyruğuna varmasını beklemek için MQGET çağrısını kullanın.
2. Başlatılacak uygulamanın adını bulmak için, tetikleme iletisinin MQTM yapısındaki alanları ve çalıştığı ortamı inceleyin.
3. Ortama özgü bir başlatma komutu verin.

**z/OS** Örneğin, z/OS toplu olarak, iç okuyucuya bir iş gönderin.

4. MQTM yapısını gerekirse, MQTMC2 yapısına dönüştürün.
5. Başlatılan uygulamaya MQTMC2 ya da MQTM yapısını geçirin. Bu, kullanıcı verilerini içerebilir.
6. Uygulama kuyruğunuzla ilişkilendirin, o kuyruğa hizmet verecek olan uygulamayı ilişkilendirin. Bunu, kuyruğun **ProcessName** öznelisinde (yaratıldıysa) süreç tanımlaması nesnesini (yaratıldıysa) adlandırmayla yapabilirsiniz. Süreç tanımlaması nesnesini adlamak için, **DEFINE QLOCAL** ya da **ALTER QLOCAL** komutunu kullanabilirsiniz.

**IBM i** IBM üzerinde, CRTMQMQ ya da CHGMQMQ olanağını da kullanabilirsiniz.

Tetikleyici izleme arabirimine ilişkin ek bilgi için [MQTMC2](#) başlıklı konuya bakın.

**IBM i** IBM üzerindeki tetikleyicileri tetikle

On IBM i, instead of the **runmqtrm** control command, use the IBM MQ for IBM i CL command **STRMQTRM**.

STRMQMTRM komutunu aşağıdaki gibi kullanın:

```
STRMQMTRM INITQNAME(InitQ) MQMNAME(QMgrName)
```

Ayrıntılar runmqtrm için geçerli.

Kendi tetikleyici izleyicilerinizi yazmak için model olarak kullanabileceğiniz aşağıdaki örnek programlar da sağlanmıştır:

#### **AMQSTRG4**

Bu, başlatılacak sürece ilişkin bir IBM i işini gönderen bir tetikleyici izleyicisi, ancak bu, her bir tetikleyici iletilisiyle ilişkili ek işlemlerin olduğu anlamına gelir.

#### **AMQSERV4**

Bu bir tetikleyici sunucudur. Her bir tetikleyici iletilisi için, bu sunucu işlemi kendi işiyle ilgili komutu çalıştırır ve CICS işlemlerini çağırabilir.

Hem tetikleme izleme programı hem de tetikleme sunucusu, başlatılacak programlara bir MQTMC2 yapısı iletir. Bu yapıyla ilgili açıklamalar için bkz. [MQTMC2](#). Bu örneklerin her ikisi de hem kaynak hem de yürütülebilir formlarda teslim edilir.

Because these trigger monitors can invoke only native IBM i programs, they cannot trigger Java programs directly, because Java classes are located in the IFS. Ancak, Java programları, daha sonra Java programını çağıran ve TMC2 yapısından geçen bir CL programını tetikleyerek dolaylı olarak tetiklenebilir. TMC2 yapısının büyüklük alt sınırı 732 bayttır.

Aşağıda, örnek bir CLP ' nin kaynağı yer alıyor:

```
PGM PARM(&TMC2)
DCL &TMC2 *CHAR LEN(800)
ADDENVVAR ENVVAR(TM) VALUE(&TMC2)
QSH CMD('java_pgname $TM')
RMVENVVAR ENVVAR(TM)
ENDPGM
```

IBM MQ MQI client: RUNMQTMC için aşağıdaki tetikleme izleme programı sağlanmıştır.

RUNMQTMC ' yi aşağıdaki gibi çağırın:

```
CALL PGM(QMQM/RUNMQTMC) PARM('-m' QMgrName '-q' InitQ)
```

### ***Tetikleme İletilerinin Özellikleri***

Aşağıdaki konularda, ileti tetikleme iletilerinin diğer bazı özellikleri açıklanmaktadır.

- [“Tetikleme iletilerinin sürekliliği ve önceliği” sayfa 916](#)
- [“Kuyruk yöneticisi yeniden başlatma ve ileti tetikleme” sayfa 917](#)
- [“İletilerin tetiklenmesi ve nesne özniteliklerinde yapılan değişiklikler” sayfa 917](#)
- [“Tetikleyici İletilerinin Biçimi” sayfa 917](#)

### **Tetikleme iletilerinin sürekliliği ve önceliği**

Tetikleme iletileri kalıcı değildir; bunun için gerekli bir gereksinim yoktur.

Ancak, tetikleme olaylarını oluşturma koşulları kalıcı olur; bu nedenle, bu koşullar karşılandığında tetikleme iletileri oluşturulur. Bir tetikleme iletilisi kaybolursa, uygulama kuyruğunda uygulama iletilisinin devam etmesi, tüm koşullar karşılandığı anda kuyruk yöneticisinin bir tetikleme iletilisi oluşturmasını sağlar.

Bir iş birimi geriye işlenirse, üretilen tüm tetikleme iletileri her zaman teslim edilir.

Tetikleme iletileri, başlatma kuyruğunun varsayılan önceliğini alır.

## Kuyruk yöneticisi yeniden başlatma ve ileti tetikleme

Bir kuyruk yöneticisinin yeniden başlatıldığı sırada, giriş için bir başlatma kuyruğu açıldığında, ilişkili bir uygulama kuyruğu üzerinde ileti varsa ve tetikleme için tanımlandıysa, bir tetikleme iletisi bu başlatma kuyruğuna konabilir.

## İletilerin tetiklenmesi ve nesne özniteliklerinde yapılan değişiklikler

Tetikleme iletileri, tetikleme olayı sırasında zorlamalı olarak tetikleme özniteliklerinin değerlerine göre yaratılır.

Tetikleme iletisi tetikleme izleme programı tarafından daha sonra kullanılabilir kılınmadıysa (yaratılmasına neden olan ileti bir iş birimi içine konduysa), bu sırada tetikleme özniteliklerinde yapılan değişiklikler tetikleme iletisinde hiçbir etkiye sahip olmaz. Özellikle, tetiklemenin geçersiz kılınması, bir tetikleme iletisinin yaratıldıktan sonra kullanılabilir kılınmasını engellemektedir. Ayrıca, uygulama kuyruğu, tetikleme iletisinin kullanılabilir kılındığı zaman artık var olmayabilir.

## Tetikleyici İletilerinin Biçimi

Bir tetikleme iletisinin biçimi, MQTM yapısı tarafından tanımlanır.

Bu, kuyruk yöneticisinin tetikleme iletisini yarattığında, uygulama kuyruğunun nesne tanımlamalarındaki ve o kuyrukla ilişkili sürecin nesne tanımlamalarındaki bilgileri kullanarak doldurduğu aşağıdaki alanları içerir:

### **StrucId**

Yapı tanıtıcısı.

### **Version**

Yapının sürümü.

### **QName**

Tetikleme olayının ortaya çıktığı uygulama kuyruğunun adı. Kuyruk yöneticisi bir tetikleme iletisi yarattığında, bu alanı uygulama kuyruğunun **QName** özniteliğini kullanarak doldurur.

### **ProcessName**

Uygulama kuyruğuyla ilişkili süreç tanımlaması nesnesinin adı. Kuyruk yöneticisi bir tetikleme iletisi yarattığında, bu alanı uygulama kuyruğunun **ProcessName** özniteliğini kullanarak doldurur.

### **TriggerData**

Tetikleyici izleme programı tarafından kullanılmak üzere serbest biçimli bir alan. Kuyruk yöneticisi bir tetikleme iletisi yarattığında, bu alanı uygulama kuyruğunun **TriggerData** özniteliğini kullanarak doldurur. IBM MQ for z/OS'de herhangi bir IBM MQ ürününde, bu alan, tetiklenecek kanalın adını belirtmek için kullanılabilir.

### **ApplType**

Tetikleme izleyicisinin başlatılacağı uygulamanın tipi. Kuyruk yöneticisi bir tetikleme iletisi yarattığında, *ProcessName* içinde tanımlanan süreç tanımlaması nesnesinin **ApplType** özniteliğini kullanarak bu alanı doldurur.

### **ApplId**

Tetikleme izleyicinin başlatılacağı uygulamayı tanıtan bir karakter dizilimi. Kuyruk yöneticisi bir tetikleme 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/OS'tarafından sağlanan CSQQTRMN ' i kullandığınızda, süreç tanımlaması nesnesinin **ApplId** özniteliği bir IMS işlem tanıtıcısıdır.

### **EnvData**

Tetikleme izleme programı tarafından kullanılmak üzere ortama ilişkin verileri içeren bir karakter alanı. Kuyruk yöneticisi bir tetikleme iletisi yarattığında, *ProcessName* içinde tanımlanan süreç tanımlaması nesnesinin **EnvData** özniteliğini kullanarak bu alanı doldurur. CICStarafından sağlanan tetikleyici

izleyicisi (CKTI) ya da IBM MQ for z/OS tarafından sağlanan tetikleme izleme programı (CSQQTRMN) bu alanı kullanmaz, ancak diğer tetikleme izleme programları bunu kullanmayı seçebilir.

### **UserData**


Tetikleyici izleme programı tarafından kullanılmak üzere kullanıcı verilerini içeren bir karakter alanı. Kuyruk yöneticisi bir tetikleme iletisi yarattığında, *ProcessName* içinde tanımlanan süreç tanımlaması nesnesinin **UserData** öznelikliğini kullanarak bu alanı doldurur. Bu alan, tetiklenecek kanalın adını belirtmek için kullanılabilir.

There is a full description of the trigger message structure in [MQTM](#).

### **Tetikleme işe yaramadığında**

Tetikleme izleme programı programı başlatamazsa ya da kuyruk yöneticisi tetikleme iletisini sunamazsa, program tetiklenmez. Örneğin, süreç nesnesindeki applid, programın artalarında başlatılacağını belirtmelidir; tersi durumda, tetikleme izleme programı programı başlatamaz.

Bir tetikleme iletisi yaratılırsa, ancak başlatma kuyruğuna (örneğin, kuyruk dolu olduğu için ya da tetikleyici iletisinin uzunluğu, başlatma kuyruğu için belirtilen ileti uzunluğu üst sınırından büyük olduğu için) konulamazsa, tetikleme iletisi, ölü harf (teslim edilmemiş ileti) kuyruğunda yerine konmaya neden olur.

Ölü-mektup kuyruğuna koyma işlemi başarıyla tamamlanamazsa, tetikleme iletisi atılır ve  z/OS konsoluna ya da sistem işletmenine bir uyarı iletisi gönderilir ya da hata günlüğüne gönderilir.

Tetikleme iletisini ölüme ilişkin ileti kuyruğuna koymak, o kuyruk için bir tetikleyici iletisi oluşturabilir. Bu ikinci tetikleme iletisi, ölü-mektup kuyruğuna bir ileti eklediğinde atılır.

Program başarıyla tetiklenirse, ancak kuyruktan iletiyi almadan önce olağandışı sona ererse, izleme yardımcı programını kullanın (örneğin, program CICS altında çalışıyorsa, CICS AUXTRACE gibi). başarısızlığın nedenini bulmak için.

## **MQI ve kümelerle çalışma**

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

Aramalar ve kümelerle kullanılacak dönüş kodlarında kullanılabilir olan seçenekler hakkında daha fazla bilgi edinmek için aşağıdaki bağlantıları kullanın:

- [“MQOPEN ve kümeler” sayfa 919](#)
- [“MQPUT, MQPUT1 ve kümeler” sayfa 920](#)
- [“MQINQ ve kümeler” sayfa 920](#)
- [“MQSET ve kümeler” sayfa 921](#)
- [“Dönüş kodları” sayfa 921](#)

### **İlgili kavramlar**

[“Message Queue Interface-Genel Bakış” sayfa 760](#)

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özneliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“Uygulamaların IBM MQ for z/OSüzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

[“IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

## ***MQOPEN ve kümeler***

Bir küme kuyruğunun açılması için bir iletinin konulduğu ya da okunduğu kuyruğun MQOPEN çağrısına bağlı olduğu kuyruk.

## **Hedef kuyruğun seçilmesi**

If you do not provide a queue manager name in the object descriptor, MQOD, the queue manager selects the queue manager to send the message to. Nesne tanımlayıcısında bir kuyruk yöneticisi adı sağlıyorsanız, iletiler her zaman seçtiğiniz kuyruk yöneticisine gönderilir.

Kuyruk yöneticisi hedef kuyruk yöneticisini seçiyorsa, seçim, bağ tanımlama seçeneklerine (MQ00\_BIND\_\*) ve yerel bir kuyruğun varsa, buna bağlıdır. Kuyruğun yerel bir eşgörünümü varsa, CLWLUSEQ özniteliği ANYolarak ayarlanmadıkça, uzak bir yönetim ortamı tercihinde her zaman açılmıştır. Ters durumda, seçim, bağlama seçeneklerine bağlıdır. Gruptaki tüm iletilerin aynı hedefte işlendiğinden emin olmak için kümelerle [ileti grupları](#) kullanılırken MQ00\_BIND\_ON\_OPEN ya da MQ00\_BIND\_ON\_GROUP belirtilmelidir.

Kuyruk yöneticisi hedef kuyruk yöneticisini seçiyorsa, iş yükü yönetimi algoritmasını kullanarak çevrimsel sıralı bir şekilde yapar; bkz. [Kümelerdeki iş yükü dengelemesi](#).

İş yükü dengeleme algoritması kullanıldığında, küme kuyruğunun açılabilmesine bağlıdır:

- MQ00\_BIND\_ON\_OPEN -algoritma, kuyruk uygulama tarafından açıldıktan sonra bir kez kullanılır.
- MQ00\_BIND\_NOT\_FIXED -algoritma, kuyruğa konan her ileti için kullanılır.
- MQ00\_BIND\_ON\_GROUP -algoritma, her ileti grubunun başlangıcında bir kez kullanılır.

## **MQ00\_BIND\_ON\_OPEN**

MQOPEN çağrısındaki MQ00\_BIND\_ON\_OPEN seçeneği, hedef kuyruk yöneticisinin düzeltileceğini belirtir. Bir küme içinde aynı kuyruğun birden çok örneği varsa MQ00\_BIND\_ON\_OPEN seçeneğini kullanın. MQOPEN çağrısından döndürülen nesne tanıtıcısını belirten kuyruğa gönderilen tüm iletiler, aynı kuyruk yöneticisine yönelir.

- İletilerin yakınlıkları varsa, MQ00\_BIND\_ON\_OPEN seçeneğini kullanın. Örneğin, bir ileti grubunun tümü aynı kuyruk yöneticisi tarafından işlenecekse, kuyruğu açtığınızda MQ00\_BIND\_ON\_OPEN değerini belirtin. IBM MQ , kuyruk yöneticisini ve o kuyruğa konarak tüm iletiler tarafından alınacak rotayı düzelir.
- MQ00\_BIND\_ON\_OPEN seçeneği belirtilirse, kuyruk seçilmek üzere yeni bir kuyruk örneği için yeniden açılmalıdır.

## **MQ00\_BIND\_NOT\_FIXED**

MQOPEN çağrısındaki MQ00\_BIND\_NOT\_FIXED seçeneği, hedef kuyruk yöneticisinin düzeltilmediğini belirtir. Messages written to the queue specifying the object handle returned from the MQOPEN call are routed to a queue manager at MQPUT time on a message-by-message basis. Tüm iletilerinizi aynı hedefe yazılacak şekilde zorlamak istemiyorsanız MQ00\_BIND\_NOT\_FIXED seçeneğini kullanın.

- M`QOO_BIND_NOT_FIXED` ve M`QMF_SEGMENTATION_ALLOWED` değerlerini aynı anda belirtmeyin. Bunu yapmazsanız, iletinizin bölümleri farklı kuyruk yöneticilerine teslim edilebilir ve küme boyunca dağılmış olabilir.

### **M`QOO_BIND_ON_GROUP`**

Bir uygulamanın, aynı hedef yönetim ortamına bir ileti grubunun ayrılmasını istemesine izin verir. Bu seçenek yalnızca kuyruklar için geçerlidir ve yalnızca küme kuyruklarını etkiler. Bir küme kuyruğu olmayan bir kuyruk için belirtilirse, bu seçenek yoksayılır.

- M`QPUT` üzerinde M`QPMO_LOGICAL_ORDER` belirtildiğinde gruplar tek bir hedefe yöneltiliyor. M`QOO_BIND_ON_GROUP` belirtildiğinde, ancak bir ileti mantıksal grubun bir parçası değilse, bunun yerine M`BIND_NOT_FIXY` davranışı kullanılır.

### **M`QOO_BIND_AS_Q_DEF`**

M`QOO_BIND_ON_OPEN`, M`QOO_BIND_NOT_FIXED` ya da M`QOO_BIND_ON_GROUP` değerini belirtmezseniz, varsayılan seçenek M`QOO_BIND_AS_Q_DEF` olur. M`QOO_BIND_AS_Q_DEF` kullanılması, kuyruk tanıtıcısı için kullanılan bağ tanımının DefBind kuyruk özniteliğinden alınacağını sağlar.

## **M`QOPEN` seçeneklerinin yakınlığı**

The M`QOPEN` options M`QOO_BROWSE`, M`QOO_INPUT_*`, or M`QOO_SET` require a local instance of the cluster queue for M`QOPEN` to succeed.

The M`QOPEN` options M`QOO_OUTPUT`, M`QOO_BIND_*`, or M`QOO_INQUIRE` do not require a local instance of the cluster queue to succeed.

## **Çözülmüş kuyruk yöneticisi adı**

Bir kuyruk yöneticisi adı M`QOPEN` saatinde çözüldüğünde, çözülen ad uygulamaya geri döndürülür. Uygulama, sonraki bir M`QOPEN` çağrısında bu adı kullanmaya çalışırsa, bu adı kullanmaya yetkili olmadığını ortaya koyabilir.

## **M`QPUT`, M`QPUT1` ve kümeler**

If M`QOO_BIND_NOT_FIXED` is specified on an M`QOPEN` the workload management routines chooses which destination M`QPUT` or M`QPUT1` select.

M`QOPEN` çağrısında M`QOO_BIND_NOT_FIXED` belirtilirse, sonraki her M`QPUT` çağrısı, iletinin hangi kuyruk yöneticisinin gönderileceğini belirlemek için iş yükü yönetimi yordamını çağırır. Alınacak hedef ve rota, iletiyle ileti temelinde seçilir. İleti, ağ değişiminde koşullar ortaya konduktan sonra, hedef ve rota değişebilir. M`QPUT1` çağrısı her zaman M`QOO_BIND_NOT_FIXED` yürürlükte olduğu 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 yönetim ortağıysa, ileti yerel kuyruğun üzerine yerleştirilir.
2. Hedef, bir kümedeki kuyruk yöneticisiyse, ileti bir küme iletim kuyruğuna yerleştirilir.
3. Hedef, bir küme dışında bir kuyruk yöneticisiyse, ileti, hedef kuyruk yöneticisiyle aynı adı taşıyan bir iletim kuyruğuna yerleştirilir.

M`QOPEN` çağrısında M`QOO_BIND_ON_OPEN` belirtilirse, hedef ve rota önceden seçildiği için M`QPUT` çağrıları iş yükü yönetimi yordamını çağırmaz.

## **M`QINQ` ve kümeler**

Hangi küme kuyruğunun sorgulansa, M`QOO_INQUIRE` ile birleştirdiğiniz seçeneklere bağlıdır.

Before you can inquire on a queue, open it using the M`QOPEN` call and specify M`QOO_INQUIRE`.

To inquire on a cluster queue, use the M`QOPEN` call and combine other options with M`QOO_INQUIRE`. Sorgulanabilen öznitelikler, küme kuyruğunun yerel yönetim ortamı olup olmadığına ve kuyruğun nasıl açıldığı ile ilgili olarak değişir:



- MQ00\_BROWSE, MQ00\_INPUT\_\*ya da MQ00\_SET ile MQ00\_INQUIRE ile birleştirilmesi, açılımın başarılı olması için küme kuyruğunun yerel bir yönetim ortamını gerektirir. Bu durumda, yerel kuyruklar için geçerli olan tüm öznitelikleri sorgulayabilirsiniz.
- MQ00\_OUTPUT ile MQ00\_INQUIRE birleştirilerek, önceki seçeneklerden hiçbiri belirtilirse, açılan yönetim ortamı aşağıdakilerden biri olur:
  - Yerel kuyruk yöneticisiyse, varsa, yönetim ortamı. Bu durumda, yerel kuyruklar için geçerli olan tüm öznitelikleri sorgulayabilirsiniz.
  - Yerel kuyruk yöneticisi yönetim ortamı yoksa, kümenin başka bir yerinde yönetim ortamı. Bu durumda yalnızca aşağıdaki öznitelikler sorgulanabilir. Bu durumda, QType özniteliği MQQT\_CLUSTER değerini içerir.
    - DefBind
    - DefPersistence
    - DefPriority
    - InhibitPut
    - QDesc
    - QName
    - QType

To inquire on the DefBind attribute of a cluster queue, use the MQINQ call with the selector MQIA\_DEF\_BIND. Döndürülen değer MQBND\_BIND\_ON\_OPEN ya da MQBND\_BIND\_NOT\_FIXED ya da MQBND\_BIND\_ON\_GROUP olur. Kümeler içeren gruplar kullanılırken MQBND\_BIND\_ON\_OPEN ya da MQBND\_BIND\_ON\_GROUP belirtilmelidir.

To inquire on the KÜME and CLUSNL attributes of the local instance of a queue, use the MQINQ call with the selector MQCA\_CLUSTER\_NAME or the selector MQCA\_CLUSTER\_NAMELIST.

**Not:** Bir küme kuyruğunu MQOPEN ' un bağlı olduğu kuyruğu düzeltmeden açarsanız, sonraki MQINQ çağrılarını, küme kuyruğunun farklı eşgörünümlerini sorgulayabilir.

### İlgili kavramlar

“Küme kuyruğu için MQOPEN seçeneği” sayfa 787

Kuyruk tanıtıcısı için kullanılan bağ tanımı, MQBND\_BIND\_ON\_OPEN, MQBND\_BIND\_NOT\_FIXED ya da MQBND\_BIND\_ON\_GROUP değerini alabilen **DefBind** kuyruk özniteliğinden alınır.

### MQSET ve kümeler

The MQOPEN option MQ00\_SET option requires there to be a local instance of a cluster queue for MQSET to succeed.

You cannot use the MQSET call to set the attributes of a queue elsewhere in the cluster.

Küme öznitelikle tanımlanmış bir yerel diğer ad ya da uzak kuyruk açabilir ve MQSET çağrısını kullanabilirsiniz. Yerel diğer adın ya da uzak kuyruğun özniteliklerini ayarlayabilirsiniz. Hedef kuyruğun, farklı bir kuyruk yöneticisinde tanımlı bir küme kuyruğu olması önemlidir.

### Dönüş kodları

Kümelere özgü dönüş kodları

#### MQRC\_CLUSTER\_EXIT\_ERROR ( 2266 X'8DA ' )

Bir küme kuyruğunu açmak için bir MQOPEN, MQPUT ya da MQPUT1 çağrısı yayınlanır ya da bu iletiyi bir ileti üzerine yerleştirir. Bir kuyruk yöneticisinin ClusterWorkloadExit özniteliği tarafından tanımlanan küme iş yükü çıkışı beklenmedik bir şekilde başarısız olur ya da saat içinde yanıt vermez.

IBM MQ for z/OS üzerindeki sistem günlüğüne bu hatayla ilgili daha fazla bilgi veren bir ileti yazılır.

Bu kuyruk tanıtıcısı için sonraki MQOPEN, MQPUT ve MQPUT1 çağrılarını, ClusterWorkloadExit özniteliğinin boş olmasına rağmen 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 olmasına rağmen işleme devam eder.

**Multi** Multiplatformsüzerinde, bir kuyruk yöneticisine bağlanmak için bir MQCONN ya da MQCONNX çağrısı yayınlanır. Kuyruk yöneticisinin kuyruk yöneticisi ClusterWorkloadExit özniteliği tarafından tanımlanan küme iş yükü çıkışı yüklenemediğinden, çağrı başarısız olur.

## **MQRC\_CLUSTER\_PUT\_INHIBITED ( 2268 X'8DC')**

Bir küme kuyruğu için geçerli olarak MQ00\_OUTPUT ve MQ00\_BIND\_ON\_OPEN seçenekleri ile bir MQOPEN çağrısı yayınlanır. All the instances of the queue in the cluster are currently put-inhibited by having the InhibitPut attribute set to MQQA\_PUT\_INHIBITED. İleti alınabilmekte olan kuyruk örneği olmadığından, MQOPEN çağrısı başarısız olur.

Bu neden kodu, aşağıdaki deyimlerin her ikisi de doğru olduğunda üretilir:

- Kuyruğun yerel eşgörünümü yok. Yerel bir yönetim ortamı varsa, yerel yönetim ortamı engellenmiş olsa bile 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, o yönetim ortamı konulsa bile MQOPEN çağrısı başarılı olur.)

MQOPEN çağrısında MQ00\_BIND\_NOT\_FIXED seçeneği belirtilirse, kümedeki tüm kuyruklar engellense bile arama başarılı olabilir. Ancak, sonraki bir MQPUT çağrısı, bu çağrı sırasında tüm kuyruklar hala engellenmiş olsa da başarısız olabilir.

## **MQRC\_CLUSTER\_RESOLUTION\_ERROR ( 2189 X'88D')**

1. Bir küme kuyruğunu açmak için bir MQOPEN, MQPUTya da MQPUT1 çağrısı yayınlanır ya da bu iletiyi bir ileti üzerine yerleştirir. Tam havuz kuyruk yöneticisinden bir yanıt gerekli olduğundan, ancak hiçbirini kullanılabilir durumda olmadığından, kuyruk tanımlaması doğru şekilde çözümlenemiyor.
2. An MQOPEN, MQPUT, MQPUT1 or MQSUB call is issued for a topic object specifying YAYINLAMA ( ALL) or ALT KAPSAM ( ALL). The cluster topic definition cannot be resolved correctly because a response is required from the full repository queue manager but none is available.

## **MQRC\_CLUSTER\_RESOURCE\_ERROR ( 2269 X'8DD')**

Bir küme kuyruğu için bir MQOPEN, MQPUTya da MQPUT1 çağrısı yayınlanır. Kümeleme için gereken bir kaynağı kullanma girişimi sırasında bir hata oluştu.

## **MQRC\_NO\_DESTINATIONS\_AVAILABLE ( 2270 X'8DE')**

Bir iletiyi küme kuyruğuna koymak için bir MQPUT ya da MQPUT1 çağrısı yayınlanır. Arama sırasında, artık kümede kuyruğun herhangi bir eşgörünümü yok. MQPUT başarısız olur ve ileti gönderilmez.

queue, kuyruğu açan MQOPEN çağrısında MQ00\_BIND\_NOT\_FIXED belirtilmişse ya da iletiyi koymak için MQPUT1 kullanılırsa hata oluşabilir.

## **MQRC\_STOPPED\_BY\_CLUSTER\_EXIT ( 2188 X'88C')**

Bir iletiyi küme kuyruğuna açmak ya da bir küme kuyruğuna yerleştirmek için MQOPEN, MQPUTya da MQPUT1 çağrısı yayınlanır. Küme iş yükü çıkışı çağrısı reddeder.

## **z/OS Uygulamaların IBM MQ for z/OSüzerinde kullanılması ve yazılması**

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

Bu bilgilerde, desteklenen ortamların her birinde çalışan programlara ilişkin IBM MQ olanakları açıklanır. Buna ek olarak,

- IBM MQ-CICS bridgekullanımına ilişkin bilgi için bkz. [Using IBM MQ with CICS](#).

- [IMS ve IMS köprüsünün kullanılmasıyla ilgili bilgi için bkz. “IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları” sayfa 65.](#)

IBM MQ for z/OSüzerindeki uygulamaları kullanma ve yazma hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- [“Ortama bağımlı IBM MQ for z/OS işlevleri” sayfa 923](#)
- [“Hata ayıklama tesisleri, eşitleme noktası desteği ve kurtarma desteği” sayfa 924](#)
- [“Uygulama ortamıyla IBM MQ for z/OS arabirimi” sayfa 925](#)
- [“z/OS UNIX System Services uygulamaları yazılıyor” sayfa 926](#)
- [“Paylaşılan kuyruklarla uygulama programlama” sayfa 929](#)

### **İlgili kavramlar**

[“Message Queue Interface-Genel Bakış” sayfa 760](#)

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“IBM MQ for z/OSüzerindeIMS ve IMS köprüsü uygulamaları” sayfa 65](#)

This information helps you to write IMS applications using IBM MQ.

### **Ortama bağımlı IBM MQ for z/OS işlevleri**

IBM MQ for z/OS işlevlerini dikkate aldığımızda bu bilgileri kullanın.

IBM MQ for z/OS ' in çalıştırdığı ortamlardaki IBM MQ işlevleri arasında dikkate alınacak temel farklar şunlardır:

- IBM MQ for z/OS , aşağıdaki tetikleme izleyicileri sağlar:

- CICS ortamında kullanım için CKTI
- CSQQTRMN, IMS ortamında kullanım için

Diğer ortamlardaki uygulamaları başlatmak için kendi modülünüzü yazmanız gerekir.

- İki aşamalı kesinleştirmeyi kullanan eşitleme, CICS ve IMS ortamlarında desteklenir. Bu, hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri (RRS) kullanılarak z/OS toplu iş ortamında da desteklenir. Single-phase commit is supported in the z/OS environment by IBM MQ itself.
- Toplu ve IMS ortamlarında, MQI, bağlantı programlarını ve kuyruk yöneticisinden olan bağlantıları kesmeye yönelik çağrılar sağlar. Programlar, birden çok kuyruk yöneticisine bağlanabilirler.

- Bir CICS sistemi tek bir kuyruk yöneticisine bağlanabiliyor. Altsistem adı CICS sistemi başlatma işinde tanımlandıysa, CICS başlatıldığında bu durum oluşabilir. The MQI connect and disconnect calls are tolerated, but have no effect, in the CICS environment.
- API geçiş çıkışı, bir programın tüm MQI çağrılarının işlenmesine müdahalede bulunmasını sağlar. Bu çıkış yalnızca CICS ortamında kullanılabilir.
- Çoklu işlemci sistemlerinde CICS 'ta, birden çok z/OS TCB' nin altında MQI çağrıları yürütülebileceği için bazı performans avantajı elde edilir. Daha fazla bilgi için [Planning on z/OS IBM MQ for z/OS Concepts and Planning Guide](#) adlı yayına bakın.

Bu özellikler Çizelge 134 sayfa 924' de özetlenmiştir.

Çizelge 134. z/OS ortam aksamaları			
	CICS	IMS	Küme/TSO
Tetikleyici izleyicisi sağlandı	Evet	Evet	Hayır
İki aşamalı kesinleştirme	Evet	Evet	Evet
Tek aşamalı kesinleştirme	Evet	Hayır	Evet
Bağlan ve bağlantı kesme MQI çağrıları	Tolerans	Evet	Evet
API geçiş çıkışı	Evet	Hayır	Hayır

**Not:** RRS ' yi kullanarak Batch/TSO ortamında iki aşamalı kesinleştirme desteklenir.

### ***Hata ayıklama tesisleri, eşitleme noktası desteği ve kurtarma desteği***

Program hata ayıklama olanakları, uyumluluk noktası desteği ve kurtarma desteği hakkında bilgi edinmek için bu bilgileri kullanın.

### **Program hata ayıklama olanakları**

IBM MQ for z/OS , tüm ortamlardaki programlarınızda hata ayıklamak için kullanabileceğiniz bir izleme olanağı sağlar.

Additionally, in the CICS environment you can use:

- CICS Execution Diagnostic Facility (CEDF)
- CICS İzleme Denetimi İşlemi (CEETR)
- IBM MQ for z/OS API-geçiş çıkışı

z/OS platformunda, kullanmakta olduğunuz programlama dili tarafından desteklenen kullanılabilir etkileşimli hata ayıklama aracını kullanabilirsiniz.

### **Syncpoint desteği**

Hareket işleme ortamında, işlem işlemlerinin güvenli bir şekilde kullanılabilmesi için, iş birimlerinin başlangıç ve bitiş zamanlarının uyumlaştırılması gereklidir.

Bu, CICS ve IMS ortamlarında IBM MQ for z/OS tarafından tam olarak desteklenir. Tam destek, kaynak yöneticileri arasında iş birimlerinin CICS ya da IMSdenetimi altındaki unison içinde kesinleştirilebilmesi ya da yedeklenebilmesi için işbirliği anlamına gelir. Kaynak yöneticisi örnekleri şunlardır: Db2, CICS File Control, IMS, ve IBM MQ for z/OS.

z/OS toplu iş uygulamaları, tek aşamalı kesinleştirme olanağı vermek için IBM MQ for z/OS çağrılarını kullanabilir. Bu, diğer kaynak yöneticilerine başvuru yapılmaksızın, uygulama tanımlı bir kuyruk işlemleri kümesinin kesinleştirilebileceği ya da yedeklenebileceği anlamına gelir.

İşlem yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri (RRS) kullanılarak z/OS toplu iş ortamında iki aşamalı kesinleştirme de desteklenir. Daha fazla bilgi için bkz. [z/OS toplu iş uygulamalarındaki eşitler](#).

## Kurtarma desteđi

Bir iřlem sırasında bir kuyruk yneticisi ile CICS ya da IMS sistemi arasındaki bađlantı kesilirse, bazı iř birimleri bařarıyla yedeklenmeyebilir.

Ancak, bu iř birimleri, CICS ya da IMS sistemiyle bađlantısı yeniden kurulduđunda, kuyruk yneticisi (syncpoint manager 'ın denetimi altında) tarafından zlr.

## Uygulama ortamıyla IBM MQ for z/OS arabirimi

Farklı ortamlarda alıřan uygulamaların ileti kuyruklama ađı aracılıđıyla ileti gndermelerine ve almasına izin vermek iin IBM MQ for z/OS , desteklediđi ortamlar iin bir *bađdařtırıcı* sađlar.

Bu bađdařtırıcılar, uygulama programları ve IBM MQ for z/OS altsistemleri arasındaki arabirimdir. Bu programlar, programların MQI ' yı kullanmalarına izin verir.

### Toplu iř bađdařtırıcısı

Toplu iř bađdařtırıcısıyla ve desteklediđi kesinleřtirme protokolleriyle ilgili bilgi edinmek iin bu bilgileri kullanın.

*Toplu iř bađdařtırıcısı* , ařađdakiler iinde alıřan programlar iin IBM MQ for z/OS kaynaklarına eriřim sađlar:

- Grev (TCB) kipi
- Sorun ya da gzetmen durumu
- Birincil adres alanı denetim kipi

Programlar, apraz bellek kipinde olmamalıdır.

Uygulama programları ile IBM MQ for z/OS arasındaki bađlantılar grev dzeyinde bulunur. Bađdařtırıcı, bir uygulama grev denetim blođundan (TCB) IBM MQ for z/OS' e tek bir bađlantı iř paracıđı sađlar.

Bađdařtırıcı, IBM MQ for z/OS ' e ait kaynaklarda yapılan deđiřiklikler iin tek ařamalı kesinleřtirme protokoln destekler. oklu faz kesinleřtirme protokollerini desteklemez.

### RRS toplu iř bađdařtırıcısı

RRS toplu bađdařtırıcısı ve IBM MQ tarafından sađlanan iki RRS toplu iř bađdařtırıcısıyla ilgili bilgi edinmek iin bu bilgileri kullanın.

Hareket ynetimi ve kurtarılabilir kaynak yneticisi hizmetleri (RRS) bađdařtırıcısı:

- Kesinleřtirme denetimi iin z/OS RRS ' yı kullanır.
- Tek bir grevdeki tek bir z/OS ynetim ortamında alıřan birden ok IBM MQ altsistemi iin eřzamanlı bađlantıları destekler.
- z/OS RRS ile uyumlu kurtarılabilir yneticiler aracılıđıyla eriřilen kurtarılabilir kaynaklar iin z/OS-wide eřgdml kesinleřtirme denetimi ( z/OS RRS kullanılarak) sađlar:
  - RRS toplu iř bađdařtırıcısını kullanarak IBM MQ ' e bađlanan uygulamalar.
  - Db2-stored procedures executing in a Db2-stored procedures address space that is managed by a workload manager (WLM) on z/OS.
- TCB ' ler arasında bir IBM MQ toplu iř paracıđı deđiřtirme yeteneđini destekler.

IBM MQ for z/OS , iki adet RRS toplu iř bađdařtırıcısı sađlar:

### CSQBRSTB

This adapter requires you to change any MQCMIT statement to SRRCMIT and any MQBACK statement to SRRBACK in your IBM MQ application. (CSQBRSTB ile bađlantılı bir uygulamaya MQCMIT ya da MQBACK kodunu kodladıysanız, MQRC\_ENVIRONMENT\_ERROR alırsınız.)

### CSQBRSI

Bu bađdařtırıcı, IBM MQ uygulamanızın MQCMIT ve MQBACK ya da SRRCMIT ve SRRBACK ' ı kullanmasını sađlar.

**Not:** CSQBRSTB ve CSQBRSI, AMODE (31) RMODE (ANY) bađ öznelikleriyle birlikte gönderilir. Uygulamanız 16 MB ' lik satırın altında sınırlı kod öbeđini yüklerse, önce sınırlı kod öbeđini RMODE ile yeniden bađlayın (24).

## Geçiş

Var olan Batch/TSO IBM MQ uygulamalarını RS koordinasyonunu birkaç deđişlikle ya da deđişiksiz olarak kullanmak için geçirebilirsiniz.

If you link-edit your IBM MQ application with the CSQBRSI adapter, MQCMIT and MQBACK syncpoint your unit of work across IBM MQ and all other RRS-enabled resource managers. IBM MQ uygulamanızı CSQBRSTB bađdaştırıcısına bađlayıp düzenseniz, MQCMIT 'yi SRRRCMIT ve MQBACK ile SRRBACK' a çevirin. İkinci yaklaşım tercih edilir; bu yaklaşım, eşitleme noktasının yalnızca IBM MQ kaynaklarıyla sınırlı olmadığını açıkça gösterir.

### IMS bađdaştırıcısı

If you are using the IMS adapter from an IBM MQ for z/OS system, ensure that IMS can obtain sufficient storage to accommodate messages up to 100 MB long.

## Kullanıcılara not

IMS bađdaştırıcısı , aşağıdakiler için IBM MQ for z/OS kaynaklarına erişim sađlar:

- Çevrimiçi ileti işleme programları (MPP ' ler)
- Etkileşimli hızlı yol programları (IFP ' ler)
- Toplu ileti işleme programları (BMP)

Bu kaynakları kullanmak için, programların görev (TCB) kipinde çalıştırılması ve sorun durumu olması gerekir; bunlar, çapraz bellek kipinde ya da erişim kaydı kipinde olmamalıdır.

Bađdaştırıcı, bir uygulama görevi denetim bloğundan (TCB) IBM MQ' e bir bađlantı iş parçacığı sađlar. The adapter supports a two-phase commit protocol for changes made to resources owned by IBM MQ for z/OS, with IMS acting as the syncpoint coordinator.

Bađdaştırıcı ayrıca, kuyruklardaki belirli tetikleme koşulları karşılandığında otomatik olarak programları başlatabilen bir tetikleme izleme programı sađlar. Daha fazla bilgi için [“Starting IBM MQ applications using triggers” sayfa 899](#) başlıklı konuya bakın.

Toplu DL/I programları yazıyorsanız, z/OS toplu iş programları için bu konuda verilen kılavuzları izleyin.

## z/OS UNIX System Services uygulamaları yazılıyor

Toplu iş bađdaştırıcısı, toplu iş ve TSO adres alanlarından kuyruk yöneticisi bađlantılarını destekler:

Bir Toplu İş adresi alanı göz önünde bulundurursak, bađdaştırıcı, bu adres alanı içindeki birden çok TCB ' den gelen bađlantıları aşağıdaki gibi destekler:

- Her TCB, MQCONN ya da MQCONNX çağrısını kullanan birden çok kuyruk yöneticisine bađlanabilir (ancak, bir TCB ' nin belirli bir kuyruk yöneticisine bir bađlantının yalnızca bir eşgörünümü olabilir).
- Birden çok TCB, aynı kuyruk yöneticisine bađlanabilir (ancak, herhangi bir MQCONN ya da MQCONNX çağrısının döndürdüğü kuyruk yöneticisi tanıtıcısı, TCB yayınına bađlanır ve başka bir TCB tarafından kullanılamaz).

z/OS UNIX System Services iki tip pthread\_create çağrısını destekler:

1. Ağır sıklet iş parçacıkları, her bir TCB için bir tane çalıştırın; bu, iş parçacığı başlangıcındaki ATTACHed ve DETACHed, z/OS ile biten bir iş parçacığıdır.
2. Orta ağırlıklı iş parçacıkları, her TCB için bir tane çalıştırın, ancak TCB, uzun süredir çalışan toplam sahip toplam sahip olma maliyeti havuzlarından biri olabilir. Uygulamanın tüm gerekli uygulama temizliğini gerçekleştirmesi gerekir; çünkü, bir sunucuya bađlıysa, görev (TCB) sonlandırmasında sunucu tarafından sađlanabilen varsayılan iş parçacığı sonlandırması, **deđil** her zaman yönlendirilir.

Hafif iş parçacıkları desteklenmez. (Bir uygulama kendi iş isteklerini dağıtan kalıcı iş parçacıkları yaratıyorsa, sonraki iş isteğini başlatmadan önce kaynakların temizlenmesinden **uygulama** sorumlu olur.)

IBM MQ for z/OS , Toplu Bağdaştırıcı 'ı kullanarak z/OS UNIX System Services iş parçacıklarını destekler:

1. Ağır sıklet iş parçacıkları toplu iş bağlantıları olarak tam olarak desteklenir. Her bir iş parçacığı kendi TCB 'de çalışır ve bu, iş parçacığı başında ve sonunda bağlanan ve ayrıık olan TBB' de çalışır. Bir MQDISC çağrısı yayınlamadan önce iş parçacığının sona ermesi durumunda, IBM MQ for z/OS standart görev temizliğini gerçekleştirir; iş parçacığı olağan biçimde sona erdirildiyse, olağandışı iş birimi kesinleştirmeyi ya da iş parçacığının olağan dışı bir şekilde sonlandırılıp sonlandırılması durumunda yedeklenmeleri de vardır.
2. Orta ağırlıklı iş parçacıkları tam olarak desteklenir, ancak TCB başka bir iş parçacığı tarafından yeniden kullanılacaksa, uygulamanın sonraki iş parçacığı başlatılmadan önce MQDISC ya da MQBACK tarafından önce bir MQDISC çağrısı yayınlandığından emin olması gerekir. Bu, uygulama bir Program Interrupt İşleyicisi oluşturduysa ve uygulama daha sonra olağandışı sona ererse, Interrupt Handler 'ın başka bir iş parçacığı için TCB ' yi yeniden kullanmadan önce MQCMIT ve MQDISC çağrılarını yayınlaması gerektiğini belirtir.

**Not:** Bu modeller, birden çok iş parçacığının ortak IBM MQ kaynaklarına erişimi **desteklememektedir** .

### ***z/OS için API geçişi çıkışı***

Bu konuda, ürüne duyarlı programlama arabirimi bilgileri yer alır.

Çıkış, IBM tarafından 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 geçişi çıkışı* sağlar. Bu bölümde, API geçidi çıkışının nasıl kullanılacağı ele alınmıştır ve IBM MQ for z/OS ile birlikte verilen örnek çıkış programını nasıl tanımların açıklanır.

Bu bölüm yalnızca CICS TS V3.1 ve önceki yayın düzeylerinden oluşan kullanıcılar için geçerlidir. CICS TS V3.2 kullanıcıları ve daha sonraki bir sürümü, CICS ürün belgelerindeki IBM MQ ile CICS ile Bütünleşme başlıklı bölüme başvurmalıdır.

### **Not**

The API-crossing exit is invoked only by the CICS adapter of IBM MQ for z/OS. Çıkış programı CICS adres alanında çalışır.

#### *Kendi çıkış programınızı yazmak*

Kendi programınız için bir çerçeve olarak IBM MQ for z/OS ile birlikte sağlanan örnek API geçişi çıkış programını (CSQCAPX) kullanabilirsiniz.

Bu, "[Örnek API-geçiş çıkış programı, CSQCAPX](#)" sayfa 928 içinde açıklanmaktadır.

Bir çıkış programı yazarken, bir uygulama tarafından yayınlanan bir MQI çağrısının adını bulmak için, MQXP yapısının *ExitCommand* alanını inceleyin. Çağrıdaki parametrelerin sayısını bulmak için *ExitParmCount* alanını inceleyin. Uygulamanın edindiği herhangi bir dinamik deponun adresini saklamak için 16 baytlık *ExitUserArea* alanını kullanabilirsiniz. This field is retained across invocations of the exit and has the same lifetime as a CICS task.

CICS Transaction Server V3.2 kullanıyorsanız, çıkış programınızı iş parçacığı korumalı olarak yazmanız ve iş parçacığı güvenliği olarak çıkış programınızı bildirmeniz gerekir. If you are using earlier CICS releases, you are also recommended to write and declare your exit programs as threadsafe to be ready for migrating to CICS Transaction Server V3.2.

Çıkış programınız, *ExitResponse* alanına MQXCC\_SUPPRESS\_FUNCTION ya da MQXCC\_SKIP\_FUNCTION döndürerek bir MQI çağrısının yürütülmesini engelleyebilir. Çağrılarının yürütülmesine izin vermek (ve çağrı tamamlandıktan sonra çıkış programının yeniden çağrılmasına izin vermek için), çıkış programınızın MQXCC\_OK değerini döndürmesi gerekir.

Bir MQI çağrısından sonra çağrıldığında, bir çıkış programı çağrıya göre belirlenen tamamlanma ve neden kodlarını inceleyebilir ve değiştirebilir.

## Kullanım notları

Çıkış programınızı yazarken dikkate almanız gereken bazı genel noktalar aşağıda yer alıyor:

- Başarım nedenlerinden dolayı, programınızı çevirici diline yazın. Bunu IBM MQ for z/OS tarafından desteklenen diğer dillerden birine yazarsanız, kendi veri tanımlama dosyanızı sağlamanız gerekir.
- Programınızı AMODE (31) ve RMODE (ANY) olarak bağlantı düzenleyin.
- Programınıza çıkış parametre bloğunu tanımlamak için, çevirici dili makrosunu (CMQXPA) kullanın.
- Çıkış programınızı ve çıkış programınızın çağırdığı programlardan birini tanımlarken CONCURRENCE (THREADSAFE) (İş parçacığı güvenliği) belirtin.
- CICS Transaction Server for z/OS depolama koruma özelliğini kullanıyorsanız, programınız CICS yürütme anahtarında çalıştırılmalıdır. Yani, EXECKEY ( CICS ) belirtmelisiniz. Hem çıkış programınızı, hem de denetimi geçtiği programlardan birini tanımlarken. CICS çıkış programları ve CICS depolama koruma tesisine ilişkin bilgi için *CICS Customization Guide* adlı belgeye bakın.
- Programınız tüm API ' leri kullanabilir (örneğin, IMS, Db2 ve CICS ) CICS görevine ilişkin bir kullanıcı çıkış programının kullanabileceği bir program. MQCONN, MQCONNX ve MQDISC dışındaki MQI çağrılarında herhangi birini de kullanabilir. Ancak, çıkış programı içindeki tüm MQI çağrıları çıkış programını ikinci kez çağırmaz.
- Programınız EXEC CICS SYNCPOINT ya da EXEC CICS SYNCPOINT ROLLBACK komutlarıyla yayınlanabilir. Ancak, bu komutlar, **Tümü** ' un çıkış tarafından çıkılan noktaya kadar yapılan güncellemeleri kesinleştirir ya da geri alabilir ve bu nedenle kullanımlarının önerilmemesi önerilir.
- Programınız EXEC CICS RETURN komutu yayınlayarak sona ermelidir. Bir XCTL komutu ile denetim aktarılmamalı.
- Çıkışlar, IBM MQ for z/OS koduna uzantılar olarak yazılır. Çıkışınızın, MQI kullanan tüm IBM MQ for z/OS programlarını ya da işlemlerini bozmadığından emin olun. Bunlar genellikle CSQ ya da CK önekiyle gösterilir.
- If CSQCAPX is defined to CICS, the CICS system attempts to load the exit program when CICS connects to IBM MQ for z/OS. If this attempt is successful, message CSQC301I is sent to the CKQC panel or to the system console. Yükleme başarısız olursa (örneğin, yükleme modülü DFHRPL birleştirmede kitaplıkların hiçbirinde yoksa), CKQC panosuna ya da sistem konsoluna CSQC315 iletisi gönderilir.
- Because the parameters in the communication area are addresses, the exit program must be defined as local to the CICS system (that is, not as a remote program).

*Örnek API-geçiş çıkış programı, CSQCAPX*

Örnek çıkış programı bir çevirici dili programı olarak sağlanır. Kaynak dosya (CSQCAPX), **thlqual**.SCSQASMS kitaplığında bulunur (burada **thlqual** , kuruluşunuz tarafından kullanılan üst düzey niteleyicidir). Bu kaynak dosya, program mantığını tanımlayan takma ad kodunu içerir.

Örnek program, kendi çıkış programlarınızı yazarken kullanabileceğiniz başlatma kodu ve yerleşim düzeni içerir.

Bu örnek, aşağıdakileri nasıl göstereceğini gösterir:

- Çıkış değiştirgesi öbeğini ayarla
- Çağrıyı ele al ve parametre öbeklerini çıkar
- Çıkışa çağrılan MQI çağrısının saptanması
- Çıkışa, MQI çağrısının işlenmesinden önce ya da sonra çağrılıp çağrılmadığını saptayın.
- CICS geçici depolama kuyruğuna ileti koy
- Yeniden girişmeyi sürdürmek için dinamik depolama edinimi için DFHEIENT makrosunu kullanın
- CICS exec arabirimi denetim bloğu için DFHEIBLK kullanın
- Tuzak hata koşulları
- Denetime geri dönüş denetimi



## Örnek çıkış programının tasarımı

Örnek çıkış programı, çıkışa ilişkin işlemi göstermek için iletilerin CICS geçici depolama kuyruğuna (CSQ1EXIT) yazar.

İletiler, çıkışa, MQI çağrısından önce ya da sonra çağrılmakta olup olmadığını gösterir. Arama işleminden sonra çıkış çağrılırsa, ileti, arama işleminin döndürdüğü tamamlanma kodunu ve neden kodunu içerir. Bu örnek, giriş tipini (yani, çağrıdan önce ya da sonra) denetlemek için CMQXPA makrosundan adlandırılan değişmezleri kullanır.

Bu örnek izleme işlevini gerçekleştirmez; ancak, zaman damglanmış iletileri, işlediği çağrı tipini belirten bir CICS kuyruğuna yerleştirmektedir. Bu, MQI ' nin performansının yanı sıra çıkış programının doğru işleyişi hakkında bir gösterge sağlar.

**Not:** Örnek çıkış programı, program çalışırken yapılan her bir MQI çağrısı için altı EXEC CICS çağrısını yayınlar. Bu çıkış programını kullanırsanız, IBM MQ for z/OS performansının düşmesini sağlar.

### *API geçiş çıkışının hazırlanması ve kullanılması*

Örnek çıkış yalnızca kaynak formda sağlanır.

Örnek çıkışı ya da yazdığınız bir çıkış programını kullanmak için, "z/OS'te CICS uygulamaları oluşturma" sayfa 1063' ta açıklandığı gibi başka bir CICS programı için olduğu gibi bir yükleme kitaplığı yaratın.

- For CICS Transaction Server for z/OS and CICS for MVS/ESA, when you update the CICS system definition (CSD) data set, the definitions you need are in the member **thlqual.SCSQPROC(CSQ4B100)**.

**Not:** Tanımlamalar MQsonekini kullanır. Kuruluşunuzda bu sonek önceden kullanıldıysa, bu, montaj aşamasından önce değiştirilmelidir.

Sağlanan varsayılan CICS program tanımlarını kullanırsanız, CSQCAPX çıkış programı bir **devre dışı** durumunda kurulur. Bunun nedeni, çıkış programının kullanılması performansında önemli bir azalma elde edebileceğidir.

API geçitini geçici olarak etkinleştirmek için:

1. Issue the command **CEMT S PROGRAM(CSQCAPX) ENABLED** from the CICS master terminal.
2. CKQC işlemini çalıştırın ve API geçiş çıkışının durumunu **Enabled**(Etkin) değerine değiştirmek için Bağlantı çekme sırasında seçenek 3 'i kullanın.

API için CICS Transaction Server for z/OS ve CICS for MVS/ESA ile birlikte IBM MQ for z/OS ' u API geçiş çıkışı ile birlikte çalıştırmak istiyorsanız aşağıdakilerden birini yapın:

- CSQ4B100 üyesinde CSQCAPX tanımlamasını değiştirin, STATUS (DISABLE) değerini değiştirerek (ENABLE) (ENABLE). CICS CSD tanımlamasını CICS sağlanan toplu iş programı DFHCSDUP kullanarak güncelleyebilirsiniz.
- CSQCAT1 grubundaki CSQCAPX tanımlamasını değiştirerek, DISABIN durumunu ENABLED durumuna değiştirerek değiştirin.

Her iki durumda da, grubu yeniden kurmanız gerekir. You can do this by cold-starting your CICS system or by using the CICS CEDA transaction to reinstall the group while CICS is running.

**Not:** Gruptaki girişlerden herhangi biri şu anda kullanılıyorsa, CEDA ' nın kullanılması bir hataya neden olabilir.

Ürüne duyarlı programlama arabirimi bilgilerinin sonu.

### ***Paylaşılan kuyruklarla uygulama programlama***

Bu konu, paylaşılan kuyrukları kullanmak için yeni uygulamalar tasarlarken ve var olan uygulamaları paylaşılan kuyruk ortamına geçirirken dikkate almak için gereksinim duyduğunuz bazı etkenlere ilişkin bilgi sağlar.

### *Uygulamalarınızın diziselleştirilmesi*

Bazı uygulama tipleri, iletilerin kuyruktan vardıkları sırayla, kuyruktan tam olarak alındığından emin olmak zorunda kalabilirler.

Örneğin, IBM MQ veri tabanı güncellemelerini uzak bir sisteme yönelik olarak gölgelemek için kullanılıyorsa, bu kaydın eklenmesini açıklayan bir iletinin ardından, güncellemeyi bir kayda açıklayan bir ileti işlenmelidir. Yerel bir kuyruğa alma ortamında, genellikle kuyruğun MQOO\_INPUT\_EXCLUSIVE seçeneği ile açılmasına neden olan uygulama tarafından elde edilen bu uygulama, aynı anda başka bir uygulamanın kuyruğun işlenmesini engelleyerek elde edilen bir uygulama tarafından gerçekleştirilir.

IBM MQ , uygulamaların paylaşılan kuyrukları yalnızca aynı şekilde açmasını sağlar. Ancak, uygulama bir kuyruğun bir bölümünden çalışıyorsa (örneğin, tüm veritabanı güncellemeleri aynı kuyruğdaysa, ancak A çizelgesine ilişkin ilinti tanıtıcısı A, B çizelgesi B ilinti tanıtıcısı ise) ve uygulamalar, bir güncelleme ve çizelge B güncellemeleri için koşut zamanlı olarak ileti almak istiyor. Kuyruğun açılmasına ilişkin basit bir düzenek mümkün değil.

Bu uygulama tipi, paylaşılan kuyrukların yüksek düzeyde kullanılabilirliğini kullanacaksa, ikincil bir kuyruk yöneticisine çalışan aynı paylaşılan kuyruklara erişen başka bir uygulamanın yönetim ortamının, birincil uygulama ya da kuyruk yöneticisi alma işlemi başarısız olursa devralması gerektiğine karar verebilirsiniz.

Birincil kuyruk yöneticisi başarısız olursa, iki şey olur:

- Paylaşılan kuyruk eşdüzey kurtarma işlemi, birincil uygulamadaki tamamlanmamış güncellemelerin tamamlanmasını ya da yedeklenmesini sağlar.
- İkincil uygulama, kuyruğun işlenmesini devralıyor.

İkincil uygulama, tüm tamamlanmamış iş birimlerinin işlenmesinden önce başlayabilir. Bu işlem, iletilerin sıralamadan alınması ikincil uygulamaya yol açabilir. Bu tür bir sorunu çözmek için uygulama, *serileştirilmiş uygulama* olarak seçilebilir.

Diziselleştirilmiş bir uygulama, kuyruk yöneticisine bağlanmak için MQCONNX çağrısını kullanır; bağlantı etiketi, bu uygulama için benzersiz bir bağlantı etiketi sağlar. Uygulama tarafından gerçekleştirilen iş birimleri, bağlantı etiketiyle işaretlenir. IBM MQ , aynı bağlantı etiketiyle kuyruk paylaşım grubundaki iş birimlerinin diziselleştirilmesini sağlar ( MQCONNX çağrısında diziselleştirme seçeneklerine göre).

This means that, if the primary application uses the MQCONNX call with a connection tag of Database shadow retriever, and the secondary takeover application attempts to use the MQCONNX call with an identical connection tag, the secondary application cannot connect to the second IBM MQ until any outstanding primary units of work have been completed, in this case by peer recovery.

Bir kuyruktaki tam ileti dizisine bağlı uygulamalar için diziselleştirilmiş uygulama tekniğini kullanmayı düşünün. Özellikle:

- Uygulamanın önceki yürütmesine ilişkin tüm kesinleştirme ve geri alma işlemleri tamamlanmaya kadar, bir uygulama ya da kuyruk yöneticisi hatasından sonra yeniden başlatılmaması gereken uygulamalar.

Bu durumda, diziselleştirilmiş uygulama tekniği yalnızca uygulama syncpoint 'te çalışırsa geçerlidir.

- Aynı uygulamanın başka bir örneği zaten çalışıyor, ancak başlatılamaması gereken uygulamalar.

Bu durumda, diziselleştirilmiş uygulama tekniği yalnızca uygulama, dışlayıcı giriş için kuyruğu açamazsa gereklidir.

**Not:** IBM MQ yalnızca belirli ölçütler karşılandığında ileti sırasını korumayı garanti eder. Bunlar, [MQGETile](#) ilgili açıklamayla açıklanmaktadır.

### *Paylaşılan kuyruklarla kullanım için uygun olmayan uygulamalar*

Paylaşılan kuyruklar kullanıyorsanız, 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 dizini oluşturma, paylaşılan kuyruklar için sınırlanmıştır. İleti tanıtıcısını ya da ilinti tanıtıcısını, kuyruktan almak istediğiniz iletiyi seçmek için kullanmak istiyorsanız, kuyruk doğru değerle dizinlenmelidir. İleti tanıtıcısı temelinde tek başına ileti seçiyorsanız, kuyruğun bir MQIT\_MSG\_ID dizin

tipine gerek vardır (MQIT\_NONE da kullanılabilir). Tek başına ilinti tanıtıcısı temelinde ileti seçiyorsanız, kuyruğun izin tipi MQIT\_COREL\_ID olmalıdır.

- Geçici dinamik kuyrukları paylaşılan kuyruklar olarak kullanamazsınız. Ancak, kalıcı dinamik kuyrukları kullanabilirsiniz. Paylaşılan dinamik kuyruklara ilişkin modeller, PERMDYN (kalıcı dinamik) kuyruklar ile aynı şekilde yaratılıp yok edilmelerine rağmen, SHAREDYN (paylaşılan dinamik) DEFTYPE modellerine sahiptir.

*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 135 sayfa 932](#).

### **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 iletilisi, 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/OS için sistem nesnelerini tanımlama başlıklı konuya](#) bakın.

Başka bir SYSTEM.\* tanımlayamazsınız kuyruklar, paylaşılan kuyruklar olarak.

*Var olan uygulamalarınızı paylaşılan kuyrukları kullanacak şekilde geçirme*

Neden kodları, tetikleme ve MQINQ API çağırısı, 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 135 sayfa 932](#) içinde, hangi başlatma kuyruğu tipinin kullanılacağına karar verirken göz önünde bulundurmanız gereken noktalar açıklanır.

Çizelge 135. Paylaşılan başlatma kuyruğunun ne zaman kullanılacağı

	Paylaşılmayan uygulama kuyruğu	Paylaşılan uygulama kuyruğu
<b>Paylaşılmayan başlatma kuyruğu</b>	Önceki yayınlara gelince.	<p>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 bridge gibi) tetiklemek için iyidir ve yüksek kullanılabilirlik sağlar.</p> <p>Tetikleyici tipi FIRST ya da DEPTH için, tetikleyici iletileri, tetikleyici izleme programı çalıştıran ve giriş için uygulama kuyruğu açık olmayan her kuyruk yöneticisinde uygulamanın bir eşgörünümünü tetikler. Her kuyruk yöneticisi için bir tetikleyici iletileri üretilir; paylaşılmayan yerel başlatma kuyruğuna karşı çalışan birden fazla tetikleyici varsa, belirli bir kuyruk yöneticisinde bu tetikleyici, iletileri işlemek için rekabet eder.</p>
<b>Paylaşılan başlatma kuyruğu</b>	Paylaşılmayan bir uygulama kuyruğuyla paylaşılan başlatma kuyruğu kullanmayın.	<p>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 iletileri oluşturulmasıdır.</p> <p><b>Not:</b> 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.</p> <p>FIRST ya da DEPTH tetikleyici tipi için, adı belirtilen başlatma kuyruğu giriş için açık olan her kuyruk yöneticisi tarafından bir tetikleyici iletileri üretilir.</p> <p><b>Not:</b> 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 iletilerini işlemesi olasılığını bırakır. Bu nedenle, belirli bir kuyruk yöneticisi için sunucu uygulamasının birden çok eşgörünümü başlatılabilir. Birden çok tetikleyici iletilerinin 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 iletileri oluşturulmaz.</p>

### 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.

This information helps you to write IMS applications using IBM MQ.

- IMS uygulamalarındaki eşitleme noktalarını ve MQI çağrılarını kullanmak için bkz. [“IBM MQkullanarak IMS uygulamaları yazılıyor” sayfa 66.](#)
- IBM MQ - IMS köprüsünü kullanan uygulamaları yazmak için bkz. [“IMS köprü uygulamaları yazılıyor” sayfa 70.](#)

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

- [“IBM MQkullanarak IMS uygulamaları yazılıyor” sayfa 66](#)
- [“IMS köprü uygulamaları yazılıyor” sayfa 70](#)

### **İlgili kavramlar**

[“Message Queue Interface-Genel Bakış” sayfa 760](#)

Message Queue Interface (MQI) bileşenleri hakkında bilgi edinin.

[“Kuyruk yöneticisine bağlanma ve kuyruk yöneticisinden bağlantı kesme” sayfa 773](#)

IBM MQ programlama hizmetlerini kullanmak için, bir programın kuyruk yöneticisiyle bağlantısı olması gerekir. Bir kuyruk yöneticisinden bağlantı kurulabilmek ve kuyruk yöneticisinden nasıl bağlantı kurulacağını öğrenmek için bu bilgileri kullanın.

[“Nesnelerin açılması ve kapatılması” sayfa 781](#)

Bu bilgiler, IBM MQ nesnelerini açmak ve kapatmak için bir kavrayış sağlar.

[“İletileri Kuyruğa Koyma” sayfa 791](#)

İletilerin kuyruğa nasıl konacağını öğrenmek için bu bilgileri kullanın.

[“Kuyruktan İleti Alınması” sayfa 806](#)

Kuyruktan ileti alma hakkında bilgi edinmek için bu bilgileri kullanın.

[“Nesne özniteliklerinin sorulmasına ve ayarlanmasını geciktirme” sayfa 885](#)

Öznitelikler, bir IBM MQ nesnesinin özelliklerini tanımlayan özelliklerdir.

[“İş birimlerinin kesinleştirilmesi ve yedeklenmesi” sayfa 888](#)

Bu bilgilerde, bir iş biriminde ortaya çıkan kurtarılabilir alma ve koyma işlemlerinin nasıl kesinleştirileceği ve geri alınacağı açıklanır.

[“Starting IBM MQ applications using triggers” sayfa 899](#)

Tetikleyiciler kullanılarak IBM MQ uygulamalarının nasıl başlatılacağı ve tetikleyiciler hakkında bilgi edinin.

[“MQI ve kümelerle çalışma” sayfa 918](#)

Aramalara ilişkin özel seçenekler ve kümeleme ile ilgili dönüş kodları vardır.

[“Uygulamaların IBM MQ for z/OSüzerinde kullanılması ve yazılması” sayfa 922](#)

IBM MQ for z/OS uygulamaları, birçok farklı ortamda çalışan programlardan da yapılabilir. Bu, birden çok ortamda bulunan olanaklardan yararlanabilecekleri anlamına gelir.

### **IBM MQkullanarak IMS uygulamaları yazılıyor**

There are further considerations when using IBM MQ in IMS applications These include which MQ API calls can be used and the mechanism used for syncpoint.

IBM MQ for z/OSüzerinde IMS uygulamaları yazılmasına ilişkin ek bilgi için aşağıdaki bağlantıları kullanın:

- [“IMS uygulamalarında eşitleme noktaları” sayfa 66](#)
- [“IMS uygulamalarında MQI çağrıları” sayfa 67](#)

### **Kısıtlamalar**

IMS bağdaştırıcısı kullanılarak bir uygulama tarafından IBM MQ API çağrılarının kullanılabilmesi kısıtlamalar vardır.

Aşağıdaki IBM MQ API çağrıları, IMS bağdaştırıcısı kullanılarak bir uygulama içinde desteklenmez:

- MQCB
- MQCB\_FUNC
- MQCTL

### İlgili kavramlar

[“IMS köprü uygulamaları yazılıyor” sayfa 70](#)

Bu konuda, IBM MQ - IMS köprüsünü kullanmak için uygulamalar yazılmasıyla ilgili bilgiler bulunur.

#### *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) çağrısı kullanabilirsiniz. Daha fazla bilgi için IMS belgelerinde [ROLB call](#) başlıklı konuya bakın.

Kuyruk yöneticisi, iki aşamalı bir kesinleştirme protokolünün katılımcısıdır; eşgüdümçü 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 belirttik 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.

#### *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 934](#)
- [“Sorgu uygulamaları” sayfa 936](#)

## 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 aşılabılır:

- **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 yaklaşılabılır. MQI yanıt iletileri, işlenmekte olan MQI iletisinin yaratıcısının yetkisiyle birlikte konabilse de, IMS kaynak güncellemelerinin güvenlik etkilerinin dikkatli bir şekilde ele alınması gerekir.

- **MPP çağırısı 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 CMQTM2L.
*
* 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 kullanılmayacağından, 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 karmaşık 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
```

### ***IMS köprü uygulamaları yazılıyor***

Bu konuda, IBM MQ - IMS köprüsünü kullanmak için uygulamalar yazılmasıyla ilgili bilgiler bulunur.

IBM MQ - IMS köprüsü hakkında bilgi için bkz. [IMS köprüsü](#).

IBM MQ for z/OS üzerinde IMS köprüsü uygulamaları hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:

- [“How the IMS bridge deals with messages” sayfa 70](#)
- [“Writing IMS transaction programs through IBM MQ” sayfa 944](#)

### ***İlgili kavramlar***

[“IBM MQ kullanarak IMS uygulamaları yazılıyor” sayfa 66](#)

There are further considerations when using IBM MQ in IMS applications These include which MQ API calls can be used and the mechanism used for syncpoint.

#### *How the IMS bridge deals with messages*

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

You must also put your messages on IBM MQ queues that have been defined with a storage class that specifies the XCF group and member name of the target IMS system. Bunlar MQ-IMS köprü kuyrukları ya da basit **köprü** kuyrukları olarak bilinir.

IBM MQ-IMS köprüsü, QSGDISP (QMGR) ile tanımlandıysa ya da NOSHEARE seçeneğiyle birlikte QSGDISP (SHARED) ile tanımlandıysa, köprü kuyruğuna özel giriş erişimi (MQOO\_INPUT\_EXCLUSIVE) gerektirir.

Bir kullanıcının bir IMS uygulamasına ileti göndermeden önce IMS ' ta oturum açması gerekmez. Güvenlik denetimi için, MQMD yapısının *UserIdentifier* alanındaki kullanıcı kimliği kullanılır. Denetleme düzeyi, IBM MQ IMS'a bağlandığında belirlenir ve *IMS köprüsü için uygulama erişim denetimi*' ta açıklanmıştır. Bu, sözde oturum açmanın 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 ayraçlar, [] isteğe bağlı birden çok kesimi temsil eder.
  2. MQMD yapısının *Format* alanını MQIIH yapısını kullanmak için MQFMT\_IMS olarak ayarlayın.
- IMS hareket verilerini içeren iletiler, ancak MQIIH yapısı yok:

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

IBM MQ , LL byte toplamını artı MQIIH (varsa) uzunluğunun ileti uzunluğuna eşit olmasını sağlamak için ileti verilerini doğrular.

IBM MQ - IMS köprüsü, köprü kuyruklarından ileti aldığı anda, bunları aşağıdaki gibi işler.

- İleti bir MQIIH yapısı içeriyorsa, köprü MQIIH (bkz. MQIIH ), OTMA üstbilgilerini oluşturur ve iletiyi IMS' e gönderir. İşlem kodu, giriş iletilerinde belirtilir. Bu bir LTERM ise, IMS bir DFS1288E iletilisiyle yanıtlanır. Hareket kodu bir komutu gösteriyorsa, IMS komutu yürütür; tersi durumda ileti, hareket için IMS içinde kuyruğa alınır.
- İleti IMS hareket verisi içeriyorsa, ancak MQIIH yapısı yoksa, IMS köprüsü aşağıdaki varsayımları yapar:
  - İşlem kodu, kullanıcı verilerinin 5-12 arasındaki baytlardır.
  - Hareket, etkileşimli olmayan kipte
  - Hareket kesinleştirme kipinde 0 (kesinleştirme-sonra-gönder)
  - MQMD ' deki *Format* , *MFSMapName* (giriş sırasında) olarak kullanılır
  - Güvenlik kipi MQISS\_CHECK ' dir

The reply message is also built without an MQIIH structure, taking the *Format* for the MQMD from the *MFSMapName* of the IMS output.

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

- Commit kipi 0 (COMMIT\_THEN\_SEND) kullanan tüm iletiler için synchronized Tpipe kullanılır (bu program, IMS /DIS TMEMBER istemci TPIPE xxxx komutunun durum alanında SYN ile gösterilir)
- Kesinleştirme kipi 1 (SEND\_THEN\_COMMIT) kullanan tüm iletiler için uyumlulaştırılmamış bir Tpipe kullanılır

Tpipes, ilk kullanılanlarda IBM MQ tarafından oluşturulur. IMS yeniden başlatılıncaya kadar, uyumlulaştırılmamış bir Tpipe vardır. IMS soğuk çalışmaya başlayıncaya kadar, eşitlenmiş Tpipes var. Bu Tpipes 'i kendiniz silemezsiniz.

IBM MQ - IMS köprüsünün iletilerle nasıl ilgilenileceğini konu ile ilgili daha fazla bilgi için aşağıdaki konulara bakın:

- [“IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi” sayfa 71](#)
- [“İleti, IMS kuyruğuna konulamazsa” sayfa 72](#)
- [“IMS köprü geribildirim kodları” sayfa 72](#)
- [“IMS köprüsünden gelen iletilerde MQMD alanları” sayfa 72](#)
- [“IMS köprüsünden gelen iletilerde MQIIH alanları” sayfa 73](#)

- “IMS' dan gelen yanıt iletileri” sayfa 74
- “IMS hareketlerinde diğer yanıt PCB ' lerinin kullanılması” sayfa 75
- “İstenmeyen iletiler IMS' den gönderiliyor” sayfa 75
- “İleti bölümlenmesi” sayfa 75
- “Data conversion for messages to and from the IMS bridge” sayfa 75

### İlgili kavramlar

“Writing IMS transaction programs through IBM MQ” sayfa 944

The coding required to handle IMS transactions through IBM MQ depends on the message format required by the IMS transaction and the range of responses it can return. Ancak, uygulamanızın IMS ekran biçimlendirme bilgilerini işleyeceği zaman dikkate alınması gereken birkaç nokta vardır.

*IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesi*

IBM MQ iletilerinin IMS işlem tipleriyle eşlenmesini açıklayan bir tablo.

Çizelge 136. 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	<ul style="list-style-type: none"> <li>• Kurtarılabılır tam işlevli hareketler</li> <li>• Kurtarılamaz hareketler IMS tarafından reddedilir</li> </ul>	<ul style="list-style-type: none"> <li>• Fastpath işlemleri</li> <li>• Etkileşimli hareketler</li> <li>• Tam işlevli hareketler</li> </ul>
Kalıcı olmayan IBM MQ iletileri	<ul style="list-style-type: none"> <li>• Kurtarılamaz tam işlevli hareketler</li> <li>• IMS V8 ve APAR PQ61404 ve IMS ürününün sonraki tüm sürümleriyle kurtarılabılır hareketlere izin verilir.</li> </ul>	<ul style="list-style-type: none"> <li>• Fastpath işlemleri</li> <li>• Etkileşimli hareketler</li> <li>• Tam işlevli hareketler</li> </ul>

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

*İleti, IMS kuyruğuna konulamazsa*

İleti, IMS kuyruğuna konamazsa yapılacak işlemler hakkında bilgi edinin.

İleti, IMS kuyruğuna konulamazsa, IBM MQ tarafından aşağıdaki işlem alınır:

- If a message cannot be put to IMS because the message is invalid, the message is put to the dead-letter queue, and a message is sent to the system console.
- İleti geçerliyse, ancak IMStarafından reddedilirse, IBM MQ sistem konsoluna bir hata iletilisi gönderir, ileti IMS durum kodunu içerir ve IBM MQ iletilisi ölü-mektup kuyruğuna konmaktadır. IMS algılama kodu 001Aise, IMS yanıt kuyruğunda başarısızlığın nedenini içeren bir IBM MQ iletilisi gönderir.

**Not:** Daha önce listelenen durumlarda, IBM MQ herhangi bir nedenle iletiyi ölü-mektup kuyruğuna koyamıyorsa, ileti kaynak IBM MQ kuyruğuna geri döndürülür. Sistem konsoluna bir hata iletilisi gönderilir ve kuyruktan başka ileti gönderilmez.

İletileri yeniden göndermek için, aşağıdakileri yapın: **bir**

- Kuyruğa karşılık gelen IMS içindeki Tpipes ' ı durdurun ve yeniden başlatın.
- GET ' i (DEVRE dışı) ve yeniden GET (ETKİN) olarak değiştirin.
- IMS ya da OTMA ' yı durdurun ve yeniden başlatın.
- IBM MQ altsisteminizi durdurun ve yeniden başlatın.
- İleti, bir ileti hatasından başka bir şey için IMS tarafından reddedilirse, IBM MQ iletilisi kaynak kuyruğa geri döndürülür, IBM MQ kuyruğu işlemeyi durdurur ve sistem konsoluna bir hata iletilisi gönderilir.

Bir kural dışı durum raporu iletisi gerekliyse, köprü, iletiyi orijinalinin yetkisiyle yanıtlama kuyruğuna koyar. İleti kuyruğa konulamazsa, rapor iletisi, köprünün yetkisiyle birlikte, ölü-mektup kuyruğuna konmaya başlanır. DLQ ' ya (DLQ) yerleştirilemiyorsa, atılır.

#### *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 0x001A için 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](#).

#### *IMS köprüsünden gelen iletilerde MQMD alanları*

Learn about the MQMD fields in messages from the IMS bridge.

Kaynak iletinin MQMD ' si, OTMA üstbilgilerinin Kullanıcı Verileri kısmında IMS tarafından taşınır. İleti IMS içinde oluştuysa, bu, IMS Hedef Çözüm Çıkışı tarafından oluşturulur. IMS ' tan alınan bir iletinin MQMD 'si şu şekilde oluşturulmuştur:

#### **StrucID**

"MD"

#### **S\**

MQMD\_VERSION\_1

#### **Rapor**

MQRO\_NONE

#### **MsgType**

MQMT\_REPLY

#### **Son kullanma tarihi**

MQIIH ' ın İşaretler alanında MQIIH\_PASS\_EXPTH değeri ayarlandıysa, bu alan kalan süre bitimi değerini içerir, aksi takdirde MQE\_UNESNC olarak ayarlanır.

#### **Geribildirim**

MQFB\_YOK

#### **Kodlama**

MQENC.Native ( z/OS sisteminin kodlaması)

#### **CodedCharSetId**

MQCCSI\_Q\_MGR ( z/OS sisteminin CodedCharSetID değeri)

#### **Biçim**

MQMD.Format

#### **Öncelik**

MQMD.Priority

#### **Kalıcılık**

Depends on commit mode: MQMD.Persistence of the input message if CM-1; persistence matches recoverability of the IMS message if CM-0

#### **MsgId**

MQMD.MsgId , MQRO\_PASS\_MSG\_ID, tersi durumda Yeni MsgId (varsayılan)

**CorrelId**

MQMD.CorrelId from the input message if MQRO\_PASS\_CORREL\_ID, otherwise MQMD.MsgId from the input message (the default)

**BackoutCount**

0

**ReplyToQ**

Boşluklar

**ReplyToQMgr**

Boşluklar (MQPUT sırasında kuyruk yöneticisi tarafından yerel qmgr adı olarak ayarlanır)

**UserIdentifier**

MQMD.UserIdentifier

**AccountingToken**

MQMD.AccountingToken

**ApplIdentityVerileri**

MQMD.ApplIdentityData giriş iletisi

**PutApplTipi**

Hata yoksa MQAT\_XCF, tersi durumda MQAT\_BRIDGE

**PutApplAdı**

<XCFgroupName> <XCFmemberName> hata yoksa, QMGR adı

**PutDate**

İletinin konulduğu tarih

**PutTime**

İletinin konulduğu saat

**ApplOriginVerileri**

Boşluklar

*IMS köprüsünden gelen iletilerde MQIIH alanları*

Learn about the MQIIH fields in messages from the IMS bridge.

IMS ' tan alınan bir iletinin MQIIH değeri şu şekilde oluşturulmuştur:

**StrucId**

"IH"

**S\u00fcr\u00fcm**

1

**StrucLength**

84

**Kodlama**

MQENC\_NATIVE

**CodedCharSetId**

MQCCSI\_Q\_MGR

**Biçim**

MQIIH.ReplyToFormat boş değilse, giriş iletisininMQIIH.ReplyToFormat ' u, tersi durumda 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 konursa giriş iletisininMQIIH.Authenticator değeri, başka bir şekilde boşluk olur.

**TranInstanceTanıtıcısı**

Sohbetlerde, OTMA üstbilgisinden Sohbet Tanıtıcısı/Sunucu Belirteci. V14öncesindeki IMS sürümlerinde, bu alan her zaman, sohbette değilse boş değerlerde olur. IMS V14 ' ten itibaren bu alan, sohbet sırasında olmasa da IMS tarafından ayarlanabilirler.

**TranState**

"C" ise, iletişim halinde, aksi takdirde boşluk

**CommitMode**

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

**SecurityScope**

Boş

**Ayrıldı**

Boş

*IMS' dan gelen yanıt iletileri*

Bir IMS hareketi ISRTS 'yi IOPCB' ye gönderdiğinde, ileti kaynak LTTerm ya da TPIPE ' ye geri yönlendirilir.

Bunlar, IBM MQ içinde yanıt iletileri olarak görülür. IMS ' dan gelen yanıt iletileri, özgün iletide belirtilen yanıtlama kuyruğuna konilir. İleti, yanıt kuyruğuna konamazsa, köprünün yetkisini kullanarak, ölü-mektup kuyruğuna konabilecektir. İleti, ölü-mektup kuyruğuna yerleştirilemiyorsa, iletinin alınmadığını belirten bir eksi alındı bildirimini IMS ' a gönderilir. İletinin sorumluluğu daha sonra IMS' a geri gönderilir.

Kesinleştirme kipi 0 kullanıyorsanız, bu Tpipe 'den gelen iletiler köprüye gönderilmez ve IMS kuyruğunda kalır; başka bir ileti yoksa, yeniden başlatma işlemi başlatılıncaya kadar başka ileti gönderilmez.

Kesinleştirme kipi 1 kullanıyorsanız, diğer işler devam edebilir.

Yanıta bir MQIIH yapısı varsa, biçimi MQFMT\_IMS; değilse, biçim tipi, ileti eklenirken kullanılan IMS Değişiklik Yönetimi adı ile belirtilir.

*IMS hareketlerinde diğer yanıt PCB ' lerinin kullanılması*

Bir IMS hareketi diğer PCB 'lerini (ALTPCB' ye ISRTS ya da değiştirilebilir bir PCB ' ye bir CHNG çağrısı) kullandığında, iletinin yeniden yönlendirilmesi gerekip gerekmediğini belirlemek için ön yöneltme çıkışı (DFSYPX0) çağrılır.

İleti yeniden yönlendirilecekse, hedef çözüm çıkışı (DFSYDRU0) hedefi doğrulamak ve üstbilgi bilgilerini hazırlamak için çağrılır ve bu çıkış programlarıyla ilgili bilgi için [Using OTMA exits in IMS](#) ve [Ön yönlendirme çıkışı DFSYPX0](#) başlıklı konuya bakın.

Çıkışlarda işlem yoksa, bir IBM MQ kuyruk yöneticisinden başlatılan IMS hareketlerinden, IOPCB 'den ya da ALTPCB' den başlatılan tüm çıkış, aynı kuyruk yöneticisine döndürülür.

*İstenmeyen iletiler IMS' den gönderiliyor*

IMS 'dan bir IBM MQ kuyruğuna ileti göndermek için, bir ALTPCB' ye ISRTS ' den bir IMS işlemi başlatmanız gerekir.

You need to write pre-routing and destination resolution exits to route unsolicited messages from IMS and build the OTMA user data, so that the MQMD of the message can be built correctly. Bu çıkış programlarıyla ilgili bilgi için [Ön yönlendirme çıkışı DFSYPX0](#) ve [Hedef çözme kullanıcı çıkışı](#) başlıklı konuya bakın.

**Not:** IBM MQ - IMS köprüsü, aldığı iletinin yanıt olup olmadığını ya da istenmemiş bir ileti olup olmadığını bilmiyor. İletiyi aynı şekilde, iletinin MQMD ve MQIIH 'lerini oluştururken, iletiyle birlikte gönderilen OTMA UserData ' ne dayalı olarak bu iletiyi işler.

İstenmeyen iletiler yeni TPipes yaratabilir. Örneğin, var olan bir IMS işlemi yeni bir LTTerm ' ye (örneğin PRINT01gibi) geçtiyse, ancak uygulama çıkışın OTMA aracılığıyla sağlanmasını gerektiriyorsa, bu örnekte yeni bir Tpipe (bu örnekte PRINT01 adı verilir) yaratılır. Varsayılan olarak bu, uyumlulaştırılmamış bir Tpipe 'dir. Uygulama iletinin kurtarılabilir olmasını gerektiriyorsa, hedef çözme çıkış çıkış işaretini ayarlayın. Ek bilgi için *IMS Customization Guide* belgesine bakın.

### *İleti bölümlenmesi*

IMS işlemlerini tek ya da çok bölümlü giriş olarak tanımlayabilirsiniz.

Kaynak IBM MQ uygulaması, bir ya da daha fazla LLZZ-veri bölümü olarak MQIIH yapısının ardından kullanıcı girişini oluşturmaktadır. Bir IMS iletinin tüm kesimleri, tek bir MQPUT ile gönderilen tek bir IBM MQ iletilinde yer almalıdır.

LLZZ veri kesiminin uzunluk üst sınırı, IMS/OTMA (32767 bayt) tarafından tanımlanır. Toplam IBM MQ ileti uzunluğu, LL baytlarının toplamını artı MQIIH yapısının uzunluğunu içerir.

Yanıtın tüm bölümleri tek bir IBM MQ iletilinde yer alır.

MQFMT\_IMS\_VAR\_STRING biçimine sahip iletilerde 32 KB sınırlamasına ilişkin daha fazla kısıtlama var. Bir ASCII karma CCSID iletilinde bulunan veriler, EBCDIC karma CCSID iletiline dönüştürüldüğünde, SBCS ile DBCS karakterleri arasında geçiş her olduğunda bir çift bayt dizilimi başlangıç baytı ya da bir çift bayt dizilimi başlangıç baytı eklenir. 32 KB sınırlaması, iletinin büyüklük üst sınırı için geçerlidir. Bunun nedeni, iletteki LL alanı 32 KB 'yi geçemediğinden, iletinin tüm üst karakter ve başlangıç karakterleri de içinde olmak üzere 32 KB' yi aşmaması gerekir. İletiyi oluşturma işlemi bunun için izin vermelidir.

### *Data conversion for messages to and from the IMS bridge*

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

Gerekli tüm çıkışların, CSQXLIB DD bildiriminde gönderme yapılan veri kümesindeki dağıtılmış kuyruğa alma olanağı için kullanılabilir olması gerekir. Bu, herhangi bir IBM MQ platformundan IBM MQ - IMS köprüsünü kullanarak bir IMS uygulamasına ileti gönderebileceğiniz anlamına gelir.

Dönüştürme hataları varsa, ileti dönüştürülmeden kuyruğa alınır; bu sonuç, köprü'nün üstbilgi biçimini tanıyamadığı için IBM MQ - IMS köprüsü tarafından bir hata olarak ele alınır. Bir dönüştürme hatası ortaya çıkarsa, z/OS konsoluna bir hata ileti gönderilir.

Genel olarak veri dönüştürme hakkında ayrıntılı bilgi için bkz. "[Veri dönüştürme çıktıları yazılıyor](#)" sayfa 1014 .

## **Sending messages to the IBM MQ - IMS bridge**

Dönüştürmenin doğru bir şekilde gerçekleştirildiğinden emin olmak için, kuyruk yöneticisine iletinin biçiminin ne olduğunu söylemelisiniz.

İletin bir MQIIH yapısı varsa, MQMD ' deki *Format* değeri yerleşik MQFMT\_IMS biçimine ayarlanmalıdır ve MQIIH içindeki *Format* , ileti verilerinizi tanımlayan biçimdeki ada ayarlanmalıdır. MQIIH yoksa, MQMD 'de *Format* ' i biçim adınıza ayarlayın.

Verileriniz (LLZZ'lerden başka), tüm karakter verileri (MQCHAR) ise, biçim adı olarak (MQIIH ya da MQMD ' de uygun olarak), MQFMT\_IMS\_VAR\_STRING yerleşik biçimi olarak kullanın. Ters durumda, kendi biçim adınızı kullanın; bu durumda, biçiminiz için bir veri dönüştürme çıkışı da sağlamanız gerekir. Çıkışta, verilerin kendisinin yanı sıra, iletinizdeki LLZZs 'lerin dönüştürülmesini de işlemesi gerekir (ancak iletinin başlangıcındaki herhangi bir MQIH' yi işlemek zorunda değildir).

Uygulamanız *MFSMapName* kullanıyorsa, bunun yerine MQFMT\_IMS ile iletileri kullanabilir ve MQIIH ' nin *MFSMapName* alanında IMS işlemine geçirilen işlem adını tanımlayabilirsiniz.

## **Receiving messages from the IBM MQ - IMS bridge**

If an MQIIH structure is present on the original message that you are sending to IMS, one is also present on the reply message.

Yanıtınızın doğru şekilde dönüştürülmesini sağlamak için:

- Özgün iletinizde bir MQIIH yapınız varsa, özgün iletinin MQIIH *ReplytoFormat* alanında yanıt iletiniz için istediğiniz biçimi belirtin. Bu değer, yanıt iletinin MQIIH *Format* alanına yerleştirilir. Bu, özellikle tüm çıkış verileriniz LLZZ < karakter verileri > biçiminde olduğunda yararlı olur.

- Özgün iletinizde MQIIH yapısına sahip değilseniz, yanıt iletisi için IMS uygulamasının ISRT ' deki MFS MOD adı olarak IOPCB olarak yanıt iletisi olmasını istediğiniz biçimi belirtin.

#### *Writing IMS transaction programs through IBM MQ*

The coding required to handle IMS transactions through IBM MQ depends on the message format required by the IMS transaction and the range of responses it can return. Ancak, uygulamanızın IMS ekran biçimlendirme bilgilerini işleyeceği zaman dikkate alınması gereken birkaç nokta vardır.

Bir 3270 ekranından bir IMS işlemi başlatıldığında, ileti IMS Message Format Services aracılığıyla geçer. Bu işlem, hareket tarafından görülen veri akışından tüm uçbirim bağımlılığını kaldırabilir. OTMA aracılığıyla bir işlem başlatıldığında MFS ' ye dahil değildir. Uygulama mantığı MFS ' de uygulandıysa, bu yeni uygulamada yeniden yaratılmalıdır.

Bazı IMS hareketlerinde, son kullanıcı uygulaması belirli 3270 ekran davranışını değiştirebilir; örneğin, geçersiz veri girilebilen bir alanı vurgular. Bu bilgi tipi, program tarafından değiştirilmesi gereken her ekran alanı için IMS iletisine iki byte 'lık bir öznitelik alanı eklenerek iletilir.

Bu nedenle, bir 3270 'i taklit etmek için bir uygulama kodluyorsanız, ileti oluştururken ya da alırken bu alanların hesabını da almanız gerekir.

İşlemek için programınızdaki bilgileri kodlamak zorunda kalabilirsiniz:

- Hangi tuşa basılır (örneğin, Enter ve PF1)
- İletiniz iletinize iletildiğinde imlecin bulunduğu yer
- Öznitelik alanlarının IMS uygulaması tarafından ayarlanıp ayarlanmadığını
  - Yüksek, olağan ya da sıfır yoğunluğu
  - Renk
  - IMS , Enter tuşuna basıldığında alanın geri gelmesini bekleyip beklemeyeceğini
- IMS uygulamasının herhangi bir alanda boş değerli karakterler (X'3F') kullandıysa da.

IMS iletiniz yalnızca karakter verileri içeriyorsa (LLZZ-data kesiminden ayrı) ve bir MQIIH yapısı kullanıyorsanız, MQMD biçimini MQFMT\_IMS olarak ve MQIIH biçimini MQFMT\_IMS\_VAR\_STRING ' ye ayarlayın.

If your IMS message contains only character data (apart from the LLZZ-data segment), and you are **değil** using an MQIIH structure, set the MQMD format to MQFMT\_IMS\_VAR\_STRING and ensure that your IMS application specifies MODname MQFMT\_IMS\_VAR\_STRING when replying. Bir sorun ortaya çıkarsa (örneğin, işlemi kullanma yetkisi olmayan kullanıcı) ve IMS bir hata iletisi gönderirse, bu, DFSMOx biçiminde bir MODname (burada x, 1-5 aralığında bir sayıdır) biçiminde bir değişiklik iletisi gönderir. Bu, MQMD.Format.

IMS iletiniz ikili, paketlenmiş ya da kayan noktalı veri içeriyorsa (LLZZ-veri kesiminden ayrı olarak), kendi veri dönüştürme yordamlarınızı kodlayın. IMS ekran biçimlendirmesiyle ilgili bilgi edinmek için *IMS/ESA Application Programming: Transaction Manager* dosyasına bakın.

Consider the following topics when writing code to handle IMS transactions through IBM MQ.

- [“IMS etkileşimli işlemlerini başlatmak için IBM MQ uygulamaları yazılıyor” sayfa 944](#)
- [“IMS komutlarını içeren yazma programları” sayfa 945](#)
- [“Tetikleme” sayfa 945](#)

### **IMS etkileşimli işlemlerini başlatmak için IBM MQ uygulamaları yazılıyor**

IMS etkileşimli işlemlerini başlatmak için IBM MQ uygulaması yazılırken dikkat edilmesi gereken noktalar için bu bilgileri kılavuz olarak kullanın.

Bir IMS etkileşmesini çağıran bir uygulama yazdığınızda, aşağıdakileri göz önünde bulundurun:

- Uygulamanızın iletisiyle birlikte bir MQIIH yapısı ekleyin.
- Set the *CommitMode* in MQIIH to MQICM\_SEND\_THEN\_COMMIT.



- Yeni bir etkileşimi başlatmak için, MQIIH 'de *TranState* ' i MQITS\_NOT\_IN\_CONVERSACE olarak ayarlayın.
- Bir sohbetin ikinci ve sonraki adımlarını başlatmak için *TranState* , MQITS\_IN\_CONVERT olarak ayarlayın ve *TranInstanceId* değerini, görüşmenin önceki adımında döndürülen alanın değerine ayarlayın.
- There is no easy way in IMS to find the value of a *TranInstanceId*, should you lose the original message sent from IMS.
- The application must check the *TranState* of messages from IMS to check whether the IMS transaction has terminated the conversation.
- Bir etkileşimi sona erdirmek için /EXIT ' yi kullanabilirsiniz. *TranInstanceId*'i de tırnak içine almalısınız, *TranState* ' ı MQITS\_IN\_CONVERSAYA olarak ayarlayın ve sohbetin gerçekleştirilmekte olduğu IBM MQ kuyruğunu kullanın.
- Bir konuşmayı tutmak ya da serbest bırakmak için /HOLD ya da /REL kullanamazsınız.
- IBM MQ - IMS köprüsünde çağrılan sohbetler, IMS yeniden başlatılırsa sonlandırılır.

## IMS komutlarını içeren yazma programları

Bir uygulama programı, bir işlem yerine, LLZZ*komut* biçiminde bir IBM MQ iletisi oluşturabilir; burada *komut* , /DIS TRAN PART ya da /DIS POOL ALL biçimidir.

Most IMS commands can be issued in this way; see *IMS V11 İletişim ve Bağlantılar* for details. Komut çıkışı, görüntü için 3270 uçbirimine gönderileceği şekilde, metin biçimindeki IBM MQ yanıt iletisinde alınır.

OTMA, çıktının bir mimarisini döndüren IMS görüntü hareketi komutunun özel bir formunu uygulamıştır. Tam biçim, *IMS V11 Communications ve Connections*' da tanımlanır. Bu formu bir IBM MQ iletisinden başlatmak için, ileti verilerini önceki gibi oluşturun, örneğin /DIS TRAN PART ve MQIIH ' daki TranState alanını MQITS\_ARCHITECTED olarak ayarlayın. IMS komutu işler ve yanıtı, archipted formunun içinde döndürür. Bir mimari yanıt, çıkışın metin biçiminde bulunabilecek tüm bilgileri ve bir ek bilgi parçasını içerir: hareketin kurtarılabilir ya da kurtarılamaz olarak tanımlanıp tanımlanmadığı.

## Tetikleme

IBM MQ - IMS köprüsü, tetikleme iletilerini desteklemez.

XCF parametreleriyle bir depolama sınıfı kullanan bir başlatma kuyruğu tanımlıyorsanız, bu kuyruğa gönderilen iletiler, köprüye geçildiğinde reddedilir.

## İstemci yordamsal uygulamaları yazılıyor

Bir yordamsal dil kullanarak IBM MQ üzerinde istemci uygulamaları yazmak için bilmeniz gerekenler.

Uygulamalar IBM MQ istemci ortamında oluşturulabilir ve çalıştırılabilir. Uygulama oluşturulmalı ve kullanılan IBM MQ MQI client ile bağlantılandırılmalıdır. Uygulamaların oluşturulacağı ve bağlantılı olduğu yol, kullanılan platforma ve programlama diline göre değişiklik gösterir. İstemci uygulamalarının nasıl oluşturulabilmesiyle ilgili bilgi için bkz. [“IBM MQ MQI clients için uygulama oluşturma” sayfa 951.](#)

Belirli koşulların karşılanabilmesi koşuluyla, bir IBM MQ uygulamasını hem tam bir IBM MQ ortamında hem de bir IBM MQ MQI client ortamında kodunuzu değiştirmeden çalıştırabilirsiniz. Uygulamalarınızı IBM MQ istemci ortamında çalıştırılmasına ilişkin daha fazla bilgi için bkz. [“IBM MQ MQI client ortamında uygulamaları çalıştırma” sayfa 953.](#)

IBM MQ MQI client ortamında çalıştırılacak uygulamaları yazmak için ileti kuyruğu arabirimini (MQI) kullanırsanız, IBM MQ uygulamasının işlenmesinin kesintiye uğramamasını sağlamak için bir MQI çağrısı sırasında bazı ek denetimlerin uygulanması gerekir. Bu denetimlerle ilgili daha fazla bilgi için bkz. [“İstemci uygulamasında MQI ' nin kullanılması” sayfa 946.](#)

İstemci uygulamaları olarak diğer uygulama tiplerini hazırlama ve çalıştırma bilgileri için aşağıdaki konulara bakın:

- [“CICS ve Tuxedo uygulamalarının hazırlanması ve çalıştırılması” sayfa 965](#)

- [“Microsoft Transaction Server uygulamalarının hazırlanması ve çalıştırılması” sayfa 46](#)
- [“IBM MQ JMS uygulamalarının hazırlanması ve çalıştırılması” sayfa 968](#)

### **İlgili kavramlar**

[“Uygulama geliştirme kavramları” sayfa 7](#)

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.

[“IBM MQ için uygulama geliştirilmesi” sayfa 5](#)

İletileri göndermek ve almak, kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için uygulamalar geliştirebilirsiniz. IBM MQ , birçok farklı dil ve çerçeveye içinde yazılmış uygulamaları destekler.

[“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#)

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğinize karar verdiğinizde, IBM MQ tarafından sunulan özelliklerin nasıl kullanılacağına karar vermeniz gerekir.

[“Kuyruğa alma için bir yordamsal uygulama yazma” sayfa 759](#)

Kuyruk yöneticisi, yayınlama/abone olma, nesnelere açma ve kapama nesnelere bağlanma ve bağlantıyı kesme, kuyruğa alma ve bağlantı kesme hakkında bilgi edinmek için bu bilgileri kullanın.

[“Yayınlama/abone olma uygulamaları yazılıyor” sayfa 845](#)

Yayınlama/abone olma IBM MQ uygulamalarını yazmaya başlayın.

[“Yordamsal uygulama oluşturulması” sayfa 1030](#)

Bir IBM MQ uygulamasını birden çok yordamsal dilden birine yazabilir ve uygulamayı farklı platformlarda çalıştırabilirsiniz.

[“Yordamsal program hatalarının işlenmesi” sayfa 1073](#)

Bu bilgiler, bir çağrı yaparken ya da ileti son hedefine teslim edildiğinde, uygulama MQI çağrılarınızla ilişkili hataları açıklar.

### **İlgili görevler**

[“IBM MQ örnek yordamsal programlarının kullanılması” sayfa 1092](#)

Bu örnek programlar yordamsal dillere yazılır ve İletim Kuyruğu Arabirimi 'nin (MQI) tipik kullanımları gösterir. Farklı platformlardaki IBM MQ programları.

## **İstemci uygulamasında MQI ' nin kullanılması**

Bu konular grubu, IBM MQ uygulamanızın bir ileti kuyruğu arabirimi (MQI) istemci ortamında çalışması ve tam IBM MQ kuyruk yöneticisi ortamında çalıştırılması arasındaki farkları göz önünde bulundurur.

Bir uygulamayı tasarlarırken, IBM MQ uygulamasının işlenmesinin kesintiye uğramadığından emin olmak için, bir MQI çağrısı sırasında hangi denetimlerin uygulamanız gerektiğini dikkate alın.

MQI kullanan uygulamaları çalıştırabilmeniz için bazı IBM MQ nesnelere oluşturmanız gerekir. Daha fazla bilgi için [MQI kullanan uygulama programları](#) başlıklı konuya bakın.

### **İstemci uygulamasındaki bir iletin boyutunu sınırlandırma**

Kuyruk yöneticisinin ileti uzunluğu üst sınırı, ancak bir istemci uygulamasından aktarabileceğiniz ileti büyüklüğü üst sınırı kanal tanımlamasıyla sınırlıdır.

Kuyruk yöneticisinin ileti uzunluğu üst sınırı (MaxMsguzunluğu) özneliği, o kuyruk yöneticisi tarafından işlenebilecek bir iletin uzunluk üst sınırıdır.

**Multi** Multiplatforms' ta, bir kuyruk yöneticisinin ileti uzunluğu üst sınırı özneliğini artırabilirsiniz. Ek bilgi için [ALTER QMGR](#) başlıklı konuya bakın.

MQINQ çağrısını kullanarak, kuyruk yöneticisi için MaxMsgLength değerini öğrenebilirsiniz.

MaxMsgLength özneliği değiştirilirse, kuyruklar ve hatta iletiler, yeni değerden daha büyük bir uzunlukla birlikte, önceden kuyruklar ve hatta ileti olmadığı için denetim yapılmadan yapılır. Bu özneliği değiştirdikten sonra, değişikliğin yürürlüğe girdiğinden emin olmak için uygulamaları ve kanalları yeniden

başlatın. Daha sonra kuyruk yöneticisinin ya da kuyruğun MaxMsguzunluğunu aşan yeni iletiler üretilemez (kuyruk yöneticisi bölümlenmesine izin verilmediği sürece).

Bir kanal tanımlamasındaki ileti uzunluğu üst sınırı, istemci bağlantısıyla iletebileceğiniz bir iletinin büyüklüğünü sınırlar. Bir IBM MQ uygulaması, MQPUT çağrısını ya da MQGET çağrısını bu iletiden daha büyük bir iletiyle kullanmaya çalışırsa, uygulamaya bir hata kodu döndürülür. Kanal tanımlamasının ileti büyüklüğü üst sınırı değiştirgesi, istemci bağlantısı üzerinden MQCB kullanılarak tüketilebilecek ileti büyüklüğü üst sınırını etkilemez.

### İlgili kavramlar

“MQCONN olanağının kullanılması” sayfa 950

MQCNO yapısındaki bir kanal tanımlaması (MQCD) yapısı belirtmek için MQCONN çağrısını kullanabilirsiniz.

### İlgili başvurular

İleti uzunluğu üst sınırı (MAXMSGL)

KANALI ALTER

2010 (07DA) (RC2010): MQRC\_DATA\_LENGTH\_ERROR

### İstemci ya da sunucu CCSID 'si seçilmesi

İstemciye ilişkin yerel kodlanmış karakter takımı tanıtıcısını (CCSID) kullanın. Kuyruk yöneticisi gerekli dönüşümü gerçekleştirir. CCSID 'yi geçersiz kılmak için MQCCSID ortam değişkenini kullanın. Uygulamanız birden çok PUT gerçekleştiriyorsa, ilk PUT işlemi tamamlandıktan sonra, MQMD 'nin CCSID ve kodlama alanlarının üzerine yazılabilir.

İleti kuyruğu arabiriminde (MQI), istemciden istemci sınırlı kod öbeğine aktarılan veriler, yerel CCSID 'de (IBM MQ MQI client) için kodlanmalıdır. Bağlı kuyruk yöneticisi verilerin dönüştürülmesini gerektiriyorsa, dönüştürme işlemi kuyruk yöneticisinden istemci destek kodu tarafından gerçekleştirilir.

In IBM WebSphere MQ 7.0 and later versions, the Java client can do the conversion if the queue manager is unable to do so. Bkz. “IBM MQ classes for Java istemci bağlantıları” sayfa 345.

İstemci kodu, istemcideki MQI 'yi geçen karakter verilerinin, o iş istasyonu için yapılandırılmış CCSID' de olduğunu varsayar. Bu CCSID desteklenmeyen bir CCSID ise ya da gereken CCSID değilse, bu komutlardan birini kullanarak MQCCSID ortam değişkeniyle geçersiz kılınabilir:

#### Windows

```
SET MQCCSID=850
```

#### UNIX

```
export MQCCSID=850
```

#### IBM i

```
ADDENVVAR ENVVAR(MQCCSID) VALUE(37)
```

Tanıtimda bu parametre belirlendiyse, tüm MQI verilerinin 850 kod sayfasında olduğu varsayılır.

**Not:** 850 kod sayfasıyla ilgili varsayım, iletteki 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.

After the first PUT, these fields contain the value used by the connected queue manager to convert the IBM MQ headers. Uygulamanızın, değerleri gerektirdiği değerlere sıfırladığından emin olun.

## ***İstemci uygulamasında MQINQ kullanılması***

MQINQ kullanılarak sorgulanan bazı değerler istemci kodu tarafından değiştirilir.

### **CCSID**

istemci CCSID değerine ayarlıdır, kuyruk yöneticisinden değil.

### **MaxMsgUzunluğu**

kanal tanımlamasıyla sınırlandırılırsa azaltılır. Bu işlem aşağıdaki gibi olacaktır:

- Kuyruk tanımlamasında tanımlanan değer ya da
- Kanal tanımlamasında tanımlanan değer

Ek bilgi için [MQINQ](#) başlıklı konuya bakın.

## ***İstemci uygulamasında eşitleme noktası eşgüdümü kullanma***

Temel istemci çalışan bir uygulama MQCMIT ve MQBACK yayınlayabilir; ancak, Sync Point denetiminin kapsamı MQI kaynaklarıyla sınırlıdır. Genişletilmiş işlemsel istemciyle bir dış hareket yöneticisi kullanabilirsiniz.

IBM MQ içinde, kuyruk yöneticisinin rollerinden biri, bir uygulama içindeki tutarlılık noktası denetmenidir. Bir uygulama bir IBM MQ temel istemcisinde çalışıyorsa, bu uygulama MQCMIT ve MQBACK yayınlayabilir, ancak Sync Point denetiminin kapsamı MQI kaynaklarıyla sınırlıdır. The IBM MQ verb MQBEGIN is not valid in a base client environment.

Sunucuda tam kuyruk yöneticisi ortamında çalışan uygulamalar, bir hareket izleme programı aracılığıyla birden çok kaynağı (örneğin veritabanları) koordine edebilir. Sunucuda IBM MQ ürünleriyle birlikte verilen İşlem İzleyicisini ya da CICS gibi başka bir hareket izleyicisini kullanabilirsiniz. Bir işlem izleyiciyi temel istemci uygulaması ile kullanamazsınız.

IBM MQ genişletilmiş işlemsel istemciyle bir dış hareket yöneticisi kullanabilirsiniz. Bkz. [Genişletilmiş işlemsel istemci nedir?](#) ayrıntılı bilgi için.

## ***İstemci uygulamasında okuma yazma işlevini kullanma***

İstemci uygulaması, iletileri istemek zorunda kalmadan bir istemciye kalıcı olmayan iletilerin gönderilmesini sağlamak için bir istemcide önceden okuma olanağını kullanabilirsiniz.

İstemci bir sunucudan ileti gerektirdiğinde, sunucuya bir istek gönderir. Tükettiği iletilerin her biri için ayrı bir istek gönderir. Bir istemcinin, bu istek iletilerini göndermekten kaçınarak kalıcı olmayan iletileri tüketmesini artırmak için, ileride okuma kullanacak şekilde yapılandırılmış bir istemci olabilir. Önden okuma, bir uygulamanın istekte bulunmadan istemciye gönderilmesine izin verir.

İleriye okumanın kullanılması, bir istemci uygulamasından kalıcı olmayan iletiler tüketirken performansı yükseltebilirler. Bu performans iyileştirmesi, hem MQI hem de JMS uygulamaları için kullanılabilir. MQGET ya da zamanuyumsuz tüketim kullanan istemci uygulamaları, kalıcı olmayan iletiler tüketildiğinde performans iyileştirmelerinden yararlanırlar.

MQOPEN ' i MQOO\_READ\_ÖNCEDEN ile çağırdığınızda, IBM MQ istemcisi belirli koşullar karşılandığında yalnızca önden okumayı etkinleştirir. Bu koşullar şunlardır:

- Hem istemci, hem de uzak kuyruk yöneticisi IBM WebSphere MQ 7.0 ya da sonraki bir yerde olmalıdır.
- 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.

Okuma seçeneği geçerli kılındığında, iletiler ileriye okuma arabelleği adı verilen istemcide bir bellek arabelleğiyle gönderilir. İstemcinin okuma yazma arabelleğiyle açık bir okuma arabelleği vardır; bu arabelleği, önceden okuma özelliği etkinleştirilmiş olarak açar. İleriye okuma arabelleğindeki iletiler kalıcı olarak saklanmaz. İstemci düzenli olarak sunucuyu, tükettiği veri miktarına ilişkin bilgilerle güncelleştirir.

Tüm seçenekler kullanılmak üzere desteklenmediği için, tüm istemci uygulaması tasarımları önden okuma özelliğini kullanmaya uygun değildir. Önceden okuma etkinleştirildiğinde, MQGET çağrıları arasında tutarlı olması için bazı seçeneklerin tutarlı olması gerekir. Bir istemci, seçim ölçütünü MQGET çağrıları arasında

değiştirirse, ileriye okuma arabelleğindeki saklanan iletiler, istemci okuma yazma arabelleğindeki iplikçik olarak kalır. Daha fazla bilgi için bkz. [“Kalıcı olmayan iletilerin performansını artırma” sayfa 823](#)

Önden okuma yapılandırması, IBM MQ istemcisi yapılandırma dosyasının MessageBuffer kısmında belirtilen üç öznitelik, MaximumSize, PurgeTime ve UpdatePercentage öznitelikleriyle denetlenir.

### ***İstemci uygulamasına zamanuyumsuz konması kullanma***

Zamanuyumsuz put kullanılması, bir uygulamanın kuyruk yöneticisinden yanıt beklemeden kuyruğa ileti yerleştirmesini sağlar. Bazı durumlarda ileti alışverişi başarımını artırmak için bunu kullanabilirsiniz.

Olağan durumda, bir uygulama bir iletiyi ya da iletileri bir kuyruğa koyduğunda, MQPUT ya da MQPUT1 kullanılarak, uygulamanın kuyruk yöneticisinin MQI isteğini işlediğini doğrulamasını beklemek gerekir. Özellikle istemci bağ tanımlarını kullanan uygulamalar için ileti alışverişi başarımını artırabilir ve iletileri zamanuyumsuz olarak koymak yerine, büyük sayıda küçük iletileri bir kuyruğa yerleştiren uygulamalar için kullanabilirsiniz. Bir uygulama bir iletiyi zamanuyumsuz olarak yerleştirdiğinde, kuyruk yöneticisi her çağrımın başarısını ya da hatasını döndürmez, ancak bunun yerine düzenli aralıklarla hata denetimi yapabilirsiniz.

Bir iletiyi zamanuyumsuz olarak bir kuyruğa koymak için, MQPMO yapısının *Options* alanında MQPMO\_ASYNC\_RESPONSE seçeneğini kullanın.

Bir ileti zamanuyumsuz koyma için uygun değilse, kuyruğa zamanuyumlu bir şekilde konmaya başlanır.

MQPUT ya da MQPUT1 için zamanuyumsuz koyma yanıtı istenirken, bir CompCode ve MQCC\_OK ve MQRC\_NONE iletilerinin nedeni, iletinin bir kuyruğa başarıyla konulduğu anlamına gelmeyebilir. Her bir MQPUT ya da MQPUT1 çağrısının başarısı ya da başarısızlığı hemen döndürülmesi de, zamanuyumsuz bir çağrı altında oluşan ilk hata, daha sonra MQSTAT çağrısı yoluyla saptlanabilir.

MQPMO\_ASYNC\_RESPONSE ile ilgili ayrıntılar için [MQPMO seçenekler](#) başlıklı konuya bakın.

Zamanuyumsuz Koyma örnek programı, kullanılabilir bazı özellikleri gösterir. Programın özelliklerinin ve tasarımının ve nasıl çalıştırılacağı konusunda ayrıntılı bilgi için bkz. [“Zamanuyumsuz Koy örnek programı” sayfa 1110](#).

### ***Bir istemci uygulamasında sohbetlerin paylaşılmasını kullanma***

Sohbet paylaşımına izin verildiği bir ortamda, sohbetler bir MQI kanalı eşgörünümünü paylaşabilir.

Her ikisi de kanal tanımlama (MQCD) yapısının bir parçası olan SharingConversations adlı iki alan tarafından yapılan paylaşımında paylaşım, kanal çıkış parametresinin (MQCXP) bir parçası olan iki alan tarafından kontrol edilir. MQCD ' deki SharingConversations (SharingConversations) alanı, kanalla ilişkilendirilmiş bir kanal yönetim ortamını paylaşabilecek etkileşim sayısı üst sınırını belirleyen bir tamsayı değeridir. MQCXP ' deki SharingConversations (SharingConversations) alanı, kanal örneğinin paylaşılıp paylaşılmadığını belirten bir Boole değeridir.

Paylaşımın paylaşılmasına izin verilmediği bir ortamda, aynı MQCD ' leri belirten yeni istemci bağlantıları bir kanal yönetim ortamını paylaşmaz.

Aşağıdaki koşullar doğru olduğunda, kanal yönetim ortamını yeni bir istemci uygulaması bağlantısı paylaşacak:

- Kanal yönetim ortamının istemci bağlantısı ve sunucu bağlantısı uçları, sohbetleri paylaşmak üzere yapılandırıldı ve bu değerler kanal çıkışlar tarafından geçersiz kılınmadı.
- İstemci bağlantısı MQCD değeri (istemcide MQCONNX çağrısında ya da istemci kanal tanımlama çizelgesinden (CCDT) sağlanır), varolan kanal yönetim ortamı ilk kez kurulduğunda, istemcide sağlanan MQCD çağrısına ya da CCDT ' den sağlanan istemci bağlantısı MQCD değeriyle tam olarak eşleşir. Özgün MQCD ' nin daha sonra çıkışlar ya da kanal anlaşması yoluyla değiştirilebileceğini, ancak bu değişikliklerin yapılmadan önce istemci sistemine sağlanan değerle eşleşmenin yapıldığını unutmayın.
- Sunucu tarafındaki paylaşım etkileşimleri sınırı aşılmaz.

Yeni bir istemci uygulaması bağlantısı, bir kanal örneğini diğer etkileşimler ile paylaşarak çalışma ölçütleriyle eşleşirse, bu karar o konuşmada herhangi bir çıkışa çağrılmadan önce yapılır. Böyle bir etkileşimde bulunan çıkışlar, kanal örneğini diğer etkileşimler ile paylaşıp paylaşmadığını değiştiremez.

Yeni kanal tanımlamasıyla eşleşen var olan kanal yönetim ortamı yoksa, yeni bir kanal yönetim ortamı bağlanır.

Kanal anlaşması, yalnızca kanal yönetim ortamındaki ilk etkileşim için gerçekleşir; kanal örneğine ilişkin anlaşmalı değerler o aşamada sabitlenir ve sonraki etkileşimler başlatıldığında değiştirilemez. TLS kimlik doğrulaması da yalnızca ilk etkileşim için gerçekleşir.

MQCD SharingConversations değeri, istemci bağlantısında ya da kanal yönetim ortamının sunucu bağlantısı sonundaki yuvadaki ilk etkileşim için tüm güvenlik, gönderme ya da alma çıkışları sırasında değiştirilirse, tüm bu çıkışlardan sonra sahip olduğu yeni değer, kanal örneğine ilişkin paylaşma etkileşimleri değerini belirlemek için kullanılır (en düşük değer öncelikli olarak uygulanır).

Sohbetlerin paylaşılması için kararlaştırılan değer sıfırsa, kanal örneği hiçbir zaman paylaşılammaktadır. Bu alanı sıfır olarak ayarlayan diğer çıkış programları da kendi kanal yönetim ortamında benzer şekilde çalışır.

Sohbetlerin paylaşılması için kararlaştırılan değer sıfırdan büyükse, MQCXP SharingConversations , sonraki çağrılar için TRUE olarak ayarlandıysa, bu kanal örneğindeki diğer çıkış programlarının bu programla eşzamanlı olarak girilebileceğini gösterir.

Bir kanal çıkış programı yazdığınızda, bu programın etkileşim paylaşımını içerebilecek bir kanal yönetim ortamında çalıştırılıp çalıştırılmayacağını göz önünde bulundurun. Kanal yönetim ortamı etkileşimleri paylaşmayı gerektiriyorsa, bu etkiyi değiştirerek, MQCD alanlarını değiştirmenin diğer örneklerindeki etkiyi göz önünde bulundurun; tüm MQCD alanları, tüm paylaşım etkileşimleri arasında ortak değerlere sahiptir. Kanal yönetim ortamı kurulduktan sonra, çıkış programları MQCD alanlarını değiştirmeye çalışırsa sorunlarla karşılaşabilirler. Bunun nedeni, kanal yönetim ortamında çalışan çıkış programlarının diğer eşgörünümülerinin aynı alanları aynı anda değiştirme girişiminde bulunmaları olabilir. Çıkış programlarınızla bu durum ortaya çıkabiliyorsa, çıkış kodunuzda MQCD ' ye erişimi diziselleştirmeniz gerekir.

Sohbetleri paylaşmak üzere tanımlanmış bir kanalla çalışıyorsanız, ancak paylaşımın belirli bir kanal yönetim ortamında gerçekleşmesini istemiyorsanız, kanal yönetim ortamındaki ilk konuşmada kanal çıkışını başlattığınızda SharingConversations adlı MQCD değerini 1 ya da 0 olarak ayarlayın. SharingConversationsdeğerlerine ilişkin bir açıklama için bkz. [SharingConversations](#) .

## Örnek

Paylaşımın paylaşılması etkin.

Bir çıkış programı belirten istemci bağlantısı kanal tanımlamasını kullanıyorsunuz.

Bu kanal ilk kez başlatıldığında, çıkış programı başlatıldığı sırada bazı MQCD parametrelerinden bazılarını değiştirmektedir. Bunlar kanalın üzerinde işlem görmektedir, yani kanalın çalıştırdığı tanım, şu anda sağlanan olandan farklı. MQCXP SharingConversations değıştirgesi TRUE olarak ayarlandı.

Uygulamanın bu kanalı kullanarak bağlantı kurmasını sağlayan etkileşim, aynı özgün kanal tanımlamasına sahip olduğu için, daha önce başlatılan kanal örneğinde çalışır. Uygulamanın ikinci kez bağlandığı kanal yönetim ortamı, ilk bağlandığı zaman ile aynı yönetim ortağıdır. Sonuç olarak, çıkış programı tarafından değiştirilmiş olan tanımlamaları kullanır. Çıkış programı ikinci etkileşim için ilk kullanıma hazırlandığında, MQCD alanlarını değiştirebilse de, bunlar kanal tarafından işlem görmez. Bu aynı özellikler, kanal yönetim ortamını paylaşan sonraki tüm etkileşimler için de geçerlidir.

## **MQCONN olanağının kullanılması**

MQCNO yapısındaki bir kanal tanımlaması (MQCD) yapısı belirtmek için MQCONN çağrısını kullanabilirsiniz.

Bu, çağırın istemci uygulamasının çalıştırma zamanında istemci bağlantı kanalının tanımını belirtmesini sağlar. Daha fazla bilgi için, [MQCONN çağrısında MQCNO yapısının kullanılması](#) başlıklı konuya bakın. MQCONN 'i kullandığınızda, sunucuda yayınlanan arama sunucu düzeyi ve dinleyici yapılandırmasına bağlıdır.

İstemciden MQCONN 'i kullandığınızda, aşağıdaki seçenekler yoksayılr:

- MQCNO\_STANDARD\_BINDING

- MQCNO\_FASTPATH\_BINDING

Kullanabileceğiniz MQCD yapısı, kullanmakta olduğunuz MQCD sürüm numarasına bağlıdır. MQCD sürümlerine (MQCD\_VERSION) ilişkin bilgi için [MQCD Version](#)(MQCD Sürümü) konusuna bakın. Örneğin, kanal çıkış programlarını sunucuya geçirmek için, MQCD yapısını kullanabilirsiniz. MQCD Sürüm 3 ya da sonraki bir sürümünü kullanıyorsanız, bu yapıyı kullanarak, bir çıkış dizisini sunucuya geçirebilirsiniz. Var olan bir çıkışı değiştirmek yerine, her işlem için bir çıkış ekleyerek, şifreleme ve sıkıştırma gibi aynı iletide birden çok işlem gerçekleştirmek için bu işlevi kullanabilirsiniz. MQCD yapısında bir dizi belirtmezseniz, tek çıkış alanları denetlenir. Kanal çıkışı programlarıyla ilgili daha fazla bilgi için bkz. "[İleti alışverişi kanallarına ilişkin kanal çıkışı programları](#)" sayfa 993.

#### *MQCONNX üzerinde paylaşılan bağlantı tanıtıcıları*

Paylaşılan bağlantı tanıtıcılarını kullanarak, aynı süreç içindeki farklı iş parçacıkları arasında çekme noktaları paylaşabilirsiniz.

Paylaşılan bir bağlantı tanıtıcısı belirttiğinizde, MQCONNX çağrısından döndürülen bağlantı tanıtıcısı, süreçteki herhangi bir iş parçacığında sonraki MQI çağrılarına aktarılabilir.






**Not:** Paylaşılan bağlantı tanıtıcılarını desteklemeyen bir sunucu kuyruk yöneticisine bağlanmak için, bir IBM MQ MQI client üzerindeki paylaşılan bağlantı tanıtıcısını kullanabilirsiniz.

Daha fazla bilgi için "[MQCONNX olanağının kullanılması](#)" sayfa 950 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 oluşturulacağı ve bağlantılı olduğu yol, kullanılan platforma ve programlama diline göre değişiklik gösterir.

Bir uygulama istemci ortamında çalıştırılacaksa, bu uygulamayı aşağıdaki tabloda gösterilen dillerde yazabilirsiniz:

<i>Çizelge 137. İstemci ortamlarında desteklenen programlama dilleri</i>						
<b>İstemci altyapısı</b>	<b>C</b>	<b>C++</b>	<b>COBOL</b>	<b>pTAL</b>	<b>RPG</b>	<b>Visual Basic</b>
 AIX	Evet	Evet	Evet			
 IBM i	Evet		Evet		Evet	
 Linux	Evet	Evet	Evet			
 Solaris	Evet	Evet	Evet			
 Windows	Evet	Evet	Evet			Evet

### **C uygulamalarını IBM MQ MQI client koduyla bağlantılandırma**

IBM MQ MQI client üzerinde çalıştırmak istediğiniz IBM MQ uygulamanızı yazmanız, it kodunu IBM MQ MQI client koduna bağlamanız gerekir.

Uygulamanızı IBM MQ MQI client koduna iki şekilde bağlayabilirsiniz:

1. Doğrudan, uygulamanızı bir kuyruk yöneticisine bağlayarak, kuyruk yöneticisinin uygulamanızla aynı makinede olması gereken bir durumda olmalıdır.
2. Aynı ya da farklı bir makinede kuyruk yöneticilerine erişim sağlayan bir istemci kitaplığı dosyası.

IBM MQ , her ortam için bir istemci kitaplığı dosyası sağlar:

#### **AIX**

İş parçacıklı uygulamalar için libmqic.a kitaplığı ya da iş parçacıklı uygulamalar için libmqic\_r.a kitaplığı.

## Linux Linux

İş parçacıklı uygulamalar için libmqic.so kitaplığı ya da iş parçacıklı uygulamalar için libmqic\_r.so kitaplığı.

## IBM i IBM i

İş parçacıklı uygulamalar için LIBMQIC istemci hizmeti programıyla ya da iş parçacıklı uygulamalar için LIBMQIC\_R hizmet programı ile istemci uygulaması arasında bağ tanımlayın.

## Solaris Solaris

libmqic.so.

Yalnızca IBM MQ MQI client for Solaris kurulu olan bir makinede bulunan programları kullanmak istiyorsanız, bunları istemci kitaplığıyla bağlamak için yeniden derlemelisiniz:

```
$ /opt/SUNWsprio/bin/cc -o prog_name prog_name.c -mt -lmqic \
-lsocket -lc -lnsl -ldl
```

Parametreler, gösterildiği gibi doğru sırayla girilmelidir.

## Windows Windows

MQIC32.LIB.

## ULW C++ uygulamalarını IBM MQ MQI client koduyla bağlantılandırma

İstemcide C++ içinde çalışacak uygulamalar yazabilirsiniz. Oluşturma yöntemleri, ortama göre değişir.

C++ uygulamalarınızı nasıl bağlayabilmeye ilişkin bilgi için [Building IBM MQ C++ programs](#) başlıklı konuya bakın.

C++ kullanmanın tüm yönleriyle ilgili tüm ayrıntılar için bkz. [C++ kullanılması](#)

## Multi COBOL uygulamalarını IBM MQ MQI client koduyla bağlantılandırma

IBM MQ MQI client üzerinde çalıştırmak istediğiniz bir COBOL uygulaması yazdığınızda, bu uygulamayı uygun bir kitaplıkla ilişkilendirmeniz gerekir.

IBM MQ, her ortam için bir istemci kitaplığı dosyası sağlar:

## AIX AIX

Link your non-threaded COBOL application with the library libmqicb.a or threaded COBOL application with libmqicb\_r.a.

## IBM i IBM i

İş parçacıklı uygulamalar için iş parçacığı dışı uygulamalar için COBOL istemci uygulaması ya da iş parçacıklı uygulamalar için AMQCSSTUB\_R hizmet programı ile bağ tanımlayın.

## Solaris Solaris

Link your non-threaded COBOL application with the library libmqicb.so or threaded COBOL application with libmqicb\_r.so.

## Windows Windows

Uygulama kodunuzu, 32 bit COBOL için MQICCB.B kitaplığınızla ilişkilendirin. IBM MQ MQI client for Windows, 16 bit COBOL'yi desteklemez.

## Windows Visual Basic uygulamalarını IBM MQ MQI client koduyla bağlantılandırma

Microsoft Visual Basic uygulamalarını Windows üzerindeki IBM MQ MQI client kodlarıyla bağlantılayabilirsiniz.

IBM MQ 9.0' tan Microsoft Visual Basic 6.0 için IBM MQ desteği kullanımdan kaldırılmıştır. IBM MQ classes for .NET are the recommended replacement technology. Daha fazla bilgi için bkz [Developing .NET applications](#).



Visual Basic uygulamanızı aşağıdaki içerme dosyalarıyla bağlantılayın:

**CMQB.bas**

MQI

**CMQBB.bas**

MQAI

**CMQCFB.bas**

PCF komutları

**CMQXB.bas**

Kanallar

Set mqtype=2 for the client in the Visual Basic compiler, to ensure the correct automatic selection of the client dll:

**MQIC32.dll**

Windows 7, Windows 8, Windows 2008 ve Windows 2012

**İlgili kavramlar**

“Visual Basic’inde kodlama” sayfa 1087

Microsoft Visual Basic' taki IBM MQ programlarını kodlarken dikkate alınacak bilgiler. Visual Basic yalnızca Windows’ünde desteklenir.

“Preparing Visual Basic programs in Windows” sayfa 1056

Windows’ünde Microsoft Visual Basic programlarını kullanırken dikkate alınması 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 client’ünde 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 istemciye 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 “Bir havuzdaki bağlanma öncesi çıkışı kullanarak bağlantı tanımlarına gönderme yapılması” sayfa 1023.

2. MQSERVER ortam değişkeni ayarlıysa, tanımladığı kanal kullanılır.
3. Bir mqclient.ini dosyası tanımlanmışsa ve bir ServerConnectionParms içeriyorsa, tanımladığı kanal kullanılır. Daha fazla bilgi için [Yapılandırma dosyası kullanılarak istemci yapılandırılması](#) ve [İstemci yapılandırma dosyasının KANAL kısmı](#) başlıklı konuya bakın.
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. [İstemci kanal tanımlama çizelgesine adreslenebilir Web erişimi](#).
5. Bir mqclient.ini dosyası tanımlıysa ve ChannelDefinitionDizini ve ChannelDefinitionDosya özniteliklerini içeriyorsa, bu öznitelikler istemci kanal tanımlama çizelgesinin yerini belirlemek için kullanılır. Daha fazla bilgi için [Yapılandırma dosyası kullanılarak istemci yapılandırılması](#) ve [İstemci yapılandırma dosyasının KANAL kısmı](#) başlıklı konuya bakın.
6. Son olarak, ortam değişkenleri ayarlanmazsa, istemci mqs.ini dosyasında DefaultPrefix ' den oluşturulan bir yol ve ada sahip bir istemci kanal tanımlama çizelgesini arar. İstemci kanal tanımlama çizelgesinin aranması başarısız olursa, istemci aşağıdaki yolları kullanır:

-   UNIX and Linux işletim tarihinde: /var/mqm/AMQCLCHL.TAB
-  Windows işletim tarihinde: C:\Program Files\IBM\MQ\amqclchl.tab
-  IBM i işletim tarihinde: /QIBM/UserData/mqm/@ipcc
-  MQ Appliance üzerinde: 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 çizelgesi](#)

[İstemci kanal tanımlama çizelgesine Web üzerinden adreslenebilir erişim](#)

### **İlgili görevler**

[Sunucu ve istemci arasındaki bağlantıların yapılandırılması](#)

### **İlgili başvurular**

[MQSERVER](#)

[MQCHLLIB](#)

[MQCHLTAB](#)

[MQCCDTURL](#)

[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 alt düzey bir kanal tanımlamak için kullanılır. MQSERVER, IBM MQ sunucusunun yerini ve kullanılacak iletişim yöntemini belirtir.

## **MQCHLLIB**

MQCHLLIB ortam deęiřkeni, istemci kanal tanımlama çizelgesini (CCDT) içeren dosyanın dizin yolunu belirtir. Dosya sunucuda yaratılır, ancak IBM MQ MQI client iş istasyonuna kopyalanabilir.

## **MQCHLTAB**

MQCHLTAB ortam deęiřkeni, istemci kanal tanımlama çizelgesini (CCDT) içeren dosyanın adını belirtir.

IBM MQ 9.0' den MQCCDTURL ortam deęiřkeni, MQCHLLIB ve MQCHLTAB ortam deęiřkenlerinin bir birleřimini ayarlamak için eřdeęer bir yetenek saęlar. MQCCDTURL, istemci kanal tanımlama çizelgesinin elde edilebileceęi tek bir deęer olarak bir dosya, ftp ya da http URL saęlamanızı saęlar. Daha fazla bilgi için bkz. [İstemci kanal tanımlama çizelgesine adreslenebilir Web eriřimi](#).

*MQCNO yapısını kullanarak istemci uygulamalarının kuyruk yöneticilerine baęlanması*

Kanala iliřkin tanımlamayı, MQCONNX çağrısının MQCNO yapısı kullanılarak saęlanan bir kanal tanımlama yapısında (MQCD) belirtebilirsiniz.

Daha fazla bilgi için [MQCONNX çağrısında MQCNO yapısının kullanılması](#) başlıklı konuya bakın.

*İstemci uygulamaları kuyruk yöneticilerine istemci kanal tanımlama çizelgesi kullanılarak baęlanması*

MQSC DEFINE CHANNEL komutunu kullanırsanız, saęladığınız ayrıntılar istemci kanal tanımlama çizelgesine (ccdt) yerleřtirilir. MQCONN ya da MQCONNX çağrısının **QMGRName** deęiřtirgesinin içerięi, istemcinin hangi kuyruk yöneticisini baęlamaya baęlandığı saptar.

Bu dosyaya istemcinin eriřeceęi kanalı belirlemek için istemci tarafından eriřilir. Birden fazla uygun kanal tanımlaması varsa, kanal seęimi, istemci kanal aęırlığı (CLNTWGHT) ve baęlantı benzerlięi (BENZEřIMI) kanalı özniteliklerinden etkilenir.

*Otomatik istemci yeniden baęlantısının kullanılması*

Bir dizi bileřen yapılandırarak, istemci uygulamalarınızın otomatik olarak yeniden baęlanarak, ek kod yazmadan otomatik olarak yeniden baęlanabilmesini saęlayabilirsiniz.

Otomatik istemci yeniden baęlantısı *yerleřik*. Baęlantı, istemci uygulama programının herhangi bir noktasında otomatik olarak geri yüklenir ve nesnelere açma tanıtıcılarının tümü geri yüklenir.

Buna karřılık, el ile yeniden baęlanma, istemci uygulamasının MQCONN ya da MQCONNX kullanarak yeniden baęlantı oluřturmasını ve nesnelere yeniden açmasını gerektirir. Otomatik istemci yeniden baęlantısı birçok istemci uygulaması için uygundur, ancak tüm istemci uygulamaları için uygun deęildir.

Ek bilgi için [Automatic client reconnection](#) başlıklı konuya bakın.

*İstemci kanalı tanımlama çizelgesinin rolü*

İstemci kanal tanımlama çizelgesi (CCDT), istemci baęlantı kanallarına iliřkin tanımlamaları içerir. Bu, özellikle istemci uygulamalarınızın çeřitli kuyruk yöneticilerine baęlanmanız gerekmesi durumunda yararlı olur.

İstemci kanalı tanımlama çizelgesi, bir kuyruk yöneticisi tanımladığınızda yaratılır. Aynı dosya, birden çok IBM MQ istemcisi tarafından kullanılabilir.

İstemci uygulamasının CCDT kullanmasını saęlamak için bir dizi yol vardır. CCDT, istemci bilgisayarına kopyalanabilir. CCDT ' yi birden çok istemci tarafından paylaşılan bir konuma kopyalayabilirsiniz. CCDT ' yi istemci tarafından paylaşılan bir dosya olarak eriřilebilir, ancak sunucu üzerinde kalır.

CCDT, IBM MQ 9.0'den, bir URI aracılıęıyla eriřilebilen merkezi bir konumda barındırılabilir ve her devreye alınan istemci için CCDT' yi tek tek güncelleřtirmeye gerek kalmazdır.

## **İlgili kavramlar**

[İstemci kanal tanımlama çizelgesi](#)

[İstemci kanal tanımlama çizelgesine Web adreslenebilir eriřimi](#)

## **İlgili görevler**

[İstemci-baęlantı kanalı tanımlarına eriřilmesi](#)

### CCDT 'deki kuyruk yöneticisi grupları

İstemci kanalı tanımlama çizelgesinde (CCDT) bir bağlantı kümesi tanımlamak için *kuyruk yöneticisi grubu* olarak tanımlayabilirsiniz. Bir uygulamayı, kuyruk yöneticisi grubunun bir parçası olan bir kuyruk yöneticisine bağlayabilirsiniz. This can be done by prefixing the queue manager name on an MQCONN or MQCONNX call with an asterisk.

Aşağıdakiler nedeniyle birden çok sunucu makinesinden bağlantı tanımlamayı seçebilirsiniz:

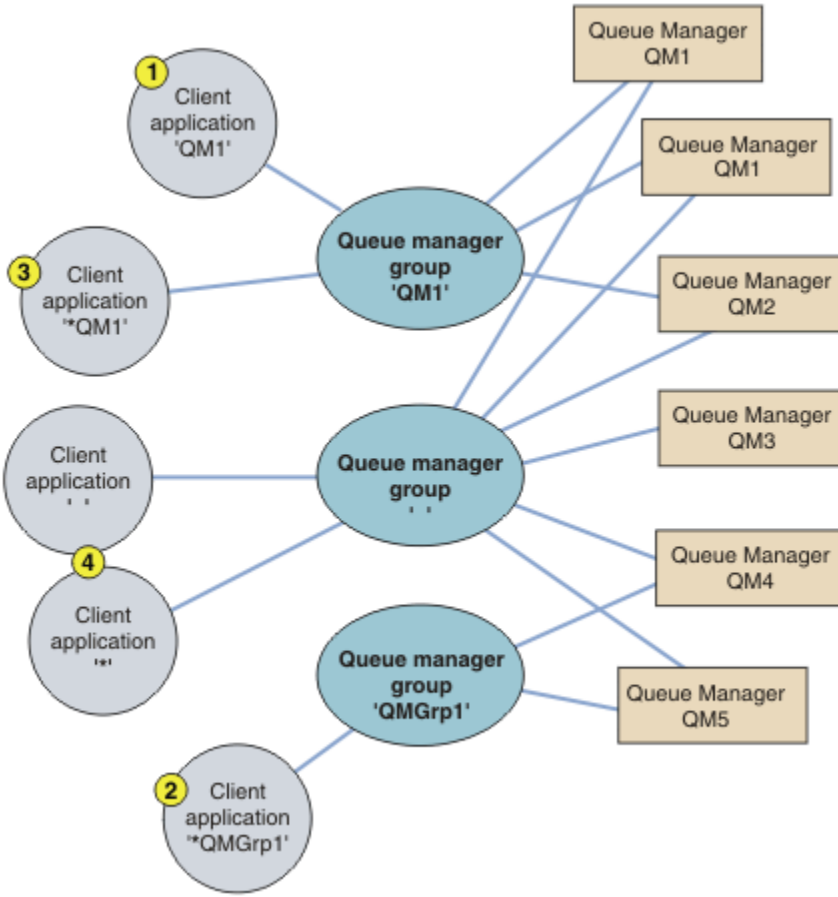
- Kullanılabilirliği artırmak için çalışmakta olan kuyruk yöneticilerinden herhangi birine bir istemci bağlamak istiyorsunuz.
- Bir istemciyi, son kez başarıyla bağlandığınız kuyruk yöneticisine yeniden bağlamak istiyorsanız, ancak bağlantı başarısız olursa, farklı bir kuyruk yöneticisine bağlanın.
- İstemci programındaki MQCONN komutunu yeniden vererek, bağlantı başarısız olursa, istemci bağlantısını farklı bir kuyruk yöneticisinde yeniden deneyebilmeyi isteyebilirsiniz.
- Herhangi bir istemci kodu yazmadan, bağlantı başarısız olursa, istemci bağlantısını otomatik olarak başka bir kuyruk yöneticisine yeniden bağlamayı isteyebilirsiniz.
- Bir istemci kodunu yazmadan, yedek yönetim ortamı gerekiyorsa, çok eşgörunümlü bir kuyruk yöneticisinin farklı bir eşgörunümesine otomatik olarak bağlanmak istiyorsunuz.
- İstemci bağlantılarınızı, bazı kuyruk yöneticileriyle ve bazı kuyruk yöneticilerine diğerlerinden daha fazla bağlanırken dengelemek istiyorsunuz.
- Çok sayıda istemci bağlantısının yeniden bağlanmasını birden çok kuyruk yöneticisi ve zaman içinde yeniden bağlamayı istiyorsanız, bu durumda, yüksek bağlantı hacmi bir hataya neden olur.
- Herhangi bir istemci uygulama kodunu değiştirmeden kuyruk yöneticilerinizi hareket ettirebilmeyi istiyorsunuz.
- Kuyruk yöneticisi adlarını bilmeye gerek olmayan istemci uygulama programları yazmak istiyorsunuz.

Her zaman farklı kuyruk yöneticilerine bağlanmak uygun değildir. An extended transactional client or a Java client in WebSphere Application Server, for example, might need to connect to a predictable queue manager instance. Otomatik istemci yeniden bağlanması IBM MQ classes for Javatafından desteklenmez.

Kuyruk yöneticisi grubu, istemci kanal tanımlama çizelgesinde (CCDT) tanımlı bir bağlantı kümesidir. Küme, kendi kanal tanımlamalarında **QMNAME** özniteliğinin aynı değerine sahip üyeleri tarafından tanımlanır.

Şekil 102 sayfa 957 is a graphical representation of a client connection table, showing three queue manager groups, two named queue manager groups written in the CCDT as **QMNAME** (QM1) and **QMNAME** (QMGRP1), and one blank or default group written as **QMNAME** ( ' ' ).

1. Queue manager group QM1 has three client connection channels, connecting it to queue managers QM1 and QM2. QM1 , iki farklı sunucu üzerinde bulunan çok eşgörunümlü 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 bağlantı kanallarına sahiptir.



Şekil 102. Kuyruk yöneticisi grupları

Bu istemci bağlantı çizelgesini kullanmanın dört örneği, Şekil 102 sayfa 957'indeki numaralandırılmış istemci uygulamalarının yardımlarıyla açıklanmaktadır.

1. İlk örnekte, istemci uygulaması bir kuyruk yöneticisi adını ( QM1) MQCONN ya da MQCONNX MQI çağrısına **QmgrName** değiştirgesi olarak geçirir. IBM MQ istemci kodu, eşleşen kuyruk yöneticisi grubunu ( QM1) seçer. The group contains three connection channels, and the IBM MQ MQI client tries to connect to QM1 using each of these channels in turn until it finds an IBM MQ listener for the connection attached to a running queue manager called QM1.

Bağlantı denemelerinin sırası, istemci bağlantısı BENZEŞİMİ özniteliliğinin değerine ve istemci kanal ağırlıklandırılmalarına bağlıdır. Bu kısıtlar içinde, bağlantı kurma yükünü dağıtmak için, her ikisi de olası üç bağlantı üzerinden ve zaman içinde bağlantı oluşturma sırası raslantılanmış olarak sıralanır.

İstemci uygulaması tarafından yayınlanan MQCONN ya da MQCONNX çağrısı, çalışmakta olan bir QM1örneğine bağlantı kurulduğunda başarılı olur.

2. İkinci örnekte, istemci uygulaması başına yıldız imi olan bir kuyruk yöneticisi adını ( \*QMGrp1) MQCONN ya da MQCONNX MQI çağrısına **QmgrName** değiştirgesi olarak geçirir. IBM MQ istemcisi, eşleşen kuyruk yöneticisi grubunu ( QMGrp1) seçer. Bu grup iki istemci bağlantı kanalı içerir ve IBM MQ MQI client , sırayla her bir kanalı kullanarak *herhangi bir* kuyruk yöneticisine bağlanmayı dener. Bu örnekte, IBM MQ MQI client ' in başarılı bir bağlantı yapması gerekir; bağlandığı kuyruk yöneticisinin adı önemli değil.

Bağlantı kurma girişimlerinin sırası daha önce de aynıdır. Tek fark, kuyruk yöneticisi adını yıldız işaretiyle önleyerek istemci, kuyruk yöneticisi adının ilgili olmadığını gösterir.

İstemci uygulaması tarafından yayınlanan MQCONN ya da MQCONNX çağrısı, QMGrp1 kuyruk yöneticisi grubundaki kanallara bağlı herhangi bir kuyruk yöneticisinin çalışan bir eşgörünümünün çalışan bir eşgörünümüdür için bağlantı kurulduğunda başarılı olur.

3. The third example is essentially the same as the second because the **QmgrName** parameter is prefixed by an asterisk, \*QM1. Bu örnek, bir kanal tanımlamasındaki QMNAME özniteliğini tek başına inceleyerek istemci kanalı bağlantısının hangi kuyruk yöneticisini denetleyeceğini belirleyemediğinizi gösterir. Kanal tanımlamasının **QMNAME** özniteliğinin QM1 olması, QM1 adlı bir kuyruk yöneticisine bağlantı yapılmasını zorunlu kılmamak için yeterli değildir. İstemci uygulamanız **QmgrName** parametresini bir yıldız işaretiyle önekler, sonra herhangi bir kuyruk yöneticisi olası bir bağlantı hedefidir.

Bu durumda, istemci uygulaması tarafından yayınlanan MQCONN ya da MQCONNX çağrılarını, çalışmakta olan bir QM1 ya da QM2 yönetim ortamında bir bağlantı kurulduğunda başarılı olur.

4. Dördüncü örnek, varsayılan grubun kullanımını gösterir. Bu durumda, istemci uygulaması MQCONN ya da MQCONNX MQI çağrısında **QmgrName** değiştirgesi olarak bir yıldız imi ( '\*' ) ya da boş ' ' değerini geçirir. İstemci kanalı tanımlamasında kural olarak, boş bir **QMNAME** özniteliği, varsayılan kuyruk yöneticisi grubunu belirtir ve boşluk ya da yıldız işareti **QmgrName** değiştirgesi boş bir **QMNAME** özniteiyle eşleşir.

Bu örnekte, varsayılan kuyruk yöneticisi grubunun tüm kuyruk yöneticilerine istemci kanalı bağlantıları vardır. Varsayılan kuyruk yöneticisi grubu seçilerek, uygulama gruptaki herhangi bir kuyruk yöneticisine bağlı olabilir.

İstemci uygulaması tarafından yayınlanan MQCONN ya da MQCONNX çağrısı, herhangi bir kuyruk yöneticisinin çalışmakta olan bir eşgörünümüye bağlantı kurulduğunda başarılı olur.

**Not:** Varsayılan bir kuyruk yöneticisinden varsayılan grup farklıdır; ancak, bir uygulama varsayılan kuyruk yöneticisi grubuna ya da varsayılan kuyruk yöneticisine bağlanmak için boş bir **QmgrName** değiştirgesi kullanıyor. Varsayılan kuyruk yöneticisi grubu kavramı yalnızca istemci uygulamalarıyla ilgilidir ve bir sunucu uygulamasına varsayılan kuyruk yöneticisi içerir.

İstemci bağlantı kanallarınızı yalnızca bir kuyruk yöneticisinden tanımlayın; bu kanallar, ikinci ya da üçüncü kuyruk yöneticisine bağlanan kanallar da içinde olmak üzere. Bunları iki kuyruk yöneticisi üzerinde tanımlamayın ve sonra iki istemci kanal tanımlama çizelgelerini birleştirmeyi deneyin. İstemci kanal tanımlama çizelgesine yalnızca bir istemci kanal tanımlama çizelgesi erişebilir.

## Örnekler

Konunun başlangıcındaki kuyruk yöneticisi gruplarının kullanılmasına ilişkin nedenlerin [listesine](#) yeniden bakın. Bir kuyruk yöneticisi grubu bu yetenekleri nasıl sağlıyor?

### **Kuyruk yöneticisi kümenlerinden herhangi birine bağlanın.**

Define a queue manager group with connections to all the queue managers in the set, and connect to the group using the **QmgrName** parameter prefixed by an asterisk.

### **Aynı kuyruk yöneticisine yeniden bağlanın, ancak son kez bağlanılan kuyruk yöneticisi kullanılmıyorsa, farklı bir kuyruk yöneticisine bağlanın.**

Daha önce olduğu gibi bir kuyruk yöneticisi grubu tanımlayın, ancak her istemci kanalı tanımlamasındaki özniteliği **AFFINITY** (PREFERENT) olarak ayarlayın.

### **Bir bağlantı başarısız olursa, başka bir kuyruk yöneticisiyle bağlantı kurmayı yeniden deneyin.**

Bir kuyruk yöneticisi grubuna bağlanın ve bağlantı bozuk ya da kuyruk yöneticisi başarısız olursa, MQCONN ya da MQCONNX MQI çağrısını yeniden yayınlayın.

### **Bir bağlantı başarısız olursa otomatik olarak başka bir kuyruk yöneticisine yeniden bağlanın.**

MQCONNX **MQCNO** seçeneğini MQCNO\_RECONNECT kullanarak bir kuyruk yöneticisi grubuna bağlanın.

### **Çok eşgörünümlü bir kuyruk yöneticisinin farklı bir eşgörünümüne otomatik olarak yeniden bağlanın.**

Önceki örneğe benzer şekilde yapın. Bu durumda, kuyruk yöneticisi grubunu belirli bir çok eşgörünümlü kuyruk yöneticisinin eşgörünümlerine bağlanmayı sınırlamak için sınırlamak istiyorsanız, bu grubu yalnızca çok eşgörünümlü kuyruk yöneticisi yönetim ortamlarıyla bağlantılarla tanımlayın.

You can also ask the client application to issue its MQCONN or MQCONNX MQI call with no asterisk prefixed to the **QmgrName** parameter. Bu şekilde, istemci uygulaması yalnızca adlandırılmış kuyruk yöneticisine bağlanabilir. Son olarak, **MQCNO** seçeneğini MQCNO\_RECONNECT\_Q\_MGR olarak ayarlayabilirsiniz. Bu seçenek, önceden bağlı olan aynı kuyruk yöneticisine yeniden bağlantı kabul eder. Bu değeri, normal bir kuyruk yöneticisinin aynı eşgörünümüyle yeniden bağlantıları sınırlandırmak için de kullanabilirsiniz.

## **Kuyruk yöneticilerindeki istemci bağlantılarını dengelemek, bazı kuyruk yöneticilerine diğerlerinden daha fazla bağlı istemci bağlantısı kurmasını sağlar.**

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, istemci yeniden bağlantı yükünü eşit olmayan bir şekilde dağıt ve zaman içinde dağıt.**

Önceki örneğe benzer şekilde yapın. IBM MQ MQI client , kuyruk yöneticilerindeki yeniden bağlantıları rasgele oluşturur ve zaman içinde yeniden bağlantıları dağıtır.

## **İstemci kodunu değiştirmeden kuyruk yöneticilerinizi taşıyın.**

CCDT, istemci uygulamanızı kuyruk yöneticisinin yerinden yalıtmanızı sağlar. CCDT, istemcide tanımlanabilen, paylaşılan bir yerden okunabilen ya da bir web sunucusundan getirilen bir veri dosyasıdır. Ek bilgi için [İstemci kanal tanımlama çizelgesibaşlıklı konuya](#) bakın.

## **Kuyruk yöneticisi adlarını tanımayan bir istemci uygulaması yazın.**

Kuyruk yöneticisi grup adlarını kullanın ve kuruluşunuzda istemci uygulamalarınızla ilgili olan 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, kuyruk paylaşım grubunun bir parçası olan bir kuyruk yöneticisine bağlayabilirsiniz. Bu, MQCONN ya da MQCONNX çağrısındaki kuyruk yöneticisi adı yerine kuyruk paylaşım grubu adı kullanılarak yapılabilir.

Kuyruk paylaşım grupları en çok dört karakterden oluşan bir ada sahiptir. Adın ağızda benzersiz olması ve kuyruk yöneticisi adlarından farklı olması gerekir.

İstemci kanalı tanımlaması, gruptaki kullanılabilir bir kuyruk yöneticisine bağlanmak için kuyruk paylaşım grubu soysal arabirimini kullanmalıdır. Daha fazla bilgi için [İstemcinin kuyruk paylaşım grubuna bağlanması](#) başlıklı konuya bakın. İletişimci, kuyruk yöneticisinin bağlandığı kuyruk yöneticisinin kuyruk paylaşım grubunun bir üyesi olduğundan emin olmak için bir onay imi yapılır.

Paylaşılan kuyruklara ilişkin ek bilgi için [Paylaşılan kuyruklar ve kuyruk paylaşım grupları](#) başlıklı konuya bakın.

## **Kanal ağırlıklandırma ve benzeşme örnekleri**

Bu örnekler, sıfırsız ClientChannelWeights kullanıldığında istemci-bağlantı kanallarının nasıl seçildiğini gösterir.

ClientChannelAğırlık ve ConnectionAffinity kanalı öznitelikleri, bir bağlantı için birden çok uygun kanal kullanılabilir olduğunda istemci bağlantı kanallarının nasıl seçildiğini denetler. Bu kanallar, daha yüksek kullanılabilirlik, iş yükü dengelemesi ya da her ikisi için farklı kuyruk yöneticilerine bağlanmak üzere yapılandırılmış. Bazı kuyruk yöneticilerinden biriyle bağlantının sonuçlanabileceği MQCONN çağrıları, şu konuda açıklandığı gibi, kuyruk yöneticisi adına bir yıldız işaretiyle önek olarak önek vermelidir: MQCONN çağrılarına ilişkin örnekler: Örnek 1. Kuyruk yöneticisi adı bir yıldız işareti (\*) içeriyor.

Bir bağlantı için geçerli aday kanalları, QMNAME özniteliğinin MQCONN çağrısında belirlenen kuyruk yöneticisi adıyla eşleştiği durumlarda yer alıyor. Bir bağlantıya ilişkin tüm uygun kanalların ClientChannelAğırlık değeri sıfır (varsayılan) ise, örnekteki gibi alfabetik sırayla seçilir: MQCONN çağrılarına örnekler: Örnek 1. Kuyruk yöneticisi adı bir yıldız işareti (\*) içeriyor.

Aşağıdaki örnekler, sıfır olmayan ClientChannelWeights kullanılmadığında ne olacağını göstermektedir. Bu özellik sözde rasgele kanal seçimini içerdiği için, örneklerin kesinlikle gerçekleştireceği gibi bir işlem dizisi gösterdiğine dikkat edin.

## **Örnek 1. ConnectionAffinity değeri TERCİH edilen olarak ayarlandığında kanal seçilmesi**

This example illustrates how an IBM MQ MQI client selects a channel from a CCDT, where the ConnectionAffinity is set to PREFERRED.

Bu örnekte, bir kuyruk yöneticisi tarafından sağlanan bir İstemci Kanal Tanımlama Çizelgesi (CCDT) kullanan istemci makinelerinden oluşan bir dizi istemci makinelerinden biri. 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 sorunları MQCONN (\*CORE)

QMNAME özniteliği eşleşmediği için, Kanal A bu bağlantı için bir aday değil. B, C ve D kanalları aday olarak tanımlanır ve ağırlıklandırmalarına dayalı olarak bir tercih sırasına konur. Bu örnekte sıra C, B, D olabilir. İstemci, core2.ops.company.exampleadresindeki kuyruk yöneticisine bağlanmayı dener. MQCONN çağrısı kuyruk yöneticisi adına bir yıldız işareti eklediği için, o adresdeki kuyruk yöneticisinin adı denetlenmez.

AFFINITY (PREFERRED) ile, bu istemci makinesinde her bağlantı kesildiğinde, kanalları aynı ilk tercih sırasına göre yerleştirecek şekilde not almanız önemlidir. Bu, bağlantılar farklı süreçlerden ya da farklı zamanlarda olduğunda da geçerlidir.

Bu örnekte, core.2.ops.company.example konumundaki kuyruk yöneticisine ulaşılamıyor. İstemci core1.ops.company.example ile bağlantı kurmaya çalışır çünkü kanal B tercih sırasının yanında yer alıyor. Buna ek olarak, C kanalı en az tercih edilen düzeye indirgenir.

Aynı uygulama tarafından ikinci bir MQCONN (\*CORE) çağrısı yayınlanıyor. Kanal C önceki bağlantı tarafından indirgenmiş, bu yüzden en çok tercih edilen kanal artık B ' dir. Bu bağlantı core1.ops.company.exampleolarak yapılır.

Aynı İstemci Kanal Tanımlaması Tablounu paylaşan ikinci bir makine, kanalları farklı bir başlangıç sırasına göre yerleştirmektedir. Örneğin, D, B, C. Normal koşullar altında, tüm kanallarda çalışan, bu makineden uygulamalar core3.ops.company.example ile bağlantılıysa, ilk makineden core2.ops.company.examplebağlantısı bulunur. Bu, her bir istemcinin kullanılabilir durumda olması durumunda aynı kuyruk yöneticisine bağlanmasını sağlarken, birden çok kuyruk yöneticisi arasında çok sayıda istemci için iş yükü dengelemesi yapılmasına olanak sağlar.

#### Örnek 2. ConnectionAffinity ' un NONE (Yok) olarak ayarlandığında kanal seçilmesi

Bu örnek, IBM MQ MQI client 'un CCDT' den ConnectionAffinity ' un NONE (Yok) değerine ayarlandığı bir kanalı nasıl seçeceğini gösterir.

Bu örnekte, bir kuyruk yöneticisi tarafından sağlanan bir İstemci Kanal Tanımlama Çizelgesi (CCDT) kullanan sayıda istemci vardır. 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) sorunlarını yayınlıyor. Önceki örnekte olduğu gibi, QMNAME eşleşmediği için Kanal A dikkate alınmıyor. Kanal B, C ya da D, %50, %30 ya da %20 olasılıkları ile ağırlıklandırmalarına dayalı olarak seçilirdir. Bu örnekte kanal B seçilebilir. Kalıcı bir tercih sırası oluşturulmadı.

İkinci bir MQCONN (\*CORE) çağrısı yapıldı. Yine aynı olasılıklara sahip, uygulanabilir üç kanaldan biri seçiliyor. Bu örnekte, C kanalı seçilirdir. Ancak, core2.ops.company.example yanıt vermez, bu nedenle kalan aday kanallar arasında başka bir seçenek de yapılır. Kanal B seçildi ve uygulama core1.ops.company.exampleile bağlantılıdır.

BENZEŞİMİ (NONE) ile, MQCONN çağrılarının her biri diğerinden bağımsızdır. Bu nedenle, bu örnek uygulama üçüncü bir MQCONN (\*CORE) yaptığında, B ya da D ' den birini seçmeden önce, bozuk kanal C ile bağlantı kurma girişiminde bulunmayı bir kez daha deneyebilir.

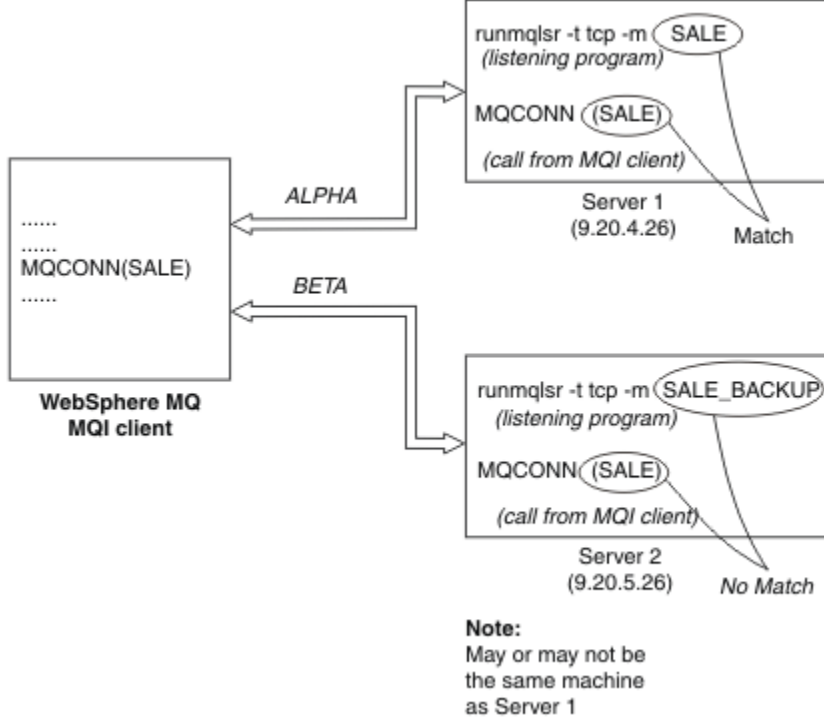


### MQCONN çağrılarının örnekleri

Belirli bir kuyruk yöneticisine ya da kuyruk yöneticilerinden birine bağlanmak için MQCONN kullanımına ilişkin örnekler.

Aşağıdaki örneklerin her birinde, ağ aynıdır; aynı IBM MQ MQI client' den iki sunucuya tanımlanmış bir bağlantı vardır. (Bu örneklerde, MQCONN çağrısının yerine MQCONNX çağrısı kullanılabilir.)

Sunucu makinelerinde çalışan iki kuyruk yöneticisi vardır; biri SALE ve diğeri adı SALE\_BACKUP.



Şekil 103. MQCONN örneği

Bu örneklerdeki kanallara ilişkin tanımlamalar şunlardır:

SALE tanımlamaları:

```
DEFINE CHANNEL(ALPHA) CHLTYPE(SVRCONN) TRPTYPE(TCP) +  
DESCR('Server connection to IBM MQ MQI client')  
  
DEFINE CHANNEL(ALPHA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +  
CONNNAME(9.20.4.26) DESCR('IBM MQ MQI client connection to server 1') +  
QMNAME(SALE)  
  
DEFINE CHANNEL(BETA) CHLTYPE(CLNTCONN) TRPTYPE(TCP) +  
CONNNAME(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 tanımlamaları aşağıdaki gibi özetlenebilir:

Ad	KLASÖR	TRPTYPE	ADı	QMNAME
Alfa	NTNTCONN	TCP	9.20.4.26	SALE
Beta	NTNTCONN	TCP	9.20.5.26	SALE

### *MQCONN örneklerinin gösterisi*

Örnekler, birden çok kuyruk yöneticisinin yedek sistem olarak kullanılmasını gösterir.

Sunucu 1 'e ilişkin iletişim bağlantısının geçici olarak bozulması olduğunu varsayın. Yedekleme sistemi olarak birden çok kuyruk yöneticisinin kullanılması gösterilir.

Her bir örnek, farklı bir MQCONN çağrısını kapsar ve aşağıdaki kuralları uygulayarak, sunulan belirli örnekte neler olduğuna ilişkin bir açıklama sunar:

1. İstemci kanal tanımlama çizelgesi (CCDT), MQCONN çağrısında belirtilen kuyruk yöneticisi adına (QMNAME alanı) karşılık gelen alfabetik kanal adı sırasından taranır.
2. Bir eşleşme bulunursa, kanal tanımlaması kullanılır.
3. Kanalı, bağlantı adı (CONNNAME) ile tanıtılan makineye başlatma girişiminde bulunmanız gerekir. Bu başarılıysa, uygulama devam eder. Bu işlem aşağıdakileri gerektirir:
  - Sunucuda çalışmakta olan bir dinleyici.
  - İstemcinin bağlanmak istediği kuyruk yöneticisine (belirtilmişse) bağlı olan dinleyici.
4. Kanalı başlatma girişimi başarısız olursa ve istemci kanal tanımlama çizelgesinde birden çok giriş varsa (bu örnekte iki giriş varsa), dosya daha fazla eşleşme için aranır. Eşleşme bulunursa, işlem adım 1 'de devam eder.
5. Eşleşme bulunamazsa ya da istemci kanal tanımlama çizelgesinde başka giriş yoksa ve kanal başlatılamadıysa, uygulama bağlanamıyor demektir. MQCONN çağrısında, uygun bir neden kodu ve tamamlanma kodu döndürülür. Uygulama, döndürülen nedene ve tamamlanma kodlarına dayalı olarak işlem yapabilir.

#### *Örnek 1. Kuyruk yöneticisi adı yıldız işareti (\*) içeriyor*

Bu örnekte, uygulama hangi kuyruk yöneticisinin bağlanacağı konusunda endişeli değildir. Uygulama, bir yıldız imi de dahil olmak üzere, kuyruk yöneticisi adı için bir MQCONN çağrısı yayınlar. Uygun bir kanal seçiliyor.

Uygulama sorunları:

```
MQCONN (*SALE)
```

Kurallardan sonra, bu örnekte olan budur:

1. The client channel definition table (CCDT) is scanned for the queue manager name SALE, matching with the application MQCONN call.
2. ALPHA ve BETA için kanal tanımları bulundu.
3. Bir kanalda CLNTWGHT değeri 0 ise, bu kanal seçilidir. If both have a CLNTWGHT value of 0, channel ALPHA is selected because it is first in alphabetical sequence. Her iki kanalda da sıfır olmayan CLNTWGHT değeri varsa, ağırlıklandırmasına dayalı olarak bir kanal rasgele seçilir.
4. Kanalı başlatma girişimi yapıldı.
5. If channel BETA was selected, the attempt to start it is successful.
6. ALPHA kanalı seçildiyse, iletişim bağlantısı bozuk olduğu için, bu kanal başlatılmaya çalışıldı. Aşağıdaki adımlar geçerli olur:
  - a. The only other channel for the queue manager name SALE is BETA.
  - b. Bu kanalı başlatma girişimi yapıldı; bu işlem başarılı oldu.
7. Bir dinleyicinin çalıştığını görmek için bir denetim, çalışmakta olan bir kişi olduğunu gösterir. Bu, SALE kuyruk yöneticisine bağlı değildir; ancak, MQI çağrısı parametresinin içinde bir yıldız (\*) işareti bulunduğundan, herhangi bir denetim yapılmamıştır. Uygulama, SALE\_BACKUP kuyruk yöneticisine bağlı 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 seçiliyor.

The application requires a connection to a specific queue manager, named SALE, as seen in the MQI call:

```
MQCONN (SALE)
```

Kurallardan sonra, bu örnekte olan budur:

1. İstemci kanal tanımlama çizelgesi (CCDT), uygulama MQCONN çağrısıyla eşleşen SALEkuyruk yöneticisi adı için, alfabetik kanal adı sırasından taranır.
2. Eşleşmeyi içeren ilk kanal tanımlaması ALPHA.
3. Kanalı başlatma girişimi yapıldı-iletişim bağlantısı bozuk olduğu için bu işlem başarılı değil.
4. The client channel definition table is again scanned for the queue manager name SALE and the channel name BETA is found.
5. Kanalı başlatma girişimi yapıldı; bu işlem başarılı oldu.
6. Bir dinleyicinin çalıştığını görmek için bir denetim, çalışmakta olan bir kişi olduğunu, ancak SALE kuyruk yöneticisine bağlı olmadığını gösterir.
7. İstemci kanal tanımlama çizelgesinde başka giriş yok. Uygulama devam edemiyor ve MQRC\_Q\_MGR\_NOT\_AVAM dönüş kodunu alıyor.

*Örnek 3. Kuyruk yöneticisi adı boş ya da yıldız işareti (\*)*

Bu örnekte, uygulama hangi kuyruk yöneticisinin bağlanacağı konusunda endişeli değildir. Uygulama, boş bir kuyruk yöneticisi adı ya da yıldız işareti belirten bir MQCONN ile ilgili sorunları içerir. Uygun bir kanal seçiliyor.

Bu işlem, "[Örnek 1. Kuyruk yöneticisi adı yıldız işareti \(\\*\) içeriyor](#)" sayfa 962ile aynı şekilde ele alınır.

**Not:** Bu uygulama IBM MQ MQI clientdışında bir ortamda çalışıyorsa ve ad boş bırakıldıysa, varsayılan kuyruk yöneticisine bağlanma girişiminde bulunulacaktır. Bu, istemci ortamından çalıştırıldığı zaman değil; erişilen kuyruk yöneticisi, kanalın bağlandığı dinleyiciyle ilişkili olan bir yöneticidir.

Uygulama sorunları:

```
MQCONN (" ")
```

ya da

```
MQCONN (*)
```

Kurallardan sonra, bu örnekte olan budur:

1. İstemci kanal tanımlama çizelgesi (CCDT), boş olan bir kuyruk yöneticisi adı için, MQCONN çağrısıyla eşleşen bir kuyruk yöneticisi adı için, alfabetik kanal adı sıralamasında 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 giriş, BETAkanal adı içindir.
4. Tanımlamadaki queue manager name değeri SALE. Bu durum, bir kez daha, kuyruk yöneticisi adının boş olmasını gerektiren MQCONN çağrı parametresiyle eşleşmiyor.
5. İstemci kanal tanımlama çizelgesinde başka giriş yok. Uygulama devam edemiyor ve MQRC\_Q\_MGR\_NOT\_AVAM dönüş kodunu alıyor.

### ***İstemci ortamında tetikleme***

IBM MQ MQI clients üzerinde çalışan IBM MQ uygulamaları tarafından gönderilen iletiler, diğer tüm iletilerle aynı şekilde tetiklemeye katkıda bulunur ve bu iletiler hem sunucudaki, hem de istemcide programları tetiklemek için kullanılabilir.

Tetikleme, Tetikleme kanalları içinde ayrıntılı olarak açıklanmaktadır.

Tetikleme izleme programı ve başlatılacak uygulamanın aynı sistemde olması gerekir.

Tetiklenen kuyruğun varsayılan özellikleri, sunucu ortamındaki özelliklerle aynıdır. Özellikle, bir z/OS kuyruk yöneticisinde yerel olan tetiklenen bir kuyruğa ileti koyan bir istemci uygulamasında MQPMO eşitleme noktası denetim seçeneği belirtilmediyse, iletiler bir iş birimi içinde yerleştirilir. Daha sonra tetikleme koşulu karşılanırsa, tetikleme iletileri aynı iş birimi içindeki başlatma kuyruğuna yerleştirilir ve iş birimi sona erinceye kadar tetikleyici izleme programı tarafından alınmaz. Tetiklenecek işlem, iş birimi sona erdirilinceye kadar başlatılmaz.

#### Süreç tanımlaması

Bu işlem, tetiklemeyi tetikleyen kuyrukla ilişkili olduğundan, sunucuda süreç tanımlamasını tanımlamalısınız.

Tetiklenecek olanları süreç nesnesi tanımlar. İstemci ve sunucu aynı altyapıda çalışmıyorsa, tetikleme izleyicisinin başlattığı tüm işlemler *AppLT*petanımlamaladır; tersi durumda, sunucu varsayılan tanımlarını (yani, normalde sunucu makinesiyle ilişkili uygulama tipi) alır ve bir hataya neden olur.

Örneğin, tetikleme izleme programı bir Windows istemcisinde çalışıyorsa ve başka bir işletim sisteminde bir sunucuya istek göndermek istiyorsa, diğer bir işletim sistemi varsayılan tanımlamalarını kullanır ve işlem başarısız olur.

#### Tetikleyici İzleme Programı

The trigger monitor provided by non-z/OS IBM MQ products runs in the client environments for

**IBM i** IBM i, UNIX, Linux, and Windows systems.

Tetikleme izleyicisini çalıştırmak için şu komutlardan birini çalıştırın:

- **IBM i** IBM i'ta:

```
CALL PGM(QMQM/RUNMQTMC) PARM('-m' QmgrName '-q' InitQ)
```

- **ULW** Windows, UNIX and Linux platformlarında:

```
runmqmmc [-m QMgrName] [-q InitQ]
```

Varsayılan başlatma kuyruğu SYSTEM.DEFAULT.INITIATION.QUEUE varsayılan kuyruk yöneticisinde KUYRUK. Başlatma kuyruğu, tetikleme izleyicinin tetikleme iletilerini göreceği yerdir. Daha sonra, uygun tetikleyici iletilerine ilişkin programları çağırır. Bu tetikleme izleme programı varsayılan uygulama tipini destekler ve istemci kitaplıklarını ilişkilendirmesi dışında, `runmqtrm` ile aynı olur.

Tetikleme izleme programı tarafından oluşturulan komut dizilimi aşağıdaki gibidir:

1. *ApplicId* , ilgili süreç tanımlamasından. *ApplicId* , çalıştırılacak programın adıdır; komut satırına girilir.
2. Başlangıç kuyruğundan elde edilen tırnak işaretleri içine alınmış MQTMC2 yapısı. Sistem komutunun tek bir parametre olarak kabul ettiği sırayla, tam olarak sağlandığı şekilde, bu dizeye sahip bir komut dizisi başlatılır.
3. *EnvrData* , ilgili süreç tanımlamasından.

Tetikleme izleme programı, başlatma kuyruğunda başlatılmış olan uygulamanın tamamlanmasına kadar başka bir ileti olup olmadığını denetleyemez. Uygulamanın yapması gereken çok işlem varsa, tetikleme izleme programı gelen tetikleyici ileti sayısına yetişmeyebilir. Bu durumla başa çıkmak için iki yol vardır:

1. Çalışan daha fazla tetikleyici izleme programı var

Daha fazla tetikleme izleme programının çalıştığını seçerseniz, herhangi bir zamanda çalışabilecek uygulama sayısı üst sınırını denetleyebilirsiniz.

2. Başlatılan uygulamaları arka planda çalıştır

Uygulamaları arka planda çalıştırabilir IBM MQ , çalışabilecek uygulama sayısı üzerinde herhangi bir kısıtlama oluşturmaz.

To run the started application in the background on UNIX and Linux systems, you must put an & (ampersand) at the end of the *EnvrData* of the process definition.

#### *CICS uygulamaları (z/OS dışı)*

Bir MQCONN ya da MQCONNX çağrısı içeren birz/OS dışıCICS uygulama programı CEDA olarak CEDA olarak tanımlanmalıdır. Bir CICS sunucusu uygulamasını istemci olarak yeniden bağlarsanız, eşitleme noktası desteğini kaybetmeyi riske atınız.

Bir MQCONN ya da MQCONNX çağrısı içeren birz/OS dışıCICS uygulama programı CEDA olarak CEDA olarak tanımlanmalıdır. Yerleşik kodu mümkün olduğunca küçük yapmak için, MQCONN ya da MQCONNX çağrısını vermek için ayrı bir programa bağlantı yapabilirsiniz.

MQSERVER ortam değişkeni istemci bağlantısını tanımlamak için kullanıldıysa, bu değişkenin CICSENV.CMD dosyası.

IBM MQ uygulamaları, bir IBM MQ sunucusu ortamında ya da kod değiştirmeden bir IBM MQ istemcisinde çalıştırılabilir. However, in an IBM MQ server environment, CICS can act as sync point coordinator, and you use EXEC CICS SYNCPOINT and EXEC CICS SYNCPOINT ROLLBACK rather than **MQCMIT** and **MQBACK**. Bir CICS uygulaması istemci olarak yeniden bağlantılandırılırsa, eşitleme noktası desteği kaybedilir. IBM MQ MQI client üzerinde çalışan uygulama için **MQCMIT** ve **MQBACK** kullanılmalıdır.

## **Windows** **UNIX** **CICS ve Tuxedo uygulamalarının hazırlanması ve çalıştırılması**

CICS ve Tuxedo uygulamalarını istemci uygulamaları olarak çalıştırmak için, sunucu uygulamalarıyla kullandığınız farklı kitaplıkları kullanıyorsunuz demektir. Uygulamanın çalıştırıldığı kullanıcı kimliği de farklıdır.

CICS ve Tuxedo uygulamalarını IBM MQ MQI client uygulamaları olarak çalıştırmak üzere hazırlamak için [Genişletilmiş işlem istemcisinin yapılandırılması](#) başlıklı konu yönergelerinde yer alan yönergeleri izleyin.

Note, however, that the information that deals specifically with preparing CICS and Tuxedo applications, including the sample programs supplied with IBM MQ, assumes that you are preparing applications to run on an IBM MQ server system. Sonuç olarak, bilgi yalnızca, bir sunucu sisteminde kullanılmak üzere hazırlanmış olan IBM MQ kitaplıklarına gönderme yapar. İstemci uygulamalarınızı hazırlarken aşağıdaki şeyleri yapmanız gerekir:

- Uygulamanızın kullandığı dil bağlamaları için uygun istemci sistemi kitaplığını kullanın. Örneğin:
  - **UNIX** For applications written in C on UNIX, use the library libmqic instead of libmqm.
  - **Windows** Windows sistemlerinde, mqm.lib yerine mqic.lib kitaplığını kullanın.
- Çizelge 138 sayfa 965 ve Çizelge 139 sayfa 965' de gösterilen sunucu sistemi kitaplıkları yerine eşdeğer istemci sistemi kitaplıklarını kullanın. Bu çizelgelerde listelenmeyen bir sunucu sistemi kitaplığı varsa, istemci sisteminde aynı kitaplığı kullanın.

<i>Çizelge 138. UNIX üzerindeki istemci sistemi kitaplıkları</i>	
<b>IBM MQ sunucu sistemine ilişkin kitaplık</b>	<b>Bir IBM MQ istemci sisteminde kullanılacak eşdeğer kitaplık</b>
libmqmxa	libmqcxa

<i>Çizelge 139. Windows sistemlerindeki istemci sistemi kitaplıkları</i>	
<b>IBM MQ sunucu sistemine ilişkin kitaplık</b>	<b>Bir IBM MQ istemci sisteminde kullanılacak eşdeğer kitaplık</b>
mqmxa.lib	mqcxa.lib
mqmtux.lib	mqcxa.lib

Çizelge 139. Windows sistemlerindeki istemci sistemi kitaplıkları (devamı var)

IBM MQ sunucu sistemine ilişkin kitaplık	Bir IBM MQ istemci sisteminde kullanılacak eşdeğer kitaplık
mqmenc.lib	mqcx.lib
mqmcics4.lib	mqccics4.lib

### İstemci uygulaması tarafından kullanılan kullanıcı kimliği

When you run an IBM MQ server application under CICS, it normally switches from the CICS user to the user ID of the transaction. Ancak, CICS altında bir IBM MQ MQI client uygulaması çalıştırdığınızda, CICS ' in ayrıcalıklı yetkisi korunur.

### Windows > UNIX CICS ve Tuxedo örnek programları

UNIX ve Windows sistemlerinde kullanılmak üzere CICS ve Tuxedo örnek programları.

Çizelge 140 sayfa 966 , UNIX istemci sistemlerinde kullanılmak üzere sağlanan CICS ve Tuxedo örnek programlarını listeler. Çizelge 141 sayfa 966 , Windows istemci sistemlerine ilişkin eşdeğer bilgileri listeler. Çizelgeler ayrıca, programları hazırlamak ve çalıştırmak için kullanılan dosyaları da listeler. Örnek programların bir açıklaması için bkz. “CICS hareket örneği” sayfa 1113 ve “UNIX ve Windows üzerinde SMOKIN örneklerini kullanma” sayfa 1156.

Çizelge 140. UNIX istemci sistemleri için örnek programlar

Tanım	Kaynak	Yürütülebilir modül
CICS PROGRAM	amqscic0.ccs	amqscicc
CICS programına ilişkin üstbilgi dosyası	amqscih0.h	-
İletileri koymak için smokin istemci programı	amqstxpx.c	-
İletileri almak için smokin istemci programı	amqstxgx.c	-
İki istemci programı için smokin sunucu programı	amqstxsx.c	-
Tuxedo programları için UBBCONFIG dosyası	ubbstxcx.cfg	-
Tuxedo programlarına ilişkin alan tablosu dosyası	amqstxvx.flds	-
Tuxedo programlarına ilişkin açıklama dosyasını görüntüle	amqstxvx.v	-

Çizelge 141. Windows istemci sistemleri için örnek programlar

Tanım	Kaynak	Yürütülebilir modül
CICS Hareket	amqscic0.ccs	amqscicc
CICS işlemine ilişkin üstbilgi dosyası	amqscih0.h	-
İletileri koymak için smokin istemci programı	amqstxpx.c	-
İletileri almak için smokin istemci programı	amqstxgx.c	-
İki istemci programı için smokin sunucu programı	amqstxsx.c	-
Tuxedo programları için UBBCONFIG dosyası	ubbstxcx.cfg	-
Tuxedo programlarına ilişkin alan tablosu dosyası	amqstxvx.fld	-
Tuxedo programlarına ilişkin açıklama dosyasını görüntüle	amqstxvx.v	-

Çizelge 141. Windows istemci sistemleri için örnek programlar (devamı var)

Tanım	Kaynak	Yürütülebilir modül
Tuxedo programları için makefile	amqstxmc.mak	-
Tuxedo programlarına ilişkin ENVFILE dosyası	amqstxen.env	-

Windows

UNIX

## Error message AMQ5203, as modified for CICS and Tuxedo

### applications

Genişletilmiş bir işlemsel istemciyi kullanan CICS ya da Tuxedo uygulamalarını çalıştırdığınızda, standart tanılama iletilerini görebilirsiniz. Bunlardan biri, genişletilmiş bir işlemsel istemciyle kullanılmak üzere değiştirildi.

IBM MQ hata günlüğü dosyalarında görebileceğiniz iletiler [Tanılama iletilerinde](#):

[AMQ4000-9999](#)belgesinde belgelenir. Message AMQ5203 has been modified for use with an extended transactional client. Değiştirilen iletilerin metni şöyledir:

## AMQ5203: XA arabirimi çağrılırken bir hata oluştu.

### Açıklama

Hata numarası & 2; 1 değeri, verilen işaret değerinin & 1 değerinin geçersiz olduğunu, 2 aynı işlemde iş parçacıklı ve iş parçacıklı olmayan kitaplıkları kullanma girişiminde bulunulduğunu, 3 ise, belirtilen kuyruk yöneticisi adı '& 3' ile ilgili bir hata olduğunu, 4, & 1 kaynak yöneticisi tanıtıcısının geçersiz olduğunu, 5 ise ikinci bir kuyruk yöneticisini kullanmak için girişimde bulunulduğunu gösterir.& 3 'başka bir kuyruk yöneticisi önceden bağlandığında, 6, uygulama bir kuyruk yöneticisine bağlı olmadığı halde Transaction Manager 'ın çağrıldığını, 7, başka bir çağrı devam ederken XA çağrısının yapıldığını, 8 ise xa\_open çağrısında' & 4 'xa\_info dizgisinin' & 5 'parametre adı için geçersiz bir parametre değeri içerdiğini, 9 ise xa\_open çağrısındaki' & 4 'xa\_info dizgisinin gerekli bir parametre,' & 5 ' parametre adı eksik olduğunu gösterir.

### Kullanıcı yanıtı

Hatayı düzeltin ve işlemi yeniden deneyin.

Windows

## Microsoft Transaction Server uygulamalarının hazırlanması ve çalıştırılması

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

IBM MQ kaynaklarına erişen Microsoft Transaction Server (MTS) uygulamalarının nasıl geliştirileceği hakkında genel bilgi edinmek için IBM MQ Yardım Merkezi 'nde MTS ' nin (MTS) bölümüne bakın.

Bir MTS uygulamasını IBM MQ MQI client uygulaması olarak çalıştırmak üzere hazırlamak için, uygulamanın her bileşeni için aşağıdakilerden birini yapın:

- Bileşen, MQI için C dili bağ tanımlarını kullanıyorsa, "[C programlarını Windows içinde hazırlama](#)" sayfa 1052 içindeki yönergeleri izleyin, ancak bileşeni mqic.libyerine mqicxa.lib kitaplığıyla bağlantılayın.
- Bileşen IBM MQ C++ sınıflarını kullanıyorsa, "[Building C++ programs on Windows](#)" sayfa 523 içindeki yönergeleri izleyin, ancak bileşeni imqc23vn.libyerine imqx23vn.lib kitaplığıyla bağlantılayın.
- Bileşen, MQI için Visual Basic dili bağ tanımlarını kullanıyorsa, "[Preparing Visual Basic programs in Windows](#)" sayfa 1056 içindeki yönergeleri izleyin, ancak Visual Basic projesini tanımlarken **Koşullu Derleme Bağımsız Değişkenleri** alanında MqType=3 yazın.
- If the component uses the IBM MQ Automation Classes for ActiveX (MQAX), define an environment variable, GMQ\_MQ\_LIB, with the value mqic32xa.dll.

Ortam değişkenini uygulamanızın içinden tanımlayabilir ya da kapsamı sistem çapında olacak şekilde tanımlayabilirsiniz. Ancak, sistemi geniş olarak tanımlamak, var olan MQAX uygulamasının, uygulama içinden ortam değişkenini tanımlamamasına, yanlış şekilde davranmasına neden olabilir.

## IBM MQ JMS uygulamalarının hazırlanması ve çalıştırılması

You can run IBM MQ JMS applications in client mode, with WebSphere Application Server as your transaction manager. Bazı uyarı iletileri görebilirsiniz.

IBM MQ JMS uygulamalarını istemci kipinde hazırlamak ve çalıştırmak için, hareket yöneticiniz olarak WebSphere Application Server ile [“kullanmaIBM MQ classes for JMS” sayfa 77](#)ile ilgili yönergeleri izleyin.

Bir IBM MQ JMS istemci uygulamasını çalıştırdığınızda, aşağıdaki uyarı iletilerini görebilirsiniz:

### MQJE080

Yetersiz lisans birimi-setmqcap komutunu çalıştırın

### MQJE081

Lisans birimi bilgilerini içeren dosya yanlış biçimde, setmqcap komutunu çalıştırın.

### MQJE082

Lisans birimi bilgilerini içeren dosya bulunamadı-setmqcap dosyasını çalıştırın

## Kullanıcı çıkışları, API çıkışları ve IBM MQ kurulabilir hizmetleri

Bu konu, bu programların kullanılmasıyla ve geliştirilmesiyle ilgili bilgilere bağlantılar içerir.

Kuyruk yöneticisi olanaklarını genişletmek için kullanıcı çıkışlarını, API çıkışlarını ve kurulabilir hizmetleri nasıl kullanabileceğiyle ilgili bir giriş için bkz. [Extending queue Manager olanakları](#).

Çıkışların ve kurulabilen hizmetlerin yazılması ve derlenmesine ilişkin bilgi edinmek için alt başlıklara bakın.

### İlgili kavramlar

[MQI kanallarına ilişkin kanal-çıkış programları](#)

### İlgili başvurular

[API çıkış başvurusu](#)

[Kurulabilir hizmetler arabirimi başvuru bilgileri](#)

 [Installable services interface reference information on IBM i](#)

### **Writing exits and installable services on UNIX, Linux and Windows**

UNIX, Linux ve Windows üzerindeki herhangi bir IBM MQ kitaplıklarına bağlanmadan çıkışlar yazabilir ve derleyebilirsiniz.

### Bu görev hakkında

Bu konu yalnızca UNIX, Linux, and Windows sistemleri için geçerlidir. Diğer platformlara ilişkin çıkışlar ve kurulabilir hizmetlerle ilgili ayrıntılar için ilgili platforma özgü konulara bakın.

IBM MQ varsayılan olmayan bir konuma kurulduysa, tüm IBM MQ kitaplıklarına bağlanmadan çıkışlarınızı yazmanız ve derlemeniz gerekir.

UNIX, Linux, and Windows sistemlerinde, bu IBM MQ kitaplıklarından herhangi birini bağlantılandırmadan çıkışlar yazabilir ve bunları derleyebilirsiniz:

- mqmzf
- mqm
- mqmvx
- mqmvxd
- mqic
- mqutl

Existing exits that are linked to these libraries continue to work, providing that on UNIX and Linux systems IBM MQ is installed in the default location.



## Yordam

1. cmqec.h üstbilgi dosyasını ekleyin.

Bu üstbilgi dosyası otomatik olarak cmqc.h, cmqxc.h ve cmqzc.h üstbilgi kütüklerini içerir.

2. Çıkışı, MQI ve DCI çağrılarının MQIEP yapısıyla yapıp yapılmaması için yazın. MQIEP yapısıyla ilgili ek bilgi için [MQIEP yapısına](#) başlıklı konuya bakın.

- Kurulabilir hizmetler
  - MQZEP çağrısını göstermek için **Hconfig** parametresini kullanın.
  - **Hconfig** parametresini kullanmadan önce, **Hconfig** ' un ilk 4 baytının MQIEP yapısının **StrucId** ile eşleşmesini denetlemeniz gerekir.
  - Kurulabilir hizmet bileşenleri yazılmasıyla ilgili ek bilgi için [MQIEP](#) başlıklı konuya bakın.
- API çıkışları
  - MQXEP çağrısını göstermek için **Hconfig** parametresini kullanın.
  - **Hconfig** parametresini kullanmadan önce, **Hconfig** ' un ilk 4 baytının MQIEP yapısının **StrucId** ile eşleşmesini denetlemeniz gerekir.
  - API çıkışlarını yazma hakkında daha fazla bilgi için bkz. [“API çıkışları yazılıyor”](#) sayfa 985.
- Kanal çıkışları
  - MQI ve DCI çağrılarını işaret etmek için MQCXP yapısının **pEntryPoints** parametresini kullanın.
  - **pEntryPoints** kullanılmadan önce MQCXP sürüm numarasının sürüm 8 ya da daha yüksek bir sürüm olduğunu doğrulayın.
  - Kanal çıkışlarını yazma hakkında daha fazla bilgi için bkz. [“Kanal-çıkış programları yazılıyor”](#) sayfa 996.
- Veri dönüştürme çıkışları
  - MQI ve DCI çağrılarını işaret etmek için MQDXP yapısının **pEntryPoints** parametresini kullanın.
  - **pEntryPoints** kullanılmadan önce MQDXP sürüm numarasının sürüm 2 ya da daha yüksek bir sürüm olduğunu doğrulayın.
  - You can use the **crtmqcvx** command and the amqsvfc0.c source file to create data conversion code that uses the **pEntryPoints** parameter. Bkz. [“IBM MQ for Windows için veri dönüştürme çıkışı yazılıyor”](#) sayfa 1021 ve [“UNIX and Linux sistemlerinde IBM MQ için veri dönüştürme çıkışı yazılıyor”](#) sayfa 1018.
  - **crtmqcvx** komutu kullanılarak oluşturulan veri dönüştürme çıkışların varsa, güncellenen komutu kullanarak çıkışı yeniden oluşturmalsınız.
  - Veri dönüştürme çıkışlarını yazma hakkında daha fazla bilgi için bkz. [“Veri dönüştürme çıkışları yazılıyor”](#) sayfa 1014.
- Bağlantı öncesi çıkışlar
  - MQI ve DCI çağrılarını işaret etmek için MQNXP yapısının **pEntryPoints** parametresini kullanın.
  - **pEntryPoints** komutunu kullanmadan önce, MQNXP sürüm numarasının sürüm 2 ya da daha yüksek bir sürüm olduğunu doğrulayın.
  - Bağlantı öncesi çıkışlar yazma hakkında daha fazla bilgi için bkz. [“Bir havuzdaki bağlanma öncesi çıkışı kullanarak bağlantı tanımlarına gönderme yapılması”](#) sayfa 1023.
- Çıkışları yayınla
  - MQI ve DCI çağrılarını işaret etmek için MQPSXP yapısının **pEntryPoints** parametresini kullanın.
  - You must check that the MQPSXP version number is at version 2 or higher before using **pEntryPoints**.
  - Yayınlama çıkışlarını yazma hakkında daha fazla bilgi için bkz. [“Yayınlama çıkışlarının yazılması ve derlenmesi”](#) sayfa 1025.
- Küme iş yükü çıkışları

- MQWXP yapısının **pEntryPoints** parametresini kullanarak MQXCLWLN çağrılarını işaret edin.
- **pEntryPoints** komutunu kullanmadan önce, MQWXP sürüm numarasının sürüm 4 ya da daha yüksek bir sürüm olduğunu doğrulayın.
- Küme iş yükü çıkışları yazma hakkında daha fazla bilgi için bkz. [“Küme iş yükü çıkışlarının yazılması ve derlenmesi” sayfa 1027.](#)

Örneğin, bir kanal çıkışında MQPUT çağrılıyor:

```
pChannelExitParms -> pEntryPoints -> MQPUT_Call(pChannelExitParms -> Hconn,
                                                Hobj,
                                                &md,
                                                &pmo,
                                                messlen,
                                                buffer,
                                                &CompCode,
                                                &Reason);
```

[“IBM MQ örnek yordamsal programlarının kullanılması” sayfa 1092](#) içinde başka örnekler de görülebilir.

### 3. Çıkışı derleyin:

- IBM MQ kitaplıklarına bağlanmayın.
- Çıkışınızdaki IBM MQ kitaplıklarına gömülü bir RPath eklemeyin.
- Çıkışınızı derlemeye ilişkin ek bilgi için aşağıdaki başlıklara bakın:
  - API çıkışları: [“API çıkışları derleniyor” sayfa 987.](#)
  - Kanal çıkışları, yayınlama çıkışları, Küme iş yükü çıkışları: [“Compiling channel exit programs on Windows, UNIX and Linux systems” sayfa 1012.](#)
  - Veri dönüştürme çıkışları: [“Veri dönüştürme çıkışları yazılıyor” sayfa 1014.](#)

### 4. Çıkışı aşağıdaki yerlerden birine koyun:

- Çıkışı yapılandırırken tam olarak nitelendirildiğiniz bir seçim yolu
- Belirli bir kuruluş dizininde 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. Ek bilgi için [Çıkış yolubaşlıklı](#) konuya bakın. Windows ve Linux üzerinde, yolu değiştirmek için IBM MQ Gezgini 'ni kullanabilirsiniz:
  - Kuyruk yöneticisi adını sağ tıklayın
  - Özellikler ...** düğmesini tıklayın.
  - Dahili'** yi tıklayın.
  - Çıkışlar varsayılan yolu alanında, çıkış programını tutan dizinin yol adını belirtin.

Bir çıkış hem belirli bir kuruluş dizinine, hem de varsayılan yol dizinine yerleştirilirse, belirli bir kuruluş dizini çıkışı, yolda adı belirtilen IBM MQ kuruluşu tarafından kullanılır. For example, the exit is placed in `/exits/installation2` and in `/exits`, but not in `/exits/installation1`. The IBM MQ installation `installation2` uses the exit from `/exits/installation2`. The IBM MQ installation `installation1` uses the exit from the `/exits` directory.

### 5. Gerekliyse, çıkışı yapılandırın:

- Kurulabilir hizmetler: [“Hizmetlerin ve bileşenlerin yapılandırılması” sayfa 978.](#)
- API çıkışları: [“API çıkışlarını yapılandırma” sayfa 990.](#)
- Kanal çıkışları: [“Kanal çıkışlarının yapılandırılması” sayfa 1013.](#)
- Çıkışlar yayınlayın: [“Yayınlama çıkışlarının yapılandırılması” sayfa 1026.](#)
- Bağlantı öncesi çıkışlar: [İstemci yapılandırma dosyasının PreConnect kısmı.](#)

## API çıkışları bir MQI kitaplığı ile bağlantılandırılmadı

Belirli koşullar altında, var olan API çıkışınızı, bir IBM MQ API kitaplığı ile birlikte MQIEP işlev göstergeleriyle yeniden kodlanamayacak şekilde bağlamalısınız.

Bu, var olan API çıkışınızın başarıyla yüklenebilmesi için, sisteminizin yürütme ortamı bağlantı oluşturucusu tarafından, işlev işaretlerinin yüklü olmadığı programlara başarıyla yüklenebilmesini sağlar.

**Not:** Bu bilgiler, doğrudan MQI çağrılarını yapan var olan API çıkışlarıyla sınırlıdır. That is, those exits that do not use , MQIEP. Mümkün olduğunda, bunun yerine MQIEP giriş noktalarını kullanmak için çıkışı yeniden kodlamayı planlamanız gerekir.

IBM MQ 8.0, **runmqsc** , doğrudan bir MQI kitaplığı ile bağlanmayan bir program örneğidir.

Bu nedenle, gerekli IBM MQ API kitaplığı ile bağlantısı olmayan ya da MQIEP 'yi kullanacak şekilde yeniden kodlanan bir API çıkışı, **runmqsc** ' e yüklenemez.

You see errors in the queue manager error log, for example, AMQ6175: Sistem, paylaşılan kitaplığı dinamik olarak yükleyemedi., together with qualifying text such as undefined symbol: MQCONN.

ve AMQ7214: API çıkışı 'mixitname' için modül yüklenemedi.

### İlgili görevler

“Writing exits and installable services on UNIX, Linux and Windows” sayfa 968

UNIX, Linux ve Windows üzerindeki herhangi bir IBM MQ kitaplıklarına bağlanmadan çıkışlar yazabilir ve derleyebilirsiniz.

## UNIX, Linux ve 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 bileşen sağlayabilmesi için belgelenmiş olabilir.

IBM MQ kurulabilir hizmetlerini sağlamanın başlıca nedenleri şunlardır:

- IBM MQ ürünleri tarafından sağlanan bileşenleri kullanıp kullanmamayı seçmenin esnekliğini sağlamak için bunları başkalarıyla değiştirin ya da bunları değiştirin.
- Satıcıların katılmasına izin vermek için, IBM MQ ürünlerinde iç değişiklikler yapmadan, yeni teknolojiler kullanabilecek bileşenler sunar.
- IBM MQ ' in yeni teknolojileri daha hızlı ve daha ucuz bir şekilde kullanmalarına izin vermek ve bu nedenle ürünleri daha önce ve daha düşük fiyatlarla sağlamak.

*Kurulabilir hizmetler ve hizmet bileşenleri* , IBM MQ ürün yapısının bir parçasıdır. Bu yapının merkezinde, kuyruk yöneticisinin, Message Queue Interface (İleti Kuyruğu Arabirimi) ile ilişkili işlevi ve kuralları uygulayan bölümü yer alıyor. 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 fazla *hizmet bileşeni* kullanılarak uygulanan bir ilgili işlev kümesidir. Her bir bileşen, düzgün bir şekilde tasarlanmış, genel kullanıma açık bir arabirim kullanılarak çağrılır. This enables independent software vendors and other third parties to provide installable components to augment or replace those provided by the IBM MQ products. [Çizelge 142 sayfa 972](#) , kullanılabilecek hizmetleri ve bileşenleri özetler.

<i>Çizelge 142. Kurulabilir hizmet bileşenleri özeti</i>			
<b>Kurulabilir hizmet</b>	<b>Sağlanan bileşen</b>	<b>İşlev</b>	<b>Gereksinmeler</b>
Yetkilendirme hizmeti	nesne yetkisi yöneticisi (OAM)	Komutlara ve MQI çağrılarına ilişkin yetki denetimi sağlar. Kullanıcılar OAM ' yi büyütme ya da değiştirmek için kendi bileşenlerinden yazabilir.  Örneğin, bir kullanıcı kimliğinin kuyruğu açma yetkisi olup olmadığını denetlemek için.	(Uygun platform yetkilendirme olanakları varsayılar)
Ad hizmeti	Yok	Kuyruk yöneticisine, belirlenen bir kuyruğa sahip olan kuyruk yöneticisinin adını aramak için destek sağlar.  • Kullanıcı tanımlı	• Üçüncü taraf ya da kullanıcı tarafından yazılan bir ad yöneticisi

Kurulabilir hizmetler arabirimi, [Kurulabilir hizmetler arabirimi başvuru bilgileri](#) içinde açıklanmıştır.

### **İlgili görevler**

[Kurulabilir hizmetlerin yapılandırılması](#)

### **Hizmet bileşeni yazılması**

Bu bölümde hizmetler, bileşenler, giriş noktaları ve dönüş kodları arasındaki ilişki anlatılır.

### **İşlevler ve bileşenler**

Her hizmet, bir dizi ilgili işlevden oluşur. Örneğin, ad hizmeti aşağıdakiler için işlev içerir:

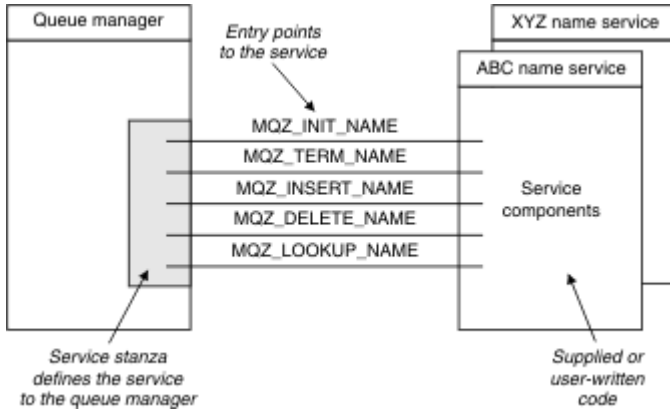
- Bir kuyruk adı aranır ve kuyruğun tanımlandığı kuyruk yöneticisinin adını döndürür.
- Hizmet dizinine kuyruk adı eklenmesi
- Bir kuyruk adının hizmet dizininden silinmesi

Ayrıca, başlatma ve sonlandırma işlevlerini de içerir.

Kurulabilir bir hizmet, bir ya da daha çok hizmet bileşeni tarafından sağlanır. Her bileşen, söz edilen hizmet için tanımlanmış işlevlerin bazılarını ya da tümünü gerçekleştirebilir. Örneğin, IBM MQ for AIX' ta sağlanan yetkilendirme hizmeti bileşeni OAM, kullanılabilir tüm işlevleri gerçekleştirir. Ek bilgi için "[Yetkilendirme hizmeti arabirimi](#)" sayfa 975 başlıklı konuya bakın. Bu bileşen ayrıca, hizmetin uygulanması için gereken temel kaynakları ya da yazılımları (örneğin, LDAP dizini) yönetmekten de sorumludur. Yapılandırma dosyaları, bileşeni yüklemek ve sağladığı işlevsel yordamların adreslerini belirlemek için standart bir yol sağlar.

[Şekil 104 sayfa 973](#) , hizmetlerin ve bileşenlerin nasıl ilgili olduğunu gösterir:

- Bir hizmet, bir yapılandırma dosyasındaki stanzas tarafından kuyruk yöneticisinde tanımlanır.
- Her hizmet, kuyruk yöneticisinde sağlanan kodla desteklenir. Kullanıcılar bu kodu değiştiremez ve bu nedenle kendi hizmetlerini oluşturamaz.
- Her hizmet bir ya da daha fazla bileşen tarafından uygulanır; bunlar ürünle birlikte ya da kullanıcı tarafından yazılmış olabilir. Bir hizmet için birden çok bileşen çağrılabilir, her biri hizmet içinde farklı tesisleri destekleyebilir.
- Giriş noktaları, hizmet bileşenlerini kuyruk yöneticisinde destekleyici kodlara bağlar.



Şekil 104. Hizmetlerin, bileşenlerin ve giriş noktalarının anlaşılması

## 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 yapılandırıldığı sırada sipariş edilmesi, hizmete ilişkin bir isteği karşılamaya yönelik giriş-noktaların çağrıldığı sırayı tanımlar.

Sağlanan cmqzc.hüstbilgi dosyasında, sağlanan giriş noktalarının her bir hizmete ilişkin bir MQZID\_ öneki var.

Hizmetler mevcutsa, hizmetler önceden tanımlanmış bir sırayla yüklenir. Aşağıdaki listede hizmetler ve bunların kullanıma hazırlandıkları sıra gösterilir.

1. NameService
2. AuthorizationService
3. UserIdentifierService

AuthorizationService , varsayılan olarak yapılandırılan tek hizmettir. NameService ve UserIdentifierService 'yi kullanmak istiyorsanız el ile yapılandırın.

Hizmetler ve hizmet bileşenleri bire bir ya da bire bir eşlemeye sahiptir. Her hizmet için birden çok hizmet bileşeni tanımlanabilir. On UNIX and Linux systems, the ServiceComponent stanza's Service value must match the Service stanza's Name value in the qm.ini file.

Windows'ta, ServiceComponent 'un Hizmet kaydı anahtar değeri, Ad kayıt dosyası anahtar 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.

UNIX and Linux sistemleri için, hizmet bileşenleri qm.ini dosyasında tanımlandıkları sırayla başlatılır. Windows'üzerinde, Windows kaydı kullanıldığı için, IBM MQ alfabetik sırada değerleri döndüren bir **RegEnumKey** çağrısı yayınlar. Bu nedenle, Windows üzerinde hizmetler, kayıta tanımlandıkça, alfabetik sırayla çağrılır.

ServiceComponent tanımlarının sıralaması önemlidir. Bu sıralama, belirli bir hizmet için bileşenlerin çalıştırılmasına ilişkin 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 belirtilmediyse, isteği işlemek için alfabetik sırada karşılaşılan ilk bileşen kullanılır ve ilgili bileşenin adı kullanılır.

## Dönüş kodları

Hizmet bileşenleri, çeşitli koşullarla ilgili raporlama yapmak üzere kuyruk yöneticisine dönüş kodları sağlar. İşlemlerin başarılı ya da başarısız olduğunu bildirirler ve kuyruk yöneticisinin bir sonraki hizmet bileşenine ilerleyip ilerlemeyeceğini belirtir. Ayrı bir *Devamı* parametresi bu göstergelyi taşır.

## Bileşen verileri

Tek bir hizmet bileşeni, verilerin çeşitli işlevleri arasında paylaşılmasını gerektirebilir. Kurulabilir hizmetler, bir hizmet bileşeninin her çağrısına geçirilmek üzere isteğe bağlı bir veri alanı sağlar. Bu veri alanı, hizmet bileşeninin dışlayıcı kullanımı içindir. Bu, farklı adres alanlarından ya da süreçlerden yapılmış olsa da, belirli bir işlevin tüm çağrıları tarafından paylaşılır. Her çağrıldığında, hizmet bileşeninden adreslenebilir bir şekilde verileceği garanti edilir. Bu alanın boyutunu *ServiceComponent* stanza içinde bildirmeniz gerekir.

### *Bileşenlerin başlatılması ve sona erdirilmesi*

Bileşen kullanıma hazırlama ve sonlandırma seçeneklerinin kullanımı.

Bileşen kullanıma hazırlama yordamı çağrıldığında, bileşen tarafından desteklenen her bir giriş noktası için kuyruk yöneticisi **MQZEP** işlevini çağırmalıdır. **MQZEP**, hizmete ilişkin bir giriş noktasını tanımlar. Tanımlanmamış tüm çıkış noktalarının boş olduğu varsayılır.

Bir bileşen, her zaman birincil kullanıma hazırlama seçeneğiyle bir kez çağrılır; bu seçenek başka bir şekilde çağrılmadan önce çağrılır.

Bazı altyapılarda ikincil kullanıma hazırlama seçeneğiyle bir bileşen çağrılabilir. Örneğin, hizmete erişildiği her işletim sistemi işlemi, iş parçacığı ya da görev için bir kez çağrılabilir.

İkincil başlatma kullanılırsa:

- İkincil kullanıma hazırlama işlemi için bileşen birden çok kez çağrılabilir. Bu tür her çağrı için, hizmet artık gerekmediği durumlarda ikincil sona erdirmeye eşleşen bir çağrı yayınlanır.

Adlandırma hizmetleri için bu, MQZ\_TERM\_NAME çağrısıdır.

Yetki hizmetleri için bu, MQZ\_TERM\_AUTHORITY çağrısıdır.

- Bileşen birincil ve ikincil kullanıma hazırlama işlemi için her çağrıldığında, giriş noktalarının yeniden belirtilmesi gerekir (MQZEP çağrılarak).
- Bileşen verilerinin yalnızca bir kopyası bileşen için kullanılır; her ikincil kullanıma hazırlama işlemi için farklı bir kopya yoktur.
- İkincil kullanıma hazırlama gerçekleştirilmeden önce, hizmetin (işletim sistemi işleminden, iş parçacığından ya da görevden uygun olduğu şekilde) başka çağrılar için çağrılmaz.
- The component must set the **Version** parameter to the same value for primary and secondary initialization.

Bileşen her zaman birincil sonlandırma seçeneği ile birlikte çağrılır ve bu seçenek artık gerekli değildir. Bu bileşen için başka arama yapılmayacak.

İkincil kullanıma hazırlama işlemi için çağrıldıysa, bileşen ikincil sonlandırma seçeneğiyle birlikte çağrılır.




### *Nesne yetkisi yöneticisi (OAM)*

IBM MQ ürünleriyle birlikte sağlanan yetkilendirme hizmeti bileşeni, Object Authority Manager (OAM) adı verilir.

Varsayılan değer olarak, OAM etkin ve denetim komutları **dspmqaout** (görüntü birimi yetkisi), **dmpmqaut** (döküm yetkisi) ve **setmqaut** (yetkiyi ayarla ya da ilk duruma getirme) komutları ile çalışır.

Bu komutların sözdizimi ve bunların nasıl kullanılacağı [Denetim komutlarını kullanan yönetimi](#)nde açıklanmıştır.

OAM, bir birincil kullanıcı ya da grubun *entite* ile çalışır:

-   UNIX and Linux sistemlerinde, bir birincil kullanıcı bir kullanıcı kimliğidir ya da bir kullanıcı adına çalışan bir uygulama programıyla ilişkilendirilmiş bir tanıtıcıdır; bir grup, sistem tarafından tanımlanan bir birincil kullanıcı derlesidir.
-  Windows sistemlerinde, bir birincil kullanıcı bir Windows kullanıcı kimliğidir ya da bir kullanıcı adına çalışan bir uygulama programıyla ilişkilendirilmiş bir tanıtıcıdır; bir grup, bir Windows grubudur.

Yetkilendirmeler, 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 yetkisinin olup olmadığını denetler ve belirtilen kuyruk yöneticisi kaynaklarına erişmeyi sağlar.

Yetkilendirme hizmeti, kendi yetkilendirme hizmeti bileşeninizi yazarak kuyruk yöneticileri için sağlanan yetki denetimini artırmanızı ya da değiştirmenizi sağlar.

#### *Ad hizmeti*

Ad hizmeti, belirlenen bir kuyruğa sahip olan kuyruk yöneticisinin adını aramak için kuyruk yöneticisine destek sağlayan kurulabilir bir hizmettir. Bir ad hizmetinden başka bir kuyruk özniteliği alınamıyor.

Ad hizmeti, bir uygulamanın çıkış için yerel kuyruklar gibi uzak kuyruklar açmasını sağlar. Kuyruklar dışındaki nesnelere için bir ad hizmeti çağrılmaz.

**Not:** The remote queues must have their **Scope** attribute set to CELL.

Bir uygulama bir kuyruğu açtığında, kuyruk yöneticisinin dizininde ilk olarak kuyruğun adını arar. Burada bulamazsa, kuyruk adını tanıyan birini buluncaya kadar, yapılandırılmış olduğu kadar çok sayıda ad hizmeti yapılandırılmıştır. Ad tanınmazsa, açma işlemi başarısız olur.

Ad hizmeti, o kuyruk için sahip olan kuyruk yöneticisini döndürür. Daha sonra kuyruk yöneticisi, özgün istekte kuyruk ve kuyruk yöneticisi adını belirtmiş gibi MQOPER isteğiyle devam eder.

Ad hizmeti arabirimi (NSI), IBM MQ çerçevesinin bir parçasıdır.

## **Ad hizmeti nasıl çalışır**

If a queue definition specifies the **Scope** attribute as queue manager, that is, SCOPE(QMGR) in MQSC, the queue definition (along with all the queue attributes) is stored in the queue manager's directory only. Bu, kurulabilir bir hizmetle değiştirilemez.

If a queue definition specifies the **Scope** attribute as cell, that is, SCOPE(CELL) in MQSC, the queue definition is again stored in the queue manager's directory, along with all the queue attributes. Ancak, kuyruk ve kuyruk yöneticisi adı aynı zamanda bir ad hizmetinde de saklanır. Bu bilgileri saklayabilen bir hizmet yoksa, *Scope* hücrelerine sahip bir kuyruk tanımlanamaz.

Bilgilerin saklanabileceği dizin, hizmet tarafından yönetilebilir ya da hizmet, bu amaçla temel bir hizmeti (örneğin, bir LDAP dizini) kullanabilir. Her iki durumda da, bileşen ve kuyruk yöneticisi belirttik olarak silininceye kadar, dizinde saklanan tanımların kalıcı olarak saklanmaması gerekir.

#### **Not:**

1. Uzak bir anasistemin yerel kuyruk tanımlamasına (CELL kapsamı ile) bir adlandırma dizini hücreindeki farklı bir kuyruk yöneticisiyle ileti göndermek için, bir kanal tanımlamanız gerekir.
2. CELL kapsamına girse bile, doğrudan uzak kuyruktan ileti alamazsınız.
3. CELL kapsamı içeren bir kuyruğa gönderilirken uzak kuyruk tanımlaması gerekmez.
4. Hedef kuyruk yöneticisi ve bir çift kanal tanımlaması için hala bir iletim kuyruğuna gereksinim duyarsanız, adlandırma hizmeti merkezi olarak hedef kuyruğu tanımlar. Bunun yanı sıra, yerel sistemdeki iletim kuyruğu, hedef kuyruğu bulandıran kuyruk yöneticisiyle aynı adı ve uzak sistemde hücre kapsamı ile aynı adı olmalıdır.

For example, if the remote queue manager has the name QM01, the transmission queue on the local system must also have the name QM01.

#### *Yetkilendirme hizmeti arabirimi*

Yetki hizmeti, kuyruk yöneticisi tarafından kullanılmak üzere giriş noktaları sağlar.

Giriş noktaları şunlardır:

#### **MQZ\_AUTHENTICATE\_USER**

Kullanıcı kimliği ve parola doğrulanır ve kimlik bağlamı alanları ayarlayabilir.

**MQZ\_CHECK\_AUTHORITY**

Bir varlığın belirtilen bir nesne üzerinde bir ya da daha fazla işlem gerçekleştirme yetkisinin olup olmadığını denetler.

**MQZ\_CHECK\_IMTIYAZLI**

Belirtilen 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 nesneyle ilişkili tüm yetkileri siler.

**MQZ\_ENUMERATE\_AUTHORITY\_DATA**

Belirtilen seçim ölçütleriyle eşleşen tüm yetki verilerini alır.

**MQZ\_FREE\_USER**

Ayrılmış kaynak ayrılmış kaynakları boşaltabiliyor.

**MQZ\_GET\_AUTHORITY**

Bir varlığın belirtilen bir nesneye erişmesi için sahip olduğu yetkiyi alır.

**MQZ\_GET\_AÇIKLANAMAZ\_YETKISI**

Adlandırılmış bir grubun belirli bir nesneye ( **Kimse** grubu ek yetkisi olmadan) erişmesi ya da belirtilen birincil kullanıcının birincil grubunun belirtilen bir nesneye erişmek zorunda olduğu yetkiye sahip olması ya da yetkisi alır.

**MQZ\_INIT\_AUTHORITY**

Yetkilendirme hizmeti bileşenini kullanıma hazırlar.

**MQZ\_SORGULAMA**

Yetkilendirme hizmetinin desteklenen işlevselliğini sorgular.

**MQZ\_REFRESH\_CACHE**

Tüm yetkileri yenileyin.

**MQZ\_SET\_AUTHORITY**

Bir varlığın belirli bir nesneye sahip olduğu yetkiyi ayarlar.

**MQZ\_TERM\_AUTHORITY**

Yetkilendirme hizmeti bileşenini sona erdirir.

In addition, on IBM MQ for Windows, the authorization service provides the following entry points for use by the queue manager:

- **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) (Güvenlik Tanıtıcısı) kullanımını destekler.

Bu adlar, bileşen işlevlerinin prototipini oluşturmak için kullanılabilen cmqzc . hüstbilgi dosyasında **tipdef** olarak tanımlanır.

Kullanıma hazırlama işlevi ( **MQZ\_INIT\_AUTHORITY** ) bileşene ilişkin ana giriş noktası olmalıdır. Diğer işlevler, başlatma işlevinin bileşen giriş noktası vektörüne eklediği giriş noktası adresinden çağrılır.

*Ad hizmeti arabirimi*

Ad hizmeti, kuyruk yöneticisi tarafından kullanılmak üzere giriş noktaları sağlar.

Aşağıdaki giriş noktaları sağlanmıştır:

**MQZ\_INIT\_NAME**

Ad hizmeti bileşenini başlatın.

**MQZ\_TERM\_ADı**

Ad hizmeti bileşenini sona erdirin.



## **MQZ\_LOOKUP\_NAME**

Belirtilen kuyruk için kuyruk yöneticisi adını arayın.

## **MQZ\_INSERT\_NAME**

Belirtilen kuyruk için sahip olan kuyruk yöneticisi adını içeren bir giriş, hizmet tarafından kullanılan dizine ekler.

## **MQZ\_DELETE\_NAME**

Belirlenen kuyruğa ilişkin girişi, hizmet tarafından kullanılan dizinden silin.

Yapılandırılmış birden fazla ad hizmeti varsa:

- Arama için, kuyruk adı çözümlünceye 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 işlemi için, bu işlevi destekleyen listedeki ilk hizmet için MQZ\_DELETE\_NAME işlevi çağrılır.

Ekleme ve silme işlevlerini destekleyen birden çok bileşende bulunmayın. Ancak, yalnızca aramanın desteklediği bir bileşen uygulanabilir ve örneğin, listedeki diğer herhangi bir ad hizmeti bileşeni tarafından adının tanımlanabileceği bir kuyruk yöneticisine herhangi bir ad tarafından tanınmayan herhangi bir adı çözmek için listedeki son bileşen olarak kullanılabilir.

C programlama dilinde, adlar, tipdef deyimi kullanılarak işlev veri tipleri olarak tanımlanır. Bu bilgiler, 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) dışında, işlevlerin başlatılması, MQZEP çağrısını kullanarak, kullanıma hazırlama işlevinin eklediği giriş noktası adresi tarafından çağrılır.

### *Birden çok hizmet bileşeninin kullanılması*

Bir hizmet için birden çok bileşen kurabilirsiniz. Bu, bileşenlerin yalnızca hizmetin kısmi somutlamalarını sağlamasına ve kalan işlevleri sağlamak için diğer bileşenlere güvenmesine olanak sağlar.

## **Birden çok bileşeni kullanma örneği**

Suppose you create two a name services components called `ABC_name_serv` and `XYZ_name_serv`.

### **ABC\_name\_serv**

Bu bileşen, hizmet dizininden ad eklemeyi ya da bir adı silmesini destekler, ancak kuyruk adını aramaktan destek olmaz.

### **XYZ\_name\_serv**

Bu bileşen, bir kuyruk adını bakmayı destekler, ancak hizmet dizininden bir adı eklemeyi ya da bir adı silmeyi desteklemez.

`ABC_name_serv` bileşeni kuyruk adlarının bir veritabanını bulundurur 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 kuyruk yöneticisi adı döndüren basit bir algoritma kullanır. Kuyruk adları veritabanı tutmaz ve bu nedenle araya ekleme ve silme işlevlerini desteklemez.

Bileşenler aynı kuyruk yöneticisine kurulur. The *ServiceComponent* stanzas are ordered so that component `ABC_name_serv` is invoked first. Bir bileşen dizinine bir kuyruk ekleme ya da silme çağrılarını, `ABC_name_serv` bileşeni tarafından işlenir; Bu fonksiyonları uygulayan tek kişi. However, a lookup call that component `ABC_name_serv` cannot resolve is passed on to the lookup-only component, `XYZ_name_serv`. Bu bileşen, basit algoritmasını kullanarak bir kuyruk yöneticisi adı sağlar.

## Birden çok bileşen kullanılırken giriş noktalarının atlanması

Bir hizmet sağlamak için birden çok bileşen kullanmaya karar verirseniz, belirli işlevleri gerçekleştirmeyen bir hizmet bileşeni tasarlayabilirsiniz. Kurulabilir hizmetler çerçevesi, atlayabileceğiniz herhangi bir kısıtlama içermiyor. Ancak, belirli kurulabilir hizmetler için, bir ya da daha çok işlevin eksik olması, hizmetin amacı ile mantıksal olarak tutarsız olabilir.

## Birden çok bileşenle kullanılan giriş noktaları örneği

Çizelge 143 sayfa 978 , iki bileşenin takıldığı kurulabilir ad hizmetine bir örnek gösterir. Her biri, bu kurulabilir hizmetle ilişkilendirilmiş farklı bir işlev kümesini destekler. Araya ekleme işlevi için, ilk olarak ABC bileşeni giriş noktası çağrılır. Hizmette tanımlanmamış giriş noktaları ( **MQZEP** kullanılarak) NULL olduğu varsayılır. Çizelgede kullanıma hazırlama noktasına ilişkin bir giriş noktası sağlanmıştır; ancak, kullanıma hazırlama, bileşenin ana giriş noktası tarafından gerçekleştirildiğinden, bu gerekli değildir.

Kuyruk yöneticisi kurulabilir bir hizmet kullanmak zorunda olduğunda, o hizmet için tanımlanan giriş noktalarını kullanır ( Çizelge 143 sayfa 978 ' taki sütunlar). Kuyruk yöneticisi, her bileşeni sırayla almak için gereken işlevi gerçekleştiren yordamın adresini belirler. Daha sonra, varsa, yordamı çağırır. İşlem başarılı olursa, herhangi bir sonuç ve durum bilgisi kuyruk yöneticisi tarafından kullanılır.

Çizelge 143. Kurulabilir bir hizmete ilişkin giriş noktaları örneği		
İşlev numarası	ABC ad hizmeti bileşeni	XYZ adı hizmet bileşeni
MQZID_INIT_NAME (Kullanıma Hazırla)	ABC_initialize ()	XYZ_initialize ()
MQZID_TERM_NAME (Sonlandır)	ABC_terminate ()	XYZ_terminate ()
MQZID_INSERT_NAME (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 bu işlemi listede sonraki bileşen için yineler. Ayrıca, yordam varsa, ancak işlemi gerçekleştiremediğini belirten bir kod döndürürse, girişim sonraki kullanılabilir bileşenle devam eder. Hizmet bileşenlerindeki yordamlar, işlemi gerçekleştirmek için başka bir girişimde bulunmayacağına işaret eden bir kod döndürebilir.

## Hizmetlerin ve bileşenlerin yapılandırılması

Hizmet bileşenlerini, kuyruk yöneticisi yapılanış kütüklerini ( Windows sistemleri dışında) kullanarak, her bir kuyruk yöneticisinin Kayıt Defteri 'nde kendi dayanağının bulunduğu bir yerde yapılandırırsmandır.

## Yordam

1. Kuyruk yöneticisine hizmet tanımlamak ve modülün yerini belirtmek için, kuyruk yöneticisi yapılanış kütüğüne stanzalar ekleyin:
  - Kullanılan her hizmetin, kuyruk yöneticisine hizmet tanımlayan bir Service stanza olmalıdır. Daha fazla bilgi için bkz. [qm.ini](#) dosyasının hizmet kısmı.
  - Bir hizmet içindeki her bir bileşen için bir ServiceComponent stanza olmalıdır. Bu kısım, o bileşene ilişkin kodu içeren modülün adını ve yolunu tanımlar. Daha fazla bilgi için bkz. [qm.ini](#) dosyasınınServiceComponent kısmı.

Object Authority Manager (OAM) olarak bilinen yetkilendirme hizmeti bileşeni ürünle birlikte sağlanır. Bir kuyruk yöneticisi yarattığınızda, kuyruk yöneticisi yapılanış kütüğü (ya da Windows sistemlerindeki kayıt dosyası), yetki hizmetine ilişkin ve varsayılan bileşen (OAM) için uygun kısmı içerecek şekilde otomatik olarak güncellenir. Diğer bileşenler için, kuyruk yöneticisi yapılanış kütüğünü el ile yapılandırmanız gerekir.

Kuyruk yöneticisi başlatıldığında, dinamik bağ tanımı kullanılarak, bu altyapıda desteklendiği durumlarda, her hizmet bileşenine ilişkin kod kuyruk yöneticisine yüklenir.

2. Bileşeni etkinleştirmek için kuyruk yöneticisini durdurup yeniden başlatın.

### İlgili başvurular

[qm.ini dosyasının hizmet kısmı](#)

[qm.ini dosyasınınServiceComponent kısmı](#)

*Bir kullanıcının yetkisini değiştirdikten sonra OAM yenileniyor*

IBM MQişletim sisteminde, bir kullanıcının yetki grubu üyeliğini değiştirdikten hemen sonra, işletim sistemi düzeyinde yapılan değişiklikleri yansıtarak kuyruk yöneticisini durdurup yeniden başlatmaya gerek kalmadan, OAM yetki grubu bilgilerini hemen yenileyebilirsiniz. Bunu yapmak için **REFRESH SECURITY** komutunu verin.

**Not:** Yetkileri setmqaut komutuyla değiştirdiğinizde, OAM hemen bu tür değişiklikleri uygular.

Kuyruk yöneticileri, yetki verilerini SYSTEM.AUTH.DATA.QUEUE. Bu veriler **amqzfuma.exe** tarafından yönetilir.

### İlgili başvurular

[Güvenliği yenileme](#)

## IBM i IBM üzerindeki 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 bileşen sağlayabilmesi için belgelenmiş olabilir.

IBM MQ kurulabilir hizmetlerini sağlamanın başlıca nedenleri şunlardır:

- IBM MQ for IBM tarafından sağlanan bileşenleri kullanıp kullanmayacağınızı ya da bunları başkalarıyla tanıyıp kullanmayacağınızı seçme esnekliği ile size yardımcı olmak için.
- Satıcıların, IBM MQ for IBM i' e iç değişiklikler yapmadan, yeni teknolojiler kullanabilecek bileşenler sunarak katkıda bulunmalarına olanak sağlamak için.
- IBM MQ ' in yeni teknolojileri daha hızlı ve daha ucuz bir şekilde kullanmalarına izin vermek ve bu nedenle ürünleri daha önce ve daha düşük fiyatlarla sağlamak.

*Kurulabilir hizmetler ve hizmet bileşenleri* , IBM MQ ürün yapısının bir parçasıdır. Bu yapının merkezinde, kuyruk yöneticisinin, Message Queue Interface (İleti Kuyruğu Arabirimi) ile ilişkili işlevi ve kuralları uygulayan bölümü yer alıyor. Bu merkezi parça, çalışmasını gerçekleştirmek için *kurulabilir hizmetler* adı verilen bir dizi hizmet işlevini gerektirir. IBM MQ for IBM i içinde kullanılabilir olan kurulabilir hizmet, yetkilendirme hizmetidir.

Her kurulabilir hizmet, bir ya da daha fazla *hizmet bileşeni* kullanılarak uygulanan bir ilgili işlev kümesidir. Her bir bileşen, düzgün bir şekilde tasarlanmış, genel kullanıma açık bir arabirim kullanılarak çağrılır. This enables independent software vendors and other third parties to provide installable components to augment or replace those provided by IBM MQ for IBM i. [Çizelge 144 sayfa 979](#) , yetkilendirme hizmetine ilişkin desteği özetler.

Çizelge 144. Yetkilendirme hizmeti bileşenleri özeti		
Sağlanan bileşen	İşlev	Gereksinmeler
Nesne yetkisi yöneticisi (OAM)	Komutlara ve MQI çağrılarına ilişkin yetki denetimi sağlar. Kullanıcılar OAM ' yi büyütme ya da değiştirmek için kendi bileşenlerinden yazabilir.	(Uygun platform yetkilendirme olanakları varsayılr)

Çizelge 144. Yetkilendirme hizmeti bileşenleri özeti (devamı var)		
Sağlanan bileşen	İşlev	Gereksinmeler
DCE adı hizmet bileşeni <b>Not:</b> DCE is only supported on versions of IBM MQ earlier than V6.0.	<ul style="list-style-type: none"> <li>Kuyruk yöneticilerinin kuyrukları paylaşmasına izin verir ya da</li> <li>Kullanıcı tanımlı</li> </ul> <b>Not:</b> Paylaşılan kuyrukların CELL olarak ayarlanmış <b>Scope</b> özneliği olmalıdır.	<ul style="list-style-type: none"> <li>Sağlanan bileşen için DCE 'nin gerekli olması ya da</li> <li>Üçüncü taraf ya da kullanıcı tarafından yazılan bir ad yöneticisi</li> </ul>

## IBM i IBM üzerindeki işlevler ve bileşenler

IBM MQ for IBM i' ta 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:

- Bir kuyruk adı aranır ve kuyruğun tanımlandığı kuyruk yöneticisinin adını döndürür.
- Hizmet dizinine kuyruk adı eklenmesi
- Bir kuyruk adının hizmet dizininden silinmesi

Ayrıca, başlatma ve sonlandırma işlevlerini de içerir.

Kurulabilir bir hizmet, bir ya da daha çok hizmet bileşeni tarafından sağlanır. Her bileşen, söz edilen hizmet için tanımlanmış işlevlerin bazılarını ya da tümünü gerçekleştirebilir. Bu bileşen ayrıca, hizmeti uygulamak için gereksinim duyduğu temel kaynakların ya da yazılımların yönetilmesinden de sorumludur. Yapılandırma dosyaları, bileşeni yüklemek ve sağladığı işlevsel yordamların adreslerini belirlemek için standart bir yol sağlar.

Hizmetler ve bileşenler aşağıdaki gibi ilişkilidir:

- Bir hizmet, bir yapılandırma dosyasındaki stanzas tarafından kuyruk yöneticisinde tanımlanır.
- Her hizmet, kuyruk yöneticisinde sağlanan kodla desteklenir. Kullanıcılar bu kodu değiştiremez ve bu nedenle kendi hizmetlerini oluşturamaz.
- Her hizmet bir ya da daha fazla bileşen tarafından uygulanır; bunlar ürünle birlikte ya da kullanıcı tarafından yazılmış olabilir. Bir hizmet için birden çok bileşen çağrılabilir, her biri hizmet içinde farklı tesisleri destekleyebilir.
- Giriş noktaları, hizmet bileşenlerini kuyruk yöneticisinde destekleyici kodlara 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 yapılandırıldığı sırada sipariş edilmesi, hizmete ilişkin bir isteği karşılamaya yönelik giriş noktalarının çağrıldığı sırayı tanımlar. Sağlanan cmqzc . hüstbilgi dosyasında, sağlanan giriş noktalarının her bir hizmete ilişkin bir MQZID\_ öneki var.

## Dönüş kodları

Hizmet bileşenleri, çeşitli koşullarla ilgili raporlama yapmak için kuyruk yöneticisine dönüş kodları sağlar. İşlemlerin başarılı ya da başarısız olduğunu bildirirler ve kuyruk yöneticisinin bir sonraki hizmet bileşenine ilerleyip ilerlemeyeceğini belirtir. Aynı bir *Devamı* parametresi bu göstergelyi taşır.

## Bileşen verileri

Tek bir hizmet bileşeni, verilerin çeşitli işlevleri arasında paylaşılmasını gerektirebilir. Kurulabilir hizmetler, belirli bir hizmet bileşeninin her çağrısına geçirmek üzere isteğe bağlı bir veri alanı sağlar. Bu veri alanı, hizmet bileşeninin dışlayıcı kullanımı içindir. Verili bir işlevin tüm çağrıları tarafından paylaşılır (bunlar farklı adres alanlarından ya da süreçlerden yapılmış olsa bile). Her çağrıldığında, hizmet bileşeninden

adreslenebilir bir şekilde verileceği garanti edilir. Bu alanın boyutunu *ServiceComponent* stanza içinde bildirmeniz gerekir.

## **IBM i** *IBM üzerinde kullanıma hazırlama*

Bileşen kullanıma hazırlama yordamı çağrıldığında, bileşen tarafından desteklenen her bir giriş noktası için kuyruk yöneticisi MQZEP işlevini çağırmalıdır. MQZEP , hizmete bir giriş noktası tanımlar. Tanımlanmamış tüm çıkış noktalarının boş olduğu varsayılır.

### **Birincil kullanıma hazırlama**

Bir bileşen her zaman bu seçenekle çağrılır ve başka bir şekilde çağrılmadan önce bu seçenekle çağrılır.

### **İkincil kullanıma hazırlama**

Belirli platformlarda bu seçenekle bir bileşen çağrılabilir. Örneğin, hizmete erişildiği her işletim sistemi işlemi, iş parçacığı ya da görev için bir kez çağrılabilir.

İkincil başlatma kullanılırsa:

- İkincil kullanıma hazırlama işlemi için bileşen birden çok kez çağrılabilir. Bu tür her çağrı için, hizmet artık gerekmediği durumlarda ikincil sona erdirmeye için eşleşen bir çağrı yayınlanır.

Yetki hizmetleri için bu, MQZ\_TERM\_AUTHORITY çağrısıdır.

- Bileşen birincil ve ikincil kullanıma hazırlama işlemi için her çağrıldığında, giriş noktalarının yeniden belirtilmesi gerekir (MQZEP çağrılarak).
- Bileşen verilerinin yalnızca bir kopyası bileşen için kullanılır; her ikincil kullanıma hazırlama işlemi için farklı bir kopya yoktur.
- İkincil kullanıma hazırlama gerçekleştirilmeden önce, hizmetin (işletim sistemi işleminden, iş parçacığından ya da görevden uygun olduğu şekilde) başka çağrılar için çağrılmaz.
- The component must set the **Version** parameter to the same value for primary and secondary initialization.

### **Birincil sonlandırma**

Bileşen, artık gerekmediği zaman bu seçenekle birlikte her zaman başlatılır. Bu bileşen için başka arama yapılmayacak.

### **İkincil sona erdirmeye**

İkincil kullanıma hazırlama işlemi için başlatıldıysa, bileşen bu seçenekle başlatılır.

## **IBM i** *IBM üzerinde hizmetlerin ve bileşenlerin yapılandırılması*

Hizmet bileşenlerini, kuyruk yöneticisi yapılandırma kütüklerini kullanarak yapılandırıyoruz.

### **Yordam**

1. Kuyruk yöneticisine bir hizmet tanımlamak ve modülün yerini belirtmek için, kuyruk yöneticisi yapılandırma kütüğüne ( *qm.ini* ) stanzas ekleyin:

- Kullanılan her hizmetin, kuyruk yöneticisine hizmet tanımlayan bir *Service* stanza olmalıdır. Daha fazla bilgi için bkz. [qm.ini dosyasının hizmet kısmı](#).
- Bir hizmet içindeki her bir bileşen için bir *ServiceComponent* stanza olmalıdır. Bu kısım, o bileşene ilişkin kodu içeren modülün adını ve yolunu tanımlar. Daha fazla bilgi için bkz. [qm.ini dosyasınınServiceComponent kısmı](#).

Object Authority Manager (OAM) olarak bilinen yetkilendirme hizmeti bileşeni ürünle birlikte sağlanır. Bir kuyruk yöneticisi yarattığınızda, kuyruk yöneticisi yapılandırma kütüğü otomatik olarak, yetki hizmeti için ve varsayılan bileşen (OAM) için uygun kısmı içerecek şekilde güncellenir. Diğer bileşenler için, kuyruk yöneticisi yapılandırma kütüğünü el ile yapılandırmanız gerekir.

Kuyruk yöneticisi başlatıldığında, dinamik bağ tanımları kullanılarak, bu altyapıda desteklediği durumlarda, her hizmet bileşenine ilişkin kod kuyruk yöneticisine yüklenir.

- 2.

## IBM üzerinde kendi hizmet bileşeninizi oluşturma

IBM MQ for IBM i' ta bir hizmet bileşeninin nasıl yaratılacağı hakkında bilgi edinmek için bu bilgileri kullanın.

Kendi hizmet bileşeninizi yaratmak için:

- cmqzc . h üstbilgi dosyasının programınıza dahil olduğundan emin olun.
- Create the shared library by compiling the program and linking it with the shared libraries libmqm\* and libmqmzf\*.

**Not:** Aracı iş parçacıklı bir ortamda çalıştırılabileceğinden, iş parçacıklı bir ortamda çalıştırmak için OAM ' ı oluşturmanız gerekir. This includes using the threaded versions of libmqm and libmqmzf.

- Kuyruk yöneticisine hizmeti tanımlamak ve modülün yerini belirtmek için, kuyruk yöneticisi yapılanış kütüğüne stanzalar ekleyin.
- Bileşeni etkinleştirmek için kuyruk yöneticisini durdurup yeniden başlatın.

## IBM üzerinde yetkilendirme hizmeti

Yetkilendirme hizmeti, kuyruk yöneticilerinin yetki olanaklarını çağrılarını sağlayan kurulabilir bir hizmettir (örneğin, bir kullanıcı kimliğinin bir kuyruğu açma yetkisi olup olmadığını denetleyerek).

This service is a component of the IBM MQ security enabling interface (SEI), which is part of the IBM MQ framework. Aşağıdaki konular ele alınmıştır:

- [“Nesne yetkisi yöneticisi \(OAM\)” sayfa 982](#)
- [“Hizmet işletim sistemine tanımlanması” sayfa 982](#)
- [“Yetkilendirme hizmeti dayanaklarının yapılandırılması” sayfa 983](#)
- [“IBM üzerinde yetkilendirme hizmeti arabirimi” sayfa 983](#)

### Nesne yetkisi yöneticisi (OAM)

IBM MQ ürünleriyle birlikte sağlanan yetkilendirme hizmeti bileşeni, nesne yetkili yöneticisi (OAM) adı verilir. Varsayılan olarak, OAM etkindir ve aşağıdaki denetim komutlarıyla çalışır:

- **WRKMQMAUT** yetkiyle çalışma
- **WRKMQMAUTD** yetki verileriyle çalışma
- **DSPMQMAUT** görüntü nesnesi yetkisi
- **GRTMQMAUT** nesne yetkisi ver
- **RVKMQMAUT** nesne yetkisini iptal et
- **RFRMQMAUT** Güvenliği yenileme

Bu komutların sözdizimi ve bunların nasıl kullanılacağı, CL komut yardımında açıklanmıştır. OAM, bir birincil kullanıcı ya da grubun *varlığı* ile çalışır.

Bir MQI isteği yapıldığında ya da bir komut verildiğinde OAM, işlemin aşağıdaki eylemleri gerçekleştirip gerçekleştiremeyeceğini görmek için işlemle ilişkili varlığın yetkilendirmesini denetler:

- İstenen işlemi gerçekleştirin.
- Belirtilen kuyruk yöneticisi kaynaklarına erişin.

Yetkilendirme hizmeti, kendi yetkilendirme hizmeti bileşeninizi yazarak kuyruk yöneticileri için sağlanan yetki denetimini artırmanızı ya da değiştirmenizi sağlar.

### Hizmet işletim sistemine tanımlanması

The authorization service stanzas in the queue manager configuration file `qm . ini` define the authorization service to the queue manager. Stanza tipleriyle ilgili bilgi için bkz. [“IBM üzerinde hizmetlerin ve bileşenlerin yapılandırılması” sayfa 981](#) .

## Yetkilendirme hizmeti dayanaklarının yapılandırılması

IBM MQ for IBM i'ta:

### principal

Bir IBM i sistemi kullanıcı tanıtımdır.

### Grup

Bir IBM i sistem grubu tanıtımdır.

Yetkilendirmeler yalnızca grup düzeyinde verilebilir ya da iptal edilebilir. Bir kullanıcının yetkisini verme ya da bu yetkiyi iptal etme isteği, o kullanıcıya ilişkin birincil grubu günceller.

Her kuyruk yöneticisinin kendi kuyruk yöneticisi yapılanış dosyası vardır. Örneğin, kuyruk yöneticisi QMNAME için kuyruk yöneticisi yapılanış kütüğünün varsayılan yolu ve kütük adı /QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini olur.

The *Service* stanza and the *ServiceComponent* stanza for the default authorization component are added to *qm.ini* automatically, but can be overridden by *WRKENVVAR*. Diğer *ServiceComponent* stanzaları el ile eklenmelidir.

Örneğin, kuyruk yöneticisi yapılanış kütüğündeki aşağıdaki stanzalar iki yetki hizmeti bileşenini tanımlar:

```
Service:
  Name=AuthorizationService
  EntryPoints=7

ServiceComponent:
  Service=AuthorizationService
  Name=MQ.UNIX.authorization.service
  Module=QMOM/AMQZFU
  ComponentDataSize=0

ServiceComponent:
  Service=AuthorizationService
  Name=user.defined.authorization.service
  Module=LIBRARY/SERVICE PROGRAM NAME
  ComponentDataSize=96
```

Şekil 105. IBM üzerinde *qm.ini* içindeki yetkilendirme hizmeti dayanakları

The first service component stanza *MQ.UNIX.authorization.service* defines the default authorization service component, the OAM. Bu stanza 'yı kaldırırsanız ve kuyruk yöneticisini yeniden başladırırsanız, OAM devre dışı bırakılır ve yetki denetimi yapılmamaktadır.

### IBM i IBM üzerinde yetkilendirme hizmeti arabirimi

Yetkilendirme hizmeti arabirimi, kuyruk yöneticisi tarafından kullanılmak üzere çeşitli giriş noktaları sağlar.

#### **MQZ\_AUTHENTICATE\_USER**

Kullanıcı kimliği ve parola doğrulanır ve kimlik bağlamı alanları ayarlayabilir.

#### **MQZ\_CHECK\_AUTHORITY**

Bir varlığın belirtilen bir nesne üzerinde bir ya da daha fazla işlem gerçekleştirme yetkisinin olup olmadığını denetler.

#### **MQZ\_COPY\_ALL\_AUTHORITY**

Başvurulan bir nesne için var olan tüm geçerli yetkileri başka bir nesneye kopyalar.

#### **MQZ\_DELETE\_AUTHORITY**

Belirtilen nesneyle ilişkili tüm yetkileri siler.

#### **MQZ\_ENUMERATE\_AUTHORITY\_DATA**

Belirtilen seçim ölçütleriyle eşleşen tüm yetki verilerini alır.

#### **MQZ\_FREE\_USER**

Ayrılmış kaynak ayrılmış kaynakları boşaltabiliyor.

#### **MQZ\_GET\_AUTHORITY**

Bir varlığın belirtilen bir nesneye erişmesi için sahip olduğu yetkiyi alır.

## **MQZ\_GET\_AÇIKLANAMAZ\_YETKISI**

Adlandırılmış bir grubun belirli bir nesneye ( **Kimse** grubu ek yetkisi olmadan) erişmesi ya da belirtilen birincil kullanıcının birincil grubunun belirtilen bir nesneye erişmek zorunda olduğu yetkiye sahip olması ya da yetkisi alır.

## **MQZ\_INIT\_AUTHORITY**

Yetkilendirme hizmeti bileşenini kullanıma hazırlar.

## **MQZ\_SORGULAMA**

Yetkilendirme hizmetinin desteklenen işlevselliğini sorgular.

## **MQZ\_REFRESH\_CACHE**

Tüm yetkileri yenileyin.

## **MQZ\_SET\_AUTHORITY**

Bir varlığın belirli bir nesneye sahip olduğu yetkiyi ayarlar.

## **MQZ\_TERM\_AUTHORITY**

Yetkilendirme hizmeti bileşenini sona erdirir.

Bu giriş noktaları, Windows Security Identifier (NT SID) (Güvenlik Tanıtıcısı) kullanımını destekler.


Bu adlar, bileşen işlevlerinin prototipini oluşturmak için kullanılabilen cmqzc . hüstbilgi dosyasında **tipdef** olarak tanımlanır.

Kullanıma hazırlama işlevi ( **MQZ\_INIT\_AUTHORITY** ) bileşene ilişkin ana giriş noktası olmalıdır. Diğer işlevler, başlatma işlevinin bileşen giriş noktası vektörüne eklediği giriş noktası adresinden çağrılır.

Ek bilgi için [“IBM üzerinde kendi hizmet bileşeninizi oluşturma” sayfa 982](#) başlıklı konuya bakın.

## **Multi API çıkışlarının yazılması ve derlenmesi**

API çıkışları, IBM MQ API çağrılarının davranışını değiştiren (MQPUT ve MQGET gibi) kod yazmanızı sağlar ve bu çağrılarının hemen ardından ya da hemen sonra bu kodu eklemenize olanak sağlar.

**Not:**  IBM MQ for z/OS üzerinde desteklenmez.

## **API çıkışları neden kullanılır?**

Uygulamalarınızın her birinin yapması gereken belirli bir işi vardır ve bu işin kodu, görevi mümkün olduğunca verimli bir şekilde yapmalıdır. Daha yüksek bir düzeyde, standart ya da iş süreçlerini, o kuyruk yöneticisini kullanan tüm uygulamalar için belirli bir kuyruk yöneticisine uygulamak isteyebilirsiniz. Bunu tek tek uygulamalar düzeyinin üzerinde yapmak daha verimli ve etkilenen her uygulamanın kodunu değiştirmek zorunda kalmaksızın.

API çıkışlarının yararlı olabileceği alanlar için birkaç öneri vardır:

### **Güvenlik**

Güvenlik için, uygulamaların bir kuyruğa ya da kuyruk yöneticisine erişme yetkisine sahip olup olmadığını denetleyerek kimlik doğrulaması sağlayabilirsiniz. Ayrıca, polis uygulamalarının API ' yı kullanarak, bireysel API çağrılarını doğrulayabilir, hatta kullandıkları parametreleri de doğrulayabilirsiniz.

### **Esneklik**

Esneklik için, bu ortamdaki verilere dayalı olan uygulamaları değiştirmeden, iş ortamınızdaki hızlı değişikliklere yanıt verebilirsiniz. Örneğin, faiz oranlarındaki değişikliklere, para birimi döviz kurlarına ya da bir üretim ortamındaki bileşenlerin fiyatlarına yanıt veren API çıkışlarına sahip olabilir.

### **Kuyruğun ya da kuyruk yöneticisinin kullanımının izlenmesi**

Bir kuyruk ya da kuyruk yöneticisinin izlenmesinde, uygulama ve ileti akışını izleyebilirsiniz, API çağrılarında hataları günlüğe kaydedebilir, muhasebe işlemleri için denetleme izlerini ayarlayabilir ya da planlama amacıyla kullanım istatistiklerini toplayabilirsiniz.



## Bir API çıkışı çalıştırıldığında ne olur?

Bir çıkış programı yazdıktan ve bunu IBM MQ' a tanıtdıktan sonra, kuyruk yöneticisi çıkış kodunuzu otomatik olarak kayıtlı noktalarda çağırır.

Çalıştırılacak API çıkış yordamları, aşağıdaki platformlardaki stanzas içinde tanımlanır:

- **IBM i** IBM i
- **Linux** **UNIX** UNIX and Linux
- **Windows** Windows

This topic covers the stanzas in the configuration files `mqs.ini` and `qm.ini`.

Yordamların tanımı üç yerde oluşabilir:

1. `ApiExitCommon`, in the `mqs.ini` file, identifies routines, for the whole of IBM MQ, applied when queue managers start. Bunlar, tek tek kuyruk yöneticileri için tanımlanan yordamlar tarafından geçersiz kılınabilir (bu listedeki “3” sayfa 985 ögesine bakın).
2. `ApiExitTemplate`, in the `mqs.ini` file, identifies routines, for the whole of IBM MQ, copied to the `ApiExitLocal` set (see item “3” sayfa 985 in this list) when a new queue manager is created.
3. `ApiExitYerel` olarak, `qm.ini` dosyasında, belirli bir kuyruk yöneticisi için geçerli olan yordamları tanımlar.

When a new queue manager is created, the `ApiExitTemplate` definitions in `mqs.ini` are copied to the `ApiExitLocal` definitions in `qm.ini` for the new queue manager. Bir kuyruk yöneticisi başlatıldığında, hem `ApiExitOrtak`, hem de `ApiExitYerel` tanımlamaları kullanılır. The `ApiExitLocal` definitions replace the `ApiExitCommon` definitions if both identify a routine of the same name. “API çıkışlarını yapılandırma” sayfa 990 içinde açıklanan Sequence özniteliği, stanzas çalıştırmasında tanımlanan yordamların sıralarını belirler.

## Using API exits across multiple installations of IBM MQ

IBM WebSphere MQ 7.1 'ta çıkışlarda yapılan değişikliklerin önceki bir sürümle çalışmaması nedeniyle, IBM MQ ' un önceki sürümü için yazılan API çıkışlarının tüm sürümlerle çalışmak için kullanıldığından emin olun. Çıkışlar için yapılan değişikliklerle ilgili daha fazla bilgi için bkz. “Writing exits and installable services on UNIX, Linux and Windows” sayfa 968.

API, amqsadm ve amqsaxe için sağlanan örnekler, çıkışlar yazılırken gerekli değişiklikleri yansıtır. İstemci uygulaması, uygulamanın başlatıldığı kuyruk yöneticisinin kuruluşuna karşılık gelen doğru IBM MQ kitaplıklarının, uygulamanın başlatılmasından önce bu kitapla bağlantılı olduğundan emin olmalıdır.

### Multi **API çıkışları yazılıyor**

C programlama dilini kullanarak her API çağrısı için çıkış yazabilirsiniz.

## Kullanılabilir çıkışlar

Her API çağrısı için aşağıdaki gibi tüm çıkışlar kullanılabilir:

- MQCB, belirtilen nesne tanıtıcısı ve denetim etkinleştirmesi için geri çağrıyı yeniden kaydettirmek ve geri bildirme için yapılan değişiklikleri denetlemek için
- MQCTL, bağlantı için açılan nesne tanıtıcılarında denetleme işlemlerini gerçekleştirmek için
- MQCONN/MQCONNX, sonraki API çağrılarında kullanılmak üzere kuyruk yöneticisi bağlantı tanıtıcısı sağlamak için
- MQDISC, kuyruk yöneticisinden bağlantıyı kesmek için
- MQBEGIN, genel iş birimini başlatmak için (UOW)
- MQBACK, bir UOW ' u yedeklemek için
- MQCMIT, bir UOW ' u kesinleştirmek için
- Sonraki erişime ilişkin bir IBM MQ kaynağını açmak için MQOPEN

- MQCLOSE, daha önce erişim için açılmış bir IBM MQ kaynağını kapatmak için
- Erişim için önceden açılmış bir kuyruktan ileti almak için MQGET
- MQPUT1, bir iletiyi kuyruğa yerleştirmek için
- MQPUT, daha önce erişim için açılmış olan bir kuyruğa ileti yerleştirecek
- MQINQ, daha önce erişim için açılmış olan bir IBM MQ kaynağının özniteliklerini sorgulamak için
- Erişim için önceden açılmış bir kuyruğun özniteliklerini ayarlamak için MQSET
- MQSTAT, durum bilgilerini almak için
- MQSUB, uygulama aboneliğini belirli bir konuya kaydettirmek için
- Bir abonelik isteği yapmak için MQSUBRQ

MQ\_CALLBACK\_EXIT, geri bildirme işleminden önce ve sonra gerçekleştirilmek üzere bir çıkış işlevi sağlar. Ek bilgi için [Callback-MQ\\_CALLBACK\\_EXIT](#) başlıklı konuya bakın.

## API çıkışları yazılıyor

API çıkışlarında, aramalar genel formu alır:

```
MQ_call_EXIT (parameters, context, ApiCallParameters)
```

Burada *call*, MQ öneki olmayan MQI adını belirtir; örneğin, PUT, GET. *parameters*, çıkış ve dış denetim blokları MQAXP (API çıkış değiştirgesi yapısı) ve MQAXC (API çıkış bağlamı yapısı) arasında iletişim sağlayan, çıkışa ilişkin işlevi denetler. *context*, API çıkışının çağrıldığı bağlamı açıklar ve *ApiCallParameters*, MQI çağrısına ilişkin parametreleri gösterir.

API çıkışınızı yazmanıza yardımcı olması için örnek bir çıkış (amqsaxe0.c) sağlanır; bu çıkış, izleme girişlerini belirlediğiniz bir dosyaya oluşturur. Bu örneği, çıkışlar yazılırken başlangıç noktanız olarak kullanabilirsiniz. Örnek çıkışı kullanmaya ilişkin daha fazla bilgi için bkz. [“API çıkış örnek programı” sayfa 1108](#).

API çıkış çağruları, dış denetim blokları ve ilişkili konular hakkında daha fazla bilgi için bkz. [API çıkış başvurusu](#).

Bir çıkışa nasıl yazılacağı, derleneceği ve yapılandırılmasına ilişkin genel bilgiler için bkz. [“Writing exits and installable services on UNIX, Linux and Windows” sayfa 968](#).

## API çıkışlarında ileti tanıtıcıları kullanılıyor

Bir API çıkışının erişimi olan ileti özelliklerini denetleyebilirsiniz. Özellikler, bir ExitMsgİleciyle ilişkilendirilir. Put exit (put) çıkışta ayarlanan özellikler, yerleştirilecek iletiye ayarlanır, ancak alma çıkışta alınan özellikler uygulamaya geri döndürülmez.

**Function** ile MQXF\_INIT ve **ExitReason** MQXR\_CONNECTION değerine ayarlanmış MQXEP MQI çağrısını kullanarak bir MQ\_INIT\_EXIT çıkış işlevini kaydettirdiğinizde, bir MQXEPO yapısını **ExitOpts** değiştirgesi olarak geçirmenizi sağlar. MQXEPO yapısı, çıkışta kullanılacak özellikler kümesini belirten ExitProperties (ExitProperties) alanını içerir. Özelliklerin önekini gösteren, bir MQRFH2 klasör adına karşılık gelen bir karakter dizisi olarak belirtilir.

Her API çıkışı, bir ExitMsgHandle alanı içeren bir MQAXP yapısı alır. Bu alan, IBM MQ tarafından oluşturulan bir değere ayarlanır ve bir bağlantıya özgüdür. Bu nedenle, aynı bağlantıda aynı ya da farklı tiplerde API çıkışları arasında değişmeden kalır.

In an MQ\_PUT\_EXIT or MQ\_PUT1\_EXIT with an **ExitReason** of MQXR\_BEFORE, that is, an API exit performed before putting a message, any properties (other than message descriptor properties) associated with the ExitMsgHandle when the exit completes are set on the message being put. Bu işlemi önlemek için, ExitMsgtanıtıcısını MQHM\_NONE olarak ayarlayın. Ayrıca, farklı bir ileti tanıtıcısı da sağlayabilirsiniz.

Bir MQ\_GET\_EXIT ve MQ\_CALLBACK\_EXIT içinde, ExitMsgtanıtıcısı özelliklerden temizlenir ve MQ\_INIT\_EXIT değeri, ileti tanımlayıcı özellikleri dışında, ExitProperties alanında belirtilen özelliklerle

doldurulur. Bu özellikler, alma uygulaması tarafından kullanılabilir kılınmaz. Alma uygulaması, MQGMO (İleti seçenekleri al) alanında bir ileti tanıtıcısı belirlediyse, ileti tanımlayıcı özellikleri de içinde olmak üzere, o tanıtıcı ile ilişkilendirilmiş tüm özellikler API çıkışa kullanılabilir. ExitMsgHandle 'in özelliklerle doldurulmasını önlemek için, bu değeri MQHM\_NONE olarak ayarlayın.

**Not:** Çıkış iletisi özelliklerinin işleneceği yer:

- MQ\_GET\_EXIT işlevinin ardından, çıkış için bir MQ\_GET\_EXIT işlevinden önce bir değer tanımlamanız gerekir.
- MQ\_CALLBACK\_EXIT işlevinden önce, çıkışa ilişkin MQ\_CB\_EXIT işlevinden önce bir değer tanımlamanız gerekir.

API çıkışlarında ileti tutamaçlarının kullanımını göstermek için bir örnek program ( amqsaem0.c) sağlanır.

## İlgili başvurular

Kullanıcı çıkışları, API çıkışları ve kurulabilir hizmetler başvurusu

### Multi **API çıkışları derleniyor**

Bir çıkış yazdıktan sonra, aşağıdaki şekilde derleyip bağlarsınız.

Aşağıdaki örneklerde, “API çıkış örnek programı” sayfa 1108’inde açıklanan örnek program için kullanılan komutlar gösterilmektedir. For platforms other than Windows systems, you can find the sample API exit code in `MQ_INSTALLATION_PATH/samp` and the compiled and linked shared library in `MQ_INSTALLATION_PATH/samp/bin`.

**Windows** Windows sistemleri için, örnek API çıkış kodunu `MQ_INSTALLATION_PATH\Tools\c\Samp`lesinde bulabilirsiniz. `MQ_INSTALLATION_PATH`, IBM MQ ' in kurulu olduğu dizini temsil eder.

**Not:** 64 bit kullanan programlamaya ilişkin yönergeler, [64 bit altyapılarda Coding standartlarındayer alan bir programdır](#).

Çok hedefli istemciler için, bazı iletiler kuyruk yöneticisinden geçemeyebileceğinden, API çıkışlar ve veri dönüştürme çıkışlarının istemci tarafında çalışabilmesi gerekir. Aşağıdaki kitaplıklar istemci paketlerinin yanı sıra sunucu paketleri de içerir:

Çizelge 145. İstemcide ve sunucu paketlerinde bulunan kitaplıklar	
İşletim sistemi	Kitaplıklar
<b>AIX</b> AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
<b>IBM i</b> IBM i	LIBMQM & LIBMQM_R
<b>Linux</b> Linux	32 bit ve 64 bit: libmqm.so & libmqm_r.so
<b>Solaris</b> Solaris	32 bit ve 64 bit: libmqm.so
<b>Windows</b> Windows	32 bit & 64 bit: mqm.dll & mqm.pdb

### Linux **UNIX** Compiling API exits on UNIX and Linux systems

Examples of how to Compile API exits on UNIX and Linux systems.

Tüm altyapılarda, modüle giriş noktası MQStart 'tır.

`MQ_INSTALLATION_PATH`, IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

## AçıkAIX

### AIX

API çıkışı kaynak kodunu derleyerek aşağıdaki komutlardan birini çalıştırın:

## 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

API çıkışı kaynak kodunu derleyerek aşağıdaki komutlardan birini çalıştırın:

## 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
```

## AçıkSolaris

### Solaris

API çıkışı kaynak kodunu derleyerek aşağıdaki komutlardan birini çalıştırın:

### 32 bit uygulamalar

#### SPARC altyapısı

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/amqsaxe \  
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc \  
-R/usr/lib/32 -lsocket -lnsl -ldl
```

#### x86-64 platformu

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/amqsaxe \  
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc \  
-R/usr/lib/32 -lsocket -lnsl -ldl
```

### 64 bit uygulamalar

#### SPARC altyapısı

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \  
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc \  
-R/usr/lib/64 -lsocket -lnsl -ldl
```

#### x86-64 platformu

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/amqsaxe \  
amqsaxe0.c -I MQ_INSTALLATION_PATH/inc \  
-R/usr/lib/64 -lsocket -lnsl -ldl
```

### Windows

API çıkışlarının Windows sistemlerinde derlenmesi

Örnek API çıkış programını ( amqsaxe0.c, Windowsüzerinde derleyin ve bağlayın)

Bildirge (manifest) dosyası, derlenmiş bir uygulamaya ya da DLL ' ye gömülebilen, sürümü içeren isteğe bağlı bir XML belgesidir.

Bu tür bir belgeniz yoksa, **mt** komutundaki **-bildirge manifest.file** parametresini kaldırın.

Adapt the commands in the examples in [Şekil 106 sayfa 990](#) or [Şekil 107 sayfa 990](#) to compile and link amqsaxe0.c on Windows. Komutlar, Microsoft Visual Studio 2008, 2010 ya da 2012 ile çalışır. Bu örneklerde, C:\Program Files\IBM\MQ\tools\c\samples dizininin yürürlükteki dizin olduğu varsayılmıştır.

## 32 bit

```
cl /c /nologo /MD /Foamsaxe0.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 106. 32 bit Windows üzerindeki amqsaxe0.c derleme ve bağlantı

## 64 bit

```
cl /c /nologo /MD /Foamsaxe0.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 107. 64 bit Windows üzerinde amqsaxe0.c derleme ve bağlantı

### İlgili kavramlar

“API çıkış örnek programı” sayfa 1108

Örnek API çıkışı, kullanıcı tarafından belirtilen bir dosyaya MQAPI\_TRACE\_LOGFILE ortam değişkeninde tanımlı bir örnek içeren bir MQI izleme oluşturur.

**IBM i** *Compiling API exits on IBM i*

Compiling API exits on IBM i.

Aşağıdaki gibi bir çıkış oluşturulur (C dili örneği için):

1. CRTCMOD kullanarak bir modül yaratın. Compile it to use teraspace by including the parameter TERASPACE(\*YES \*TSIFC).
2. CRTSRVPGM kullanarak modülden bir hizmet programı yaratın. Çok iş parçacıklı API çıkışları için, bu hizmet programı QMQM/LIBMQMZF\_R hizmet programına bağlanmanız gerekir.

### API çıkışlarını yapılandırma

Yapılandırma bilgilerini değiştirerek API çıkışlarını etkinleştirmek için IBM MQ özelliğini yapılandırıyorsunuz.

Yapılandırma bilgilerini değiştirmek için, çıkış yordamlarını tanımlayan stanzaları ve bunların çalıştırıldığı sırası değiştirmelisiniz. Bu bilgiler aşağıdaki şekillerde değiştirilebilir:

- **Linux** **Windows** Using the IBM MQ Explorer on Windows and Linux (x86 and x86-64 platforms).
- **Windows** Windows üzerinde **amqmdain** komutunu kullanma.

- **Multi** mqs.ini ve qm.ini dosyalarını doğrudan Windows, **IBM i** IBM i, ve UNIX and Linux sistemlerinde kullanma.

mqs.ini dosyası, belirli bir düğümdeki tüm kuyruk yöneticilerine ilişkin bilgileri içerir. Bunu aşağıdaki konumlarda bulabilirsiniz:

- **IBM i** IBM idizinde /QIBM/UserData/mqm dizininde.
- **Linux** **UNIX** UNIX and Linux dizininde /var/mqm dizininde.
- **Windows** Windows sistemlerinde HKLM\SOFTWARE\IBM\WebSphere MQ anahtarında belirtilen WorkPath 'nda.

qm.ini dosyası, belirli bir kuyruk yöneticisiyle ilgili bilgileri içerir. Kuyruk yöneticisi tarafından meşgul edilen izin ağacının kökünde tutulan her kuyruk yöneticisi için bir kuyruk yöneticisi yapılandırma kütüğü vardır. Örneğin, QMNAME adı verilen bir kuyruk yöneticisine ilişkin bir yapılandırma kütüğünün yolu ve adı:

**IBM i** IBM i sistemlerinde:

```
/QIBM/UserData/mqm/qmgrs/QMNAME/qm.ini
```

**Linux** **UNIX** UNIX 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, bir kopyaya sahip olmak için gereksinim duyarsa geri dönebileceğiniz bir dosyayı yedeklemeniz gerekir.

Yapılandırma kütüklerini aşağıdaki yöntemlerden birini kullanarak düzenleyebilirsiniz:

- Düğümdeki kuyruk yöneticilerinin yapılandırmasını değiştiren komutları kullanarak otomatik olarak.
- Standart bir metin düzenleyicisi kullanarak el ile.

Bir yapılandırma dosyası öznitelğe yanlış bir değer ayarladıysanız, değer yoksayılr ve sorunu belirtmek için bir işletmen iletisi yayınlanır. Bu etki, öznitelğin tümüyle eksik olduğu şekilde aynıdır.

## Yapılandırılacak stanzas

Değiştirilmesi gereken stanzalar şunlardır:

### ApiExitOrtak

Defined in mqs.ini and in the IBM MQ Explorer on the IBM MQ properties page, under Exits.

Kuyruk yöneticisi başlatıldığında, bu stanza içindeki öznitelikler okunur ve qm.ini içinde tanımlanan API çıkışlarıyla geçersiz kılınır.

### ApiExitŞablonu

Defined in mqs.ini and in the IBM MQ Explorer on the IBM MQ properties page, under Exits.

Herhangi bir kuyruk yöneticisi yaratıldığında, bu stanza içindeki öznitelikler, ApiExitLocal stanza altındaki yeni oluşturulan qm.ini dosyasına kopyalanır.

### ApiExitYerel

Defined in qm.ini and in the IBM MQ Explorer on the queue manager properties page, under Exits.

Kuyruk yöneticisi başlatıldığında, burada tanımlanan API çıkışları, mq.s.ini içinde tanımlanan varsayılan değerleri geçersiz kılar.

## Stanzalara ilişkin öznitelikler

- API çıkışa aşağıdaki özniteliği kullanarak ad girin:

### **Ad=ApiExit\_name**

MQAXP yapısının ExitInfoAd alanında geçirilen API çıkışa ilişkin açıklayıcı ad.

Bu ad benzersiz olmalı, 48 karakterden uzun olmamalıdır ve yalnızca IBM MQ nesnelерinin adları (örneğin, kuyruk adları) için geçerli karakterler içermelidir.

- Aşağıdaki öznitelikleri kullanarak çalıştırmak için API çıkış kodunun modül ve giriş noktasını tanımlayın:

### **Function=function\_name**

İşlev giriş noktasının adı, API çıkış kodunu içeren modüle işaret eder. Bu giriş noktası, MQ\_INIT\_EXIT işlevidir.

Bu alanın uzunluğu, MQ\_EXIT\_NAME\_LENGTH ile sınırlıdır.

### **Module=modüle\_adi**

API çıkış kodunu içeren modül.

Bu alan, olduğu gibi kullanılan modülün tam yol adını içeriyorsa.

Bu alan yalnızca modül adı içeriyorsa, modül qm.ini içindeki ExitPath içindeki ExitsDefaultPath özniteliği kullanılarak bulunur.

Ayrı iş parçacıklı kitaplıkları destekleyen altyapılarda, API çıkış modülünün iş parçacıklı ve iş parçacıklı bir sürümünü sağlamanız gerekir. Yıvli sürümdeki bir \_r soneki olmalıdır. The threaded version of the IBM MQ application stub implicitly appends \_r to the given module name before it is loaded.

Bu alanın uzunluğu, platformun desteklediği yol uzunluğu üst sınırı ile sınırlıdır.

- İsteğe bağlı olarak, aşağıdaki özniteliği kullanarak çıkışa veri iletin:

### **Veri=veri\_adi**

MQAXP yapısındaki ExitData alanında API çıkışa geçirilecek veriler.

Bu özniteliği eklerseniz, baştaki ve sondaki boşluklar kaldırılırsa, kalan dizgi 32 karaktere kesilir ve sonuç çıkışa geçirilir. Bu özniteliği atlarsanız, çıkışa 32 boşluktan oluşan varsayılan değer iletilir.

Bu alanın uzunluk üst sınırı 32 karakterdir.

- Aşağıdaki özniteliği kullanarak diğer çıkışlarla ilgili olarak bu çıkışa ilişkin sırayı tanımlayın:

### **Sequence=sıra\_numarası**

Bu API çıkışının diğer API çıkışlarına göre çağrıldığı sıra. Sıra numarası düşük olan bir çıkış, daha yüksek sıra numarasına sahip bir çıkıştan önce çağrılır. Çıkışların sıra numaralandırmasına bitişik olacak şekilde gerek yoktur. 1, 2, 3 gibi bir sıra, 7, 42, 1096 diziyle aynı sonucu elde eder. İki çıkış aynı sıra numarasına sahip olursa, kuyruk yöneticisi hangisinin önce arayacağına karar verir. MQAXP ' de ExitChainAreaPtr tarafından belirtilen ExitChainArea alanına saati ya da bir imleyiciyi koyarak ya da kendi günlük dosyanızı yazarak, olayın ardından hangilerinin çağrıldığını söyleyebilirsiniz.

Bu öznitelik işaretersiz bir sayısal değer.

## Örnek stanzas

Örnek mq.s.ini dosyası aşağıdaki stanzaları içerir:

### **ApiExitŞablonu**

Bu stanza, açıklayıcı adı OurPayrollQueueAuditor, modül adı auditorve sıra numarası 2 ile bir çıkış tanımlar. Çıkışa 123 veri değeri iletilir.



## ApiExitOrtak

This stanza defines an exit with the descriptive name MQPoliceman, module name tmqp, and sequence number 1. İletilen 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, birim adı ClientAppChecker ve sıra numarası 3 ile çıkışa ilişkin bir ApiExitYerel tanımlaması içerir.

```
qm.ini

ApiExitLocal:
  Name=ClientApplicationAPIchecker
  Sequence=3
  Function=EntryPoint
  Module=/usr/Dev/ClientAppChecker
  Data=9.20.176.20
```

## İleti alışverişi kanallarına ilişkin kanal çıkışı programları

Bu konu derlemi, ileti alışverişi kanallarına ilişkin IBM MQ kanal çıkış programlarıyla ilgili bilgileri içerir.

İleti kanalı araçları (MCA ' lar) veri dönüştürme çıkışları da çağırabilir. Veri dönüştürme çıkışlarını yazma hakkında daha fazla bilgi için bkz. [“Veri dönüştürme çıkışları yazılıyor” sayfa 1014.](#)

Bu bilgilerin bazıları, IBM MQ MQI clients ' u kuyruk yöneticilerine bağlayan MQI kanallarındaki çıkışlar için de geçerlidir. Ek bilgi için [MQI kanallarına ilişkin kanal çıkışı programları](#) başlıklı konuya bakın.

Kanal çıkış programları, MCA programları tarafından gerçekleştirilen işlemde tanımlı yerlerde çağrılır.

Bu kullanıcı çıkışı programlarından bazıları tamamlayıcı çiftlerde çalışır. Örneğin, ileti gönderme işlevi tarafından iletilecek iletileri şifrelemek için bir kullanıcı çıkışı programı çağrılırsa, süreci tersine çevirmek için tamamlayıcı işlem, alma uçta çalışır durumda olmalıdır.

[Çizelge 146 sayfa 993](#) , her kanal tipi için kullanılabilir olan kanal çıkışı tiplerini gösterir.

Çizelge 146. Her kanal tipi için kanal çıkışları kullanılabilir						
Kanal Tipi	İleti çıkışı	İleti-çıkışı yeniden dene	Çıkış al	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

Çizelge 146. Her kanal tipi için kanal çıkışları kullanılabilir (devamı var)

Kanal Tipi	İleti çıkışı	İleti-çıkışı yeniden dene	Çıkış al	Güvenlik Çıkışı	Çıkış gönder	Otomatik tanımlama çıkışı
İstekte bulunanın 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/OS' ta, otomatik tanımlama çıkışı yalnızca küme gönderici ve küme alıcı kanalları için geçerlidir.

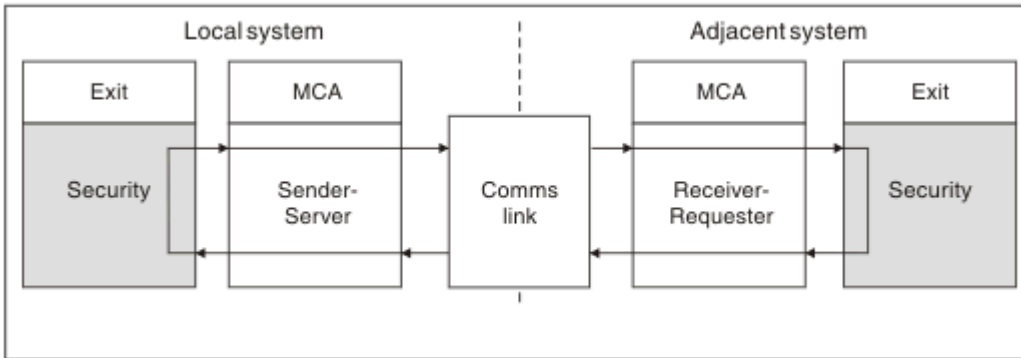
Bir istemcide kanal çıkışlarını çalıştırabiliyorsanız, MQSERVER ortam değişkenini kullanamazsınız. Bunun yerine, İstemci kanal tanımlama çizelgesi içinde açıklandığı gibi bir istemci kanal tanımlama çizelgesi (CCDT) yaratın ve başvurun.

### İşleme genel bakış

MCA ' ların kanal çıkış programlarını nasıl kullanlarına genel bakış.

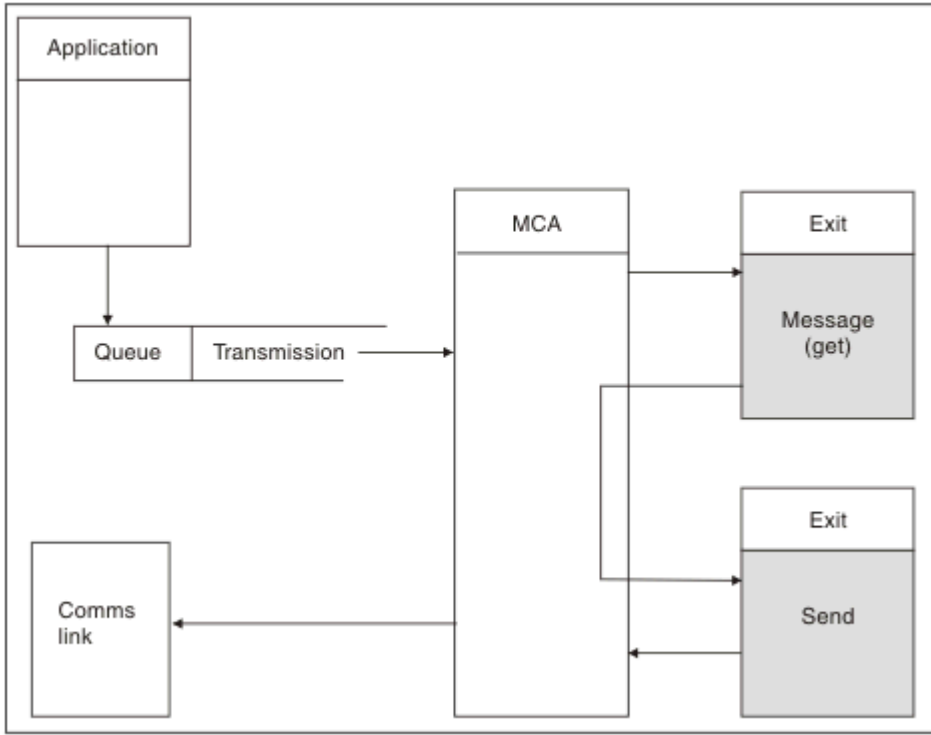
Başlatma sırasında, MCA ' lar işlemeyi eşitlemek için bir başlatma iletişim kutusu değiştirir. Daha sonra, güvenlik çıkışlarını içeren bir veri değiş tokasına geçiyorlar. Başlatma aşamasını tamamlamak ve iletilerin aktarılmasına izin vermek için bu çıkışlar başarıyla sona ermelidir.

Güvenlik denetimi aşaması bir döngüdür ( Şekil 108 sayfa 994 içinde gösterildiği gibi).

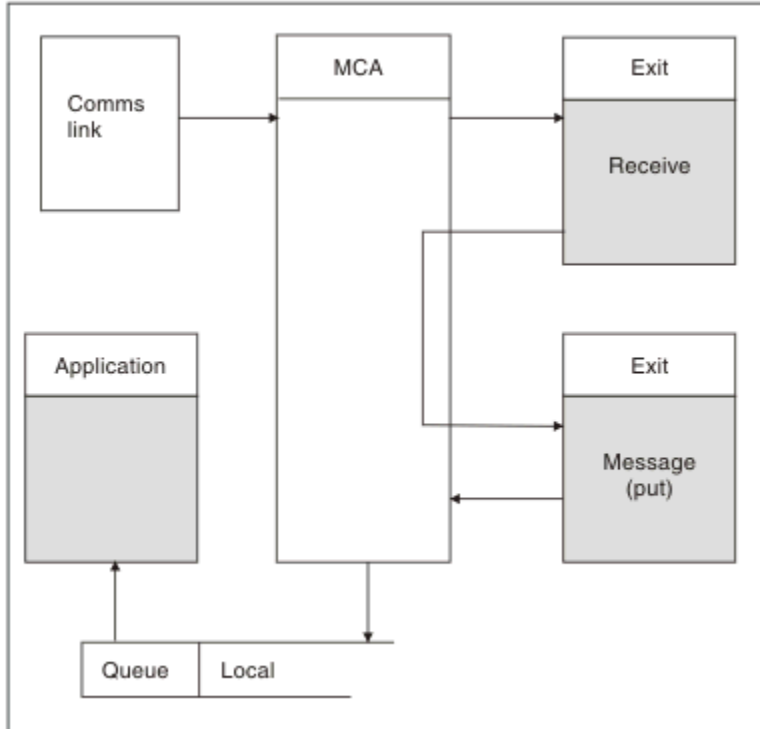


Şekil 108. Güvenlik çıkış döngüsü

İleti aktarma aşaması sırasında, MCA 'yı gönderme işlemi iletileri bir iletim kuyruğundan alır, ileti çıkışını çağırır, gönderme çıkışını çağırır ve Şekil 109 sayfa 995 içinde gösterildiği şekilde iletiyi alma MCA' sına gönderir.



Şekil 109. İleti kanalının gönderici bitişindeki gönderme çıkışı örneği



Şekil 110. İleti kanalının günlük nesnesindeki alma çıkışa örneği

The receiving MCA receives a message from the communications link, calls the receive exit, calls the message exit, and then puts the message on the local queue, as shown in Şekil 110 sayfa 995. (Alma çıkışa, 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 çıkışı programları, izleyen kısımlarda belirtilenler dışında tüm MQI çağrılarını kullanabilir. MQ V7 ve sonraki sürümleri için, MQCXP yapısı sürüm 7 ve üstü, MQCONN komutu vermek yerine kullanılacak hConnbağlantı tanıtıcısını içerir. Önceki sürümler için, kanal kendisi kuyruk yöneticisine bağlı olduğundan, bir MQRC\_ALREADY\_CONNECTED uyarısı döndürülse de, bağlantı tanıtıcısını almak için bir MQCONN yayınlanmalıdır.

Kanal çıkışının iş parçacığı korumalı olması gerektiğini unutmayın.

İstemci-bağlantı kanallarındaki çıkışlar için, çıkışa bağlanmayı denediği kuyruk yöneticisi, çıkışa nasıl bağlı olduğuna bağlıdır. Çıkış MQM.LIB (ya da IBM i üzerinde QMQM/LIBMQM) ile bağlantılıysa 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 çıkışa geçirilen kuyruk yöneticisinin adını, MQCD ' nin QMgrName alanı aracılığıyla belirtirseniz, çıkış o kuyruk yöneticisine bağlanmayı dener. Çıkış MQIC.LIB ya da başka bir kitaplık için, MQCONN çağrısı bir kuyruk yöneticisi adı belirtme ya da not belirtme belirtmediğiniz için başarısız olur.

You should avoid altering the state of the transaction associated with the passed hConn in a channel exit; you must not use the MQCMIT, MQBACK or MQDISC verbs with the channel hConn, and you cannot use the MQBEGIN verb specifying the channel hConn.

MQCONNX, yeni bir IBM MQ bağlantısı yaratmak için MQCNO\_HANDLE\_SHARE\_BLOCK ya da MQCNO\_HANDLE\_SHARE\_NO\_BLOCK belirtilerek kullanılıyorsa, bağlantının doğru olarak yönetilmesini ve kuyruk yöneticisinden bağlantıyı doğru olarak kesmesini sağlamak sizin sorumluluğunuzda olur. Örneğin, bağlantı kesmeden her çağrıda kuyruk yöneticisine yeni bir bağlantı oluşturan bir kanal çıkışı, bağlantı tanıtıcılarının oluşturulması ve aracı iş parçacıklarının sayısının artmasını sağlar.

Bir çıkış MCA ' nın kendisi ile aynı iş parçacığıda çalışır ve aynı bağlantı tanıtıcısını kullanır. Bu nedenle, MCA ile aynı UOW ' un içinde çalışır ve tutarlılık noktası altında yapılan tüm aramalar, toplu işin sonundaki kanaldan kesinleştirilir ya da yedeklenir.

Bu nedenle, bir kanal ileti çıkışı, özgün iletiyi içeren toplu iş kesinleştirildiğinde, yalnızca o kuyruk için kesinleştirilen bildirim iletileri gönderebilir. Bu nedenle, bir kanal iletileri çıkışından Sync Point MQI çağrıları yayınlanıyor olabilir.

Bir kanal çıkışı, MQCD ' deki alanları değiştirebilir. Ancak, bu değişikliklerin listelendiği durumlar dışında, bu değişiklikler üzerinde işlem yapmamış olabilir. 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 zincirindeki geri kalan çıkışlara ve kanal yönetim ortamını paylaşan herhangi bir konuşmaya geçirilir. Ek bilgi için [Kanal çıkışta MQCD alanlarının değiştirilmesibaşlıklı konuya](#) bakın.

Ayrıca, C kitaplığında yazılan programlar için, yeniden giriş-dışı C kitaplığı işlevi, kanal çıkış programında kullanılmamalıdır.

**Linux** → **UNIX** Aynı anda birden çok kanal çıkış kitaplığı kullanıyorsanız, iki farklı çıkışa ilişkin kod aynı şekilde adlandırılmış işlevler içeriyorsa, bazı UNIX and Linux platformlarında sorunlar ortaya çıkabilir. Bir kanal çıkışı yüklendiğinde, dinamik yükleyici çıkış kitaplığındaki işlev adlarını kitaplığın yüklendiği adreslere çözer. İki çıkış kitaplığı, aynı adlara sahip ayrı işlevler tanımlarsa, bu çözüm süreci, bir kitaplığın işlev adlarını başka bir kitaplığın işlevlerini kullanabilecek şekilde çözebilir. Bu sorun ortaya çıkarsa, bu işlevler etkilenmeden yalnızca gerekli çıkışı ve MQStart işlevlerini dışa aktarması gerektiğini linker ' a belirtin. Diğer işlevlere, kendi çıkış kitaplıklarının dışındaki işlevler tarafından kullanılmamaları için yerel görünürlük verilmelidir. Ek bilgi için bağlantı oluşturucuya ilişkin belgelere bakın.

Tüm çıkışlar, bir kanal çıkış parametresi yapısı (MQCXP), bir kanal tanımlama yapısı (MQCD), hazırlanmış veri arabelleği, veri uzunluğu parametresi ve arabellek uzunluğu parametresiyle çağrılır. Arabellek uzunluğu aşılmamalı:

- İleti çıkışları için, kanal genelinde gönderilmesi gereken en büyük iletiye ve MQXQH yapısının uzunluğuna izin vermelisiniz.
- 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 uzunluğa göre 2 bayt daha az olabilir. Ayrıntılar için MaxSegmentUzunluğu altında döndürülen değeri denetleyin. MaxSegmentLength ile ilgili daha fazla bilgi için bkz. [MaxSegmentUzunluğu](#).

### NetBIOS:

64 KB

### SPX:

64 KB

**Not:** Alıcı kanallarındaki gönderici kanallardan ve gönderici çıkışlardan, TCP için 2 KB arabellekten çıkar çıkar.

- Güvenlik çıkışları için, dağıtılmış kuyruğa alma olanağı 4000 baytlık bir arabellek ayırır.

Çıkışta, ilgili değiştirgelerle birlikte alternatif bir arabellenin döndürülmesi olanaklıdır. Arama ayrıntıları için "[İleti alışverişi kanallarına ilişkin kanal çıkışı programları](#)" sayfa 993 ' e bakın.

## ▶ z/OS Writing channel exit programs on z/OS

You can use the following information to help you write and compile channel-exit programs for z/OS.

Aşağıdaki durumlarda, çıkışlar z/OS LINK tarafından başlatılmış olarak başlatılır:

- Yetkili olmayan sorun programı durumu
- Birincil adres alanı denetim kipi
- Çapraz olmayan bellek kipi
- Erişim olmayan kayıt kipi
- 31 bit adresleme kipi

Bağlantı düzenlenmiş modüller, kanal başlatıcı adres alanı yordamında CSQXLIB DD açıklamasıyla belirlenen veri kümesine yerleştirilmelidir; yükleme modüllerinin adları, kanal tanımlamasındaki çıkış adları olarak belirtilir.

z/OS için kanal çıkışları yazıldığında aşağıdaki kurallar geçerlidir:

- Exits must be written in assembler or C; if C is used, it must conform to the C systems programming environment for system exits, described in the [z/OS C/C++ Programlama Kılavuzu](#).
- Çıkışlar, bir CSQXLIB DD deyimi tarafından tanımlanan yetkili olmayan kitaplıklardan yüklenir. CSQXLIB ' de DISP=SHR değeri sağlanırken, kanal başlatıcı çalışırken de çıkışlar güncellenebilir. Yeni sürüm, kanal yeniden başlatıldığında kullanılır.
- Çıkışlar yeniden girişli olmalı ve sanal saklama alanı içinde herhangi bir yerde çalıştırma yeteneğine sahip olmalıdır.
- Çıkışlar, girişteki bu ortamı yeniden ayarlamalıdır.
- Çıkışlar, elde edilen herhangi bir depolamayı serbest bırakmalı ya da bundan sonraki bir çıkış çağırması tarafından serbest bırakıldığından emin olmalıdır.

Çağrılar arasında kalıcı olacak depolama alanı için z/OS STORAGE hizmetini ya da Sistem Programlama C için 4kmalc kitaplık işlevini kullanın.

Bu işlevle ilgili daha fazla bilgi için bkz. [4kmalc\(\) -- Allocate Page-Hizland Storage](#)(Sayfa Hizalama Depolaması) seçeneğini belirleyin.

- MQCMIT ya da CSQBCMT ve MQBACK ya da CSQBBAK dışındaki tüm IBM MQ MQI çağrılarını kullanılabilir. Bunlar, MQCONN ' un (boş bir kuyruk yöneticisi adı ile birlikte) içerilmesinin ardından bulundurulmalıdır. Bu çağrılar kullanılırsa, çıkışta CSQXSTUB sınırlı kod öbeğiyle bağlantı düzenlenmelidir.

Bu kuralın dışında, güvenlik kanalı çıkışlarının kesinleştirme ve geri alma MQI çağrılarını yayınlanabileceği. Bu tür çağrılar vermek için, MQCMIT ya da CSQBCMT ve MQBACK ya da CSQBBAK yerine CSQXCMT ve CSQXBAK fiillerini kodlayın.

- All exits that use stub CSQXSTUB from IBM WebSphere MQ 7.0 or later must be link-edited in a CSQXLIB load library with format PDS-E.
- Sistem hizmetlerinin kullanılması diğer tüm kanalların ya da diğer tüm kanalların işlenmesini ciddi şekilde etkileyeceğinden, çıkışların bekleme nedeniyle hiçbir sistem hizmeti kullanılmaması gerekir. Birçok kanal tipik olarak tek bir TCB altında çalıştırılır. Bir çıkışta bekleme nedenine neden olan bir şey yaparsanız ve MQXSWT kullanmayacaksa, bu, tüm kanalların beklemesine neden olur. Kanalların bekleme kanallarına neden olması, herhangi bir işlevsel sorun yaşamaz, ancak performans üzerinde olumsuz bir etki yaratmış olabilir. En çok SVC 'ler bekleme işlemlerini içerir, bu nedenle aşağıdaki SVC ' ler dışında bunlardan kaçınmanız gerekir:

- GETMAIN/FREEMAIN/STORAGE
- LOAD/DELETE

Bu nedenle, genel olarak SVC 'ler, kişisel bilgisayarlar ve G/Ç' yi önlemektedir. Bunun yerine MQXWINE çağrısını kullanın.

- Çıkışlar, bağlı oldukları alt görevler dışında ESTAS ya da SPEI ' leri yayınlamaz; çünkü, hata işleme işlemi IBM MQ tarafından gerçekleştirilen hata işleme engeline engel olabilir. Bu, IBM MQ ' un bir hatadan kurtarılamayabileceği ya da çıkış programınızın tüm hata bilgilerini alamayabileceği anlamına gelir.
- MQXMINE çağrısı (bkz. MQXWEKE ) G/Ç ve diğer olayları bekleyen bir bekleme hizmeti sağlar; bu hizmet kullanılırsa, çıkışlar bağ yığını kullanmamalıdır.

G/Ç ve engelleyici olmayan tesisler ya da beklenen bir ECB sağlamayan G/Ç ve diğer tesisler için, ayrı bir alt görev ATTACHED olmalıdır ve MQXWHOD işleminin tamamlanması beklenir; bu tekniğin kullanışsız olduğu işleme nedeniyle, bu olanak yalnızca güvenlik çıkışıyla kullanılmalıdır.

- MQDISC MQI çağrısı, çıkış programı içinde örtük kesinleştirmenin gerçekleştirilmesine neden olmaz. Kanal işlemi için kesinleştirme işlemi yalnızca kanal protokolü gerektirdiğinde gerçekleştirilir.

Aşağıdaki çıkış örnekleri IBM MQ for z/OS ile sağlanır:

#### **CSQ4BAX0**

Bu örnek çevirici olarak yazılmıştır ve MQXWEKE kullanımını gösterir.

#### **CSQ4BCX1 ve CSQ4BCX2**

Bu örnekler C ' de yazılır ve parametrelere nasıl erişileceğini gösterir.

#### **CSQ4BCX3 ve CSQ4BAX3**

Bu örnekler sırasıyla C ve derleyici olarak yazılır.

SCSQAUTH LOADLIB içine önceden derlenmiş olan CSQ4BCX3 örneği, çıkışta hiçbir değişiklik gerekmemesiyle işlev görmelidir. Bir LOADLIB (örneğin, MY.TEST.LOADLIB)) ve SCSQAUTH (CSQ4BCX3) üyesini bu kopyaya kopyalayın.

İstemci bağlantısında bir güvenlik çıkışı ayarlamak için aşağıdaki yordamı kullanın:

1. Kanal başlatıcı tarafından kullanılan kullanıcı kimliği için geçerli bir OMVS kesimi oluşturun.

This allows the IBM MQ for z/OS channel initiator to use TCP/IP with the UNIX System Services (USS) socket interface, in order to facilitate exit processing. Bağlantı kuran herhangi bir istemcinin kullanıcı kimliği için bir OMVS kesiminin tanımlanmasında gereksiz olduğunu unutmayın.

2. Çıkış kodunun yalnızca program tarafından denetlenen bir ortamda çalıştırıldığından emin olun.

Bu, CHINIT adres alanına yüklenen her şeyin, program tarafından denetlenen bir kitaplıktan (STEPLIB içindeki tüm kitaplıkların anlamı) yüklenmesi ve CSQXLIB ve

```
++hlq++.SCSQANLx
++hlq++.SCSQMVR1
++hlq++.SCSQAUTH
```

Bir yükleme kitaplığını program denetimli olarak ayarlamak için, bu örneğe benzer bir komut kullanın:

```
RALTER PROGRAM * ADDMEM('MY.TEST.LOADLIB'//NOPADCHK)
```

Daha sonra, şu komutu vererek program denetimli ortamı etkinleştirebilir ya da yenileyebilirsiniz:

```
SETRPTS WHEN(PROGRAM) REFRESH
```

3. Aşağıdaki komutu girerek, LOADLIB çıkışı CSQXLIB DD ' ye (CHINIT başlangıç yordamında) ekleyin:

```
ALTER CHANNEL(xxxx) CHLTYPE(SVRCONN)SCYEXIT(CSQ4BCX3)
```

Bu, adı belirtilen kanala ilişkin 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.

See [IBM MQ for z/OS sunucusu bağlantı kanalı](#) for further information on setting up a security exit using the sample CSQ4BCX3.

#### CSQ4BCX4

Bu örnek C 'de yazılmıştır ve MQCXP' de **RemoteProduct** ve **RemoteVersion** alanlarının kullanılmasını gösterir.

#### İlgili kavramlar


[IBM MQ for z/OS sunucusu bağlantı kanalı](#)

[“Writing channel exit programs on IBM i” sayfa 999](#)

You can use the following information to help you write and compile channel-exit programs for IBM i.

[“Writing channel-exit programs on UNIX, Linux, and Windows” sayfa 1000](#)

You can use the following information to help you write channel-exit programs for UNIX, Linux, and Windows systems.

 [Writing channel exit programs on IBM i](#)

You can use the following information to help you write and compile channel-exit programs for IBM i.

Çıkış, ILE C, ILE RPG ya da ILE COBOL dillerinde yazılmış bir program nesnesidir. Çıkış programı adları ve bunların kitaplıkları kanal tanımında adlandırılır.

Bir çıkış programı yaratırken ve derlerken aşağıdaki koşulları göz önünde bulundurun:

- Program, iş parçacığı güvenli kılınmalı ve ILE C, ILE RPG ya da ILE COBOL derleyicisiyle yaratılmış olmalıdır. ILE RPG için, THREAD (\*SERIALIZE) denetim belirtimini ve ILE COBOL için, PROCESS deyimindeki THREAD seçeneği için SERIALIZE belirtmeniz gerekir. Ayrıca, programların iş parçacıklı IBM MQ kitaplıklarına da bağlanması gerekir: ILE C ve ILE RPG durumunda QMQM/LIBMQM\_R ve ILE COBOL durumunda AMQ0STUB\_R . RPG ya da COBOL uygulamalarının iş parçacığı güvenliğini sağlamak hakkında ek bilgi için, dil için uygun Programcı Kılavuzu 'na bakın.
- IBM MQ for IBM i , çıkış programlarının teraspace desteği için etkinleştirilmesini gerektirir. (Teraspace is a form of shared memory introduced in OS/400 V4R4.) For the ILE RPG and COBOL compilers, any programs compiled on OS/400 V4R4 or later are so enabled. C için, programlar, CRTCMOD ya da CRTBNDK komutlarında belirlenen TERASPACE (\*YES \*TSIFC) seçenekleriyle derlenmelidir.
- Bir işaretçiyi kendi arabellek alanına döndüren bir çıkış, nesnenin, kanal çıkış programının zaman dilinden sonra var olduğunu doğrulamalıdır. Gösterge, program yığınındaki bir değişkenin adresi ya da program öbeğindeki bir değişkenin adresi olamaz. Bunun yerine, işaretçinin sistemden elde edilmesi

gerekir. Örnek, kullanıcı çıkışında yaratılan bir kullanıcı alanıdır. Program sona erdiğinde, kanal çıkışı programı tarafından ayrılan veri alanının hala MCA için kullanılabilmesini sağlamak için, kanal çıkışı, çağırana ya da adlandırılmış bir etkinleştirme grubunun etkinleştirme grubunda çalıştırılmalıdır. Bu işlemi, CRTPGM 'deki ACTGRP parametresini kullanıcı tanımlı bir değere ya da \*CALLER değerine ayarlayarak yapın. Program bu şekilde yaratılırsa, kanal çıkış programı dinamik bellek ayırabilir ve bu belleğin bir işaretçisini MCA 'ya iletebilirler.

### İlgili kavramlar

[“Writing channel-exit programs on UNIX, Linux, and Windows” sayfa 1000](#)

You can use the following information to help you write channel-exit programs for UNIX, Linux, and Windows systems.

[“Writing channel exit programs on z/OS” sayfa 997](#)

You can use the following information to help you write and compile channel-exit programs for z/OS.

**ULW** *Writing channel-exit programs on UNIX, Linux, and Windows*

You can use the following information to help you write channel-exit programs for UNIX, Linux, and Windows systems.

[“Writing exits and installable services on UNIX, Linux and Windows” sayfa 968](#) içinde belirtilen yönergeleri izleyin. Uygun durumlarda, aşağıdaki kanal çıkışından özel bilgileri kullanın:

Çıkış C 'de yazılmalı ve Windows' da bir DLL olmalıdır.

Define a dummy MQStart() routine in the exit and specify MQStart as the entry point in the library. [Şekil 111 sayfa 1000](#) , programınıza bir girdi nasıl ayarlayacağını gösterir:

```
#include <cmqec.h>

void MQStart() {} /* dummy entry point - for consistency only */
void MQENTRY ChannelExit ( PMQCDX  pChannelExitParms,
                          PMQCD    pChannelDefinition,
                          PMQLONG  pDataLength,
                          PMQLONG  pAgentBufferLength,
                          PMQVOID  pAgentBuffer,
                          PMQLONG  pExitBufferLength,
                          PMQPTR   pExitBufferAddr)
{
  ... Insert code here
}
```

*Şekil 111. Kanal çıkışa ilişkin örnek kaynak kodu*

Visual C ++ kullanılarak Windows için kanal çıkışları yazarken, kendi DEF dosyanızı yazmanız gerekir. [Şekil 112 sayfa 1000](#) 'ta nasıl gösterildiğini gösteren bir örnek. Kanal çıkışı programlarının yazılmasına ilişkin ek bilgi için [“Kanal-çıkış programları yazılıyor” sayfa 996](#) başlıklı konuya bakın.

```
EXPORTS
ChannelExit
```

*Şekil 112. Windows için örnek DEF dosyası*

### İlgili kavramlar

[“Writing channel exit programs on IBM i” sayfa 999](#)

You can use the following information to help you write and compile channel-exit programs for IBM i.

[“Writing channel exit programs on z/OS” sayfa 997](#)

You can use the following information to help you write and compile channel-exit programs for z/OS.

#### *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 doğrulama kayıtları , belirli kullanıcılardan ve kanallardan kuyruk yöneticilerine erişimin önlenmesinde ve uzak kullanıcıların IBM MQ kullanıcı kimliklerine eşlenmesinde büyük esneklik sağlar. TLS desteği, kullanıcılarınızın kimliğini doğrulamak ve verileriniz için şifreleme ve veri bütünlüğü denetlemesi sağlamak için IBM MQ tarafından sağlanır. TLS hakkında daha fazla bilgi için bkz. IBM MQ'deki TLS güvenliği iletişim kuralları. Ancak, yine de güvenlik işlemleri için daha karmaşık (ya da farklı) formlara ve diğer denetim türlerine ve güvenlik bağlamına gerek duyuyorsanız, güvenlik çıkışlarını yazmayı düşünün.

For security exits written prior to IBM WebSphere MQ 7.1 it is worth noting that earlier versions of IBM MQ queried the underlying secure sockets provider (e.g. GSKit) to determine the remote partner's certificate Subject Distinguished Name (SSLPEER) and Issuer Distinguished Name (SSLCERTI). In IBM WebSphere MQ 7.1 support was added for a range of new security attributes. Bu özniteliklere erişmek için IBM WebSphere MQ 7.1 , sertifikanın DER kodlamasını alır ve Konu ve Sertifika Veren DN ' ini belirlemek için bu kodlamayı kullanır. Aşağıdaki kanal durumu özniteliklerinde Konu ve Sertifika Veren DN öznitelikleri 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, listelenen kanal güvenliği çıkışlarına aktarılan verilerin de döndürülmesini sağlar:

- MQCD SSLPeerNamePtr
- MQCXP SSLRemCertIssNamePtr

IBM WebSphere MQ 7.1'ta, bir SERIALNUMERT özniteliği, Subject DN' de de yer alır ve uzak ortağın sertifikasına ilişkin seri numarasını içerir. Ayrıca, bazı DN öznitelikleri önceki yayın düzeylerinden farklı bir sırada döndürülür. Sonuç olarak, SSLPEER ve SSLCERTI alanlarının bileşimi, önceki yayın düzeylerinden IBM WebSphere MQ 7.1 ' de değiştirilir ve bu nedenle, bu alanlara bağımlı olan tüm güvenlik çıkışlarının ya da uygulamaların incelenip güncellenmemesine neden olur.

Bir kanal tanımlamasının SSLPEER alanı aracılığıyla belirtilen var olan IBM MQ eşdüzey ad süzgeçleri etkilenmez ve önceki yayınlarla aynı şekilde çalışmaya devam eder. Bunun nedeni, IBM MQ eş adı eşleştirme algoritmasının, kanal tanımlamalarını değiştirmeye gerek kalmaksızın var olan SSLPEER süzgeçlerini işlemek üzere güncellendiğinden kaynaklanır. Bu değişiklik büyük olasılıkla güvenlik çıkışlarını ve PCF programlama arabirimi tarafından döndürülen Sertifika Sahibi DN 'si ve Sertifika Veren DN değerlerine bağımlı olan uygulamaları etkiler.

C ya da Java'inde bir güvenlik çıkışı yazılabilir.

Kanal güvenlik çıkış programları, MCA ' nın işlem çevriminde aşağıdaki yerlerde çağrılır:

- MCA ' da kabul ve sonlandırma.
- Kanal başlatma sırasında ilk veri kararlaştığı sona erdikten hemen sonra. Kanalın alıcı ya da sunucu ucu, uzak uçta güvenlik çıkışa teslim edilmesi için bir ileti sağlayarak uzak ucuyla bir güvenlik ileti değişik tokası başlatabilir. Ayrıca, bunu yapmayı da reddedebilir. Çıkış programı, uzak uçtan alınan her güvenlik iletiyi işlemek için yeniden başlatılır.
- Kanal başlatma sırasında ilk veri kararlaştığı sona erdikten hemen sonra. Kanalın gönderen ya da istekte bulunan ucu, uzak uçtan alınan bir güvenlik iletiyi işlerken ya da uzak sona erdirilemeyeceğini bildiren bir güvenlik değişik tokadı başlatır. Çıkış programı, alınabilecek sonraki tüm güvenlik iletilerini işlemek için yeniden başlatılır.

Bir istekte bulunanın kanalı hiçbir zaman MQXR\_INIT\_SEC ile çağrılmaz. Kanal, sunucuya bir güvenlik çıkış programı olduğunu bildirir ve sunucu, daha sonra bir güvenlik çıkışı başlatma olanağına sahiptir. Bir değer yoksa, istekte bulunana bildirir ve çıkış programına sıfır uzunluklu bir akış döndürülür.

**Not:** Sıfır uzunluklu güvenlik iletileri göndermekten kaçının.

Güvenlik çıkışı programlarıyla değişik tokuş edilen verilerin örnekleri Şekil 113 sayfa 1002 ile Şekil 116 sayfa 1004arasındaki şekillerle gösterilir. Bu örnekler, alıcısının güvenlik çıkışı ile gönderenin güvenlik çıkışıyla ilgili olayların sırasını gösterir. Rakamlardaki ardışık satırlar, zaman geçişini temsil eder. Bazı durumlarda, alıcı ve göndericindeki olaylar ilintili değildir ve bu nedenle aynı anda ya da farklı zamanlarda

gerçekleşebilir. Diğer durumlarda, bir çıkış programındaki bir olay, diğer çıkış programında daha sonra oluşan tamamlayıcı bir olaya neden olur. Örneğin, Şekil 113 sayfa 1002’inde:

1. Alıcı ve gönderen her biri MQXR\_INIT ile çağrılır, ancak bu çağrılar ilintili değildir ve bu nedenle aynı anda ya da farklı zamanlarda ortaya çıkabilir.
2. Alıcı bir sonraki MQXR\_INIT\_SEC ile çağrılır, ancak gönderici çıkışında tamamlayıcı bir olay gerektirmeyecek MQXCC\_OK değerini döndürür.
3. Gönderen bir sonraki MQXR\_INIT\_SEC ile çağrılır. Bu, alıcının MQXR\_INIT\_SEC ile çağrısıyla ilintili değildir. Gönderen, alıcı çıkışında tamamlayıcı bir olaya neden olan MQXCC\_SEND\_SEC\_MSG değerini döndürür.
4. Daha sonra, alıcı MQXR\_SEC\_MSG ile çağrılır ve gönderici çıkışında tamamlayıcı bir olaya neden olan MQXCC\_SEND\_SEC\_MSG döndürülür.
5. Daha sonra, gönderen MQXR\_SEC\_MSG ile çağrılır ve alıcı çıkışında tamamlayıcı bir olay gerektirmeyecek MQXCC\_OK değerini 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
<i>Message transfer begins</i>	

Şekil 113. Sözleşme ile gönderici 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_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
	<i>Channel closes</i>
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Şekil 114. Sözleşme olmadan gönderen ile başlatılan deęiş 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	
<i>Message transfer begins</i>	
Invoked with MQXR_TERM Responds with MQXCC_OK	Invoked with MQXR_TERM Responds with MQXCC_OK

Şekil 115. Alıcı ile sözleşme ile 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	
<i>Channel closes</i>	


Şekil 116. Sözleşme olmadan alıcı tarafından başlatılan değiş tokuş

Kanal güvenlik çıkış programı, güvenlik çıkışı tarafından oluşturulan iletim üstbilgileri dışında, güvenlik verilerini içeren bir aracı arabelleğinden geçirilir. Bu veriler, kanaldan her iki ucunun güvenlik doğrulaması gerçekleştirilebilmesi için uygun bir veri olabilir.

İleti kanalının gönderme ve alma sonundaki güvenlik çıkış programı, herhangi bir çağrıya iki yanıt kodundan birini döndürebilir:

- Güvenlik deęiş tokası hata olmadan sona erdi
- Kanalı engelle ve kapat

#### Not:

1. Kanal güvenlik çıkışları genellikle çiftler halinde çalışır. Uygun kanalları tanımladığınızda, kanalın her iki ucu için uyumlu çıkış programlarının adlandırıldığından emin olun.
2.  IBM i 'ta, Use adopted authority (USEADPAUT = \*YES) ile derlenmiş güvenlik çıkış programları QMQM ya da QMQMADM yetkisini benimseyebilir. Çıkışa dikkat edin, bu özellięi sisteminizin güvenlik riski taşıması için kullanmaz.
3. Kanalın dięer ucunun sertifikası sağladığı bir TLS kanalında, güvenlik çıkışı, SSLPeerNamePtr tarafından erişilen MQCD alanına ve SSLRemCertIssNamePtr tarafından erişilen MQCXP alanına verenin Ayırt Edici Adı ile bu sertifikanın konularının ayırt edici adını alır. Bu adın koyabileceęi kullanım alanları şunlardır:
  - Erişimi TLS kanalı üzerinden kısıtlamak için.
  - MQCD.MCAUserIdentifier adı temelinde.

#### İlgili kavramlar

[Kanal doğrulama kayıtları](#)

[Transport Layer Security \(TLS\) kavramları](#)

*Güvenlik çıkışı yazılıyor*

güvenlik çıkış iskelet kodunu kullanarak bir güvenlik çıkışı yazabilirsiniz.

[Şekil 117 sayfa 1005](#) , bir güvenlik çıkışının nasıl yazılacağını gösterir.

```
void MQENTRY MQStart() {}  
void MQENTRY EntryPoint (PMQVOID pChannelExitParms,  
                          PMQVOID pChannelDefinition,  
                          PMQLONG pDataLength,  
                          PMQLONG pAgentBufferLength,  
                          PMQVOID pAgentBuffer,  
                          PMQLONG pExitBufferLength,  
                          PMQPTR pExitBufferAddr)  
{  
    PMQCXP pParms = (PMQCXP)pChannelExitParms;  
    MQCD pChDef = (MQCD)pChannelDefinition;  
    /* TODO: Add Security Exit Code Here */  
}
```

Şekil 117. Güvenlik çıkışı İskelet kodu

Standart IBM MQ Giriş Noktası MQStart var olmalıdır, ancak herhangi bir işlevi gerçekleştirmek için gerekli değildir. İşleve ilişkin ad (bu örnekteEntryPoint ) değiştirilebilir, ancak kitaplık derlenip bağlandığında işlev dışa aktarılmalıdır. Önceki örnekte olduğu gibi, işaretçiler pChannelExitParms , PMQCXP ve pChannelDefinition için MQCD ' ye dönüştürülmelidir. Kanal çıkışlarının çağrılmasına ve parametrelerin kullanılmasına ilişkin genel bilgiler için [MQ\\_CHANNEL\\_EXIT](#) başlıklı konuya bakın. Bu deęiştirgeler bir güvenlik çıkışta aşağıdaki gibi kullanılır:

#### PMQVOID pChannelExitParms

giriş/çıkış

MQCXP yapısına ilişkin gösterge, alanlara erişmek için PMQCXP ' ye çevrilecek. Bu yapı, Çıkış ve MCA arasında iletişim kurmak için kullanılır. MQCXP ' deki aşağıdaki alanlar, Security Exits ile ilgili özel ilgi alanlarıdır:

**ExitReason**

Security Exit 'e güvenlik alışverişindeki mevcut durumu bildirir ve hangi işlemin yapılması kararlaştırılırken kullanılır.

**ExitResponse**

Güvenlik alışverişindeki bir sonraki aşamayı belirleyen MCA ' ya verilen yanıt.

**ExitResponse2**

MCA ' nın Güvenlik Çıkışı yanıtını nasıl yorumlayacağını yönetmek için ek denetim işaretleri.

**ExitUserAlanı**

Aramalar arasında durumu korumak için Security Exit (Güvenlik Çıkışı) tarafından kullanılabilir 16 bayt (üst sınır).

**ExitData**

Kanal tanımlamasının SCYDATA alanında belirlenen verileri (boşluklarla sağa doldurulmuş 32 bayt) içerir.

**PMQVOID pChannelTanımlaması**

giriş/çıkış

Alanlara erişmek için MQCD yapısı-PMQCD ' ye dönüştürme yapın. Bu değiştirge, kanala ilişkin tanımlamayı içerir. MQCD ' deki aşağıdaki alanlar, Security Exits ile ilgili özel ilgi alanlarıdır:

**ChannelName**

Kanal adı (boşluklarla sağa doğru 20 bayt doldurulur).

**ChannelType**

Kanal tipini tanımlayan bir kod.

**MCA Kullanıcı Kimliği**

Bu üç alan grubu, kanal tanımında belirlenen MCAUSER alanı değeriyle ilk kullanıma hazırlandı. 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**

İlk 12 baytlık tanıtıcı boşluklarla dolduruldu.

**LongMCAUserIntPtr**

Tam uzunluk tanıtıcısını içeren bir arabelleğe (garantili boş değer sonlandırılmamış) ilişkin gösterge, MCAUserIdentifier' un üzerinde önceliği alır.

**LongMCAUserIdLength**

Length of string pointed to by LongMCAUserIntPtr - must be set if LongMCAUserIntPtr is set.

**Uzak Kullanıcı Tanıtıcısı**

Yalnızca CLNTCONN/SVRCONN kanal çiftleri için geçerlidir. CLNTCONN Security Exit tanımlanmadıysa, bu üç alan istemci MCA tarafından başlatılır; böylece, kimlik doğrulama için bir SVRCONN Güvenlik Çıkışı ve MCA Kullanıcı Tanıtıcısı belirtilirken kullanılabilir istemci ortamından bir kullanıcı kimliği içerebilirler. Bir CLNTCONN Security Exit tanımlıysa, bu alanlar kullanıma hazırlanmaz ve CLNTCONN Security Exit ile ayarlanabilir ya da güvenlik iletileri, bir kullanıcı kimliğini İstemciden Sunucuya geçirmek için kullanılabilir.

**RemoteUserTanıtıcısı**

İlk 12 bayt tanıtıcısı boşluklarla dolduruldu.

**LongRemoteUserIntPtr**

Pointer to a buffer containing the full length identifier (not guaranteed null terminated) takes priority over RemoteUserIdentifier.

**LongRemoteUserIdUzunluğu**

LongRemoteUserIntPtr-byPtrPtr ayarlandıysa, LongRemoteUserIntPtr tarafından işaret edilen dizgi uzunluğu ayarlanmalıdır.

**PMQUT pDataUzunluğu**

giriş/çıkış

İşaretçiyi MQUZE. Security Exit (Güvenlik Çıkışı) çağrısının ardından AgentBuffer (AgentArabelleği) içinde bulunan herhangi bir Güvenlik Çıkışı uzunluğunu içerir. Must be set by a Security Exit to the length of any message being sent in the AgentBuffer or ExitBuffer.

### **PMQHOT pAgentBufferLength**

Giriş

İşaretçiyi MQUZE. Güvenlik çıkışı çağrılırken, AgentBuffer içinde yer alan verilerin uzunluğu.

### **PMQVOID pAgentArabelleği**

giriş/çıkış

Güvenlik Çıkışı çağrılırken, bu, ortak çıkıştan gönderilen herhangi bir iletiye işaret eder. MQCXP yapısında ExitResponse2 varsa, MQXR2\_USE\_AGENT\_BUFFER işaret kümesi (varsayılan değer) varsa, bir Güvenlik Çıkışı 'nın gönderilmekte olan ileti verilerini göstermek için bu değiştirgeyi ayarlamaya gerek vardır.

### **PMQlong pExitBufferLength**

giriş/çıkış

İşaretçiyi MQUZE. Bu parametre, bir Security Exit 'in ilk çağrısında 0 ile başlatılır ve döndürülen değer, güvenlik değiş tokası sırasında Güvenlik Çıkışı çağrılarında tutulur.

### **PMQPTR pExitBufferAddr**

giriş/çıkış

Bu değiştirge, bir Security Exit 'in ilk çağrısında boş değerli bir işaretçi olarak başlatılır ve döndürülen değer, güvenlik değiş tokası sırasında Güvenlik Çıkışı çağrılarında tutulur. If the MQXR2\_USE\_EXIT\_BUFFER flag is set in the ExitResponse2 in the MQCXP structure then a Security Exit needs to set this parameter to point to any message data being sent.

*CLNTCONN/SVRCONN kanal çiftlerinde ve diğer kanal çiftlerinde tanımlanan güvenlik çıkışlarıyla ilgili davranış farklılıkları*

Güvenlik çıkışları, tüm kanal tiplerinde tanımlanabilir. Ancak, CLNTCONN/SVRCONN kanal çiftlerinde tanımlanan güvenlik çıkışlarının işleyişi, diğer kanal çiftlerinde tanımlanan güvenlik çıkışlarından biraz farklıdır.

Bir CLNTCONN kanalındaki bir Güvenlik Çıkışı, iş ortağı SVRCONN çıkışa göre işlenmek üzere kanal tanımlamasındaki Uzak Kullanıcı Tanıtıcısını ya da SVRCONN Güvenlik Çıkışı tanımlanmadıysa ve SVRCONN ' un MCAUSER alanı ayarlanmadıysa OAM yetkilendirmesi olarak ayarlayabilir.

CLNTCONN Security Exit tanımlanmadıysa, kanal tanımlamasındaki Uzak Kullanıcı Tanıtıcısı istemci ortamından (boş olabilir) istemci ortamından bir kullanıcı kimliği olarak ayarlanır.

SSVRCONN Security Exit, MQXCC\_OK ExitResponse değerini döndürdüğünde, CLNTCONN ve SVRCONN kanal çiftinde tanımlanan Güvenlik Exits ile SVRCONN kanal çiftinde tanımlanan bir güvenlik değiş tokası başarıyla tamamlanır. Diğer kanal çiftleri arasındaki bir güvenlik değiş tokası başarıyla tamamlanırsa, değiş tokuş işlemi başlatan Security Exit, MQXCC\_OK için ExitResponse (ExitResponse) değerini döndürür.

Ancak, güvenlik değiş tokuşunu devam ettirmeye zorlamak için MQXCC\_SEND\_AND\_REQUEST\_SEC\_MSG ExitResponse kodu kullanılabilir: MQXCC\_SEND\_AND\_REQUEST\_SEC\_MSG çizelgesinin bir ExitResponse bir CLNTCONN ya da SVRCONN Security Exit tarafından döndürülürse, iş ortağı çıkışı bir güvenlik iletilisi göndererek yanıt vermelidir (MQXCC\_OK ya da boş bir yanıt değil) ya da kanal sonlandırılır. For Security Exits defined on other types of channel, an ExitResponse of MQXCC\_OK returned in response to a MQXCC\_SEND\_AND\_REQUEST\_SEC\_MSG from the partner Security Exit results in continuation of the security exchange as if a null response was returned and not in termination of the channel.

### *SSPI güvenlik çıkışı*

IBM MQ for Windows , Security Services Programming Interface (SSPI) olanağını kullanarak IBM MQ kanalları için kimlik doğrulaması sağlayan bir güvenlik çıkışı sağlar. The SSPI provides the integrated security facilities of Windows.

Bu güvenlik çıkışı hem IBM MQ istemcisi hem de IBM MQ sunucusu içindir.

Güvenlik paketleri security.dll ya da secur32.dll'inden yüklenir. Bu DLL 'ler işletim sisteminiz ile birlikte sağlanır.

One-way authentication is provided on Windows, using NTLM authentication services. Two-way authentication is provided on Windows 2000, using Kerberos authentication services.

Güvenlik çıkış programı kaynak ve nesne biçiminde sağlanır. Nesne kodunu olduğu gibi kullanabilir ya da kaynak kodu, kendi kullanıcı çıkış programlarınızı yaratmak için başlangıç noktası olarak kullanabilirsiniz. SSPI güvenlik çıkışının nesne ya da kaynak kodunun kullanılmasına ilişkin ek bilgi için bkz. [“Windows üzerindeki SSPI güvenlik çıkışının kullanılması” sayfa 1167](#)

#### *Kanal gönderme ve alma çıkış programları*

Veri sıkıştırma ve açma gibi görevleri gerçekleştirmek için gönderme ve alma çıkışlarını kullanabilirsiniz. Gönderme ve alma çıkış programlarının art arda çalıştırılacağı bir listesini belirtebilirsiniz.

Kanal gönderme ve alma çıkış programları, MCA 'nın işlem çevriminde aşağıdaki yerlerde çağrılır:

- Gönderme ve alma çıkış programları, MCA başlatma sırasında kullanıma hazırlama ve MCA sonlandırmasında sona erdirmeye için çağrılır.
- Çıkış gönderme programı, bir ileti aktarımın gönderileceği uca bağlı olarak, kanalın bir ya da başka bir ucunda çağrılır ve bağlantı üzerinden bir iletim gönderilmeden hemen önce gönderilir. Not 4, ileti kanalları iletileri tek bir yöne gönderse de, çıkışlar neden her iki yönde de kullanılabilir olduğunu açıklar.
- Alma çıkış programı, bir ileti aktarımı için bir iletim alındığı sona erme durumuna bağlı olarak, kanalın bir ya da diğer ucunda çağrılır ve bağlantıdan hemen bir iletim alınır. Not 4, ileti kanalları iletileri tek bir yöne gönderse de, çıkışlar neden her iki yönde de kullanılabilir olduğunu açıklar.

Bir ileti aktarımı için birçok iletim olabilir ve bir ileti alan uca ileti çıkışa ulaşmadan önce, gönderme ve alma çıkış programlarının birçok yinelenmesi olabilir.

Kanal gönderme ve alma çıkış programları, iletişim bağlantısından gönderilen ya da alınan iletim verilerini içeren bir aracı arabelleğinden geçirilir. Çıkış programları göndermek için, arabelleğin ilk 8 baytı MCA tarafından kullanılmak üzere ayrılır ve değiştirilmemelidir. Program farklı bir arabellek döndürürse, bu ilk 8 byte yeni arabellekte var olmalıdır. Çıkış programlarına sunulan verilerin biçimi tanımlı değil.

Çıkış programları gönderme ve alma yoluyla iyi bir yanıt kodu döndürülebilir. Diğer herhangi bir yanıt MCA 'nın olağandışı bitmesine (olağandışı bitme) neden olur.

**Not:** Gönderme ya da alma çıkışından tutarlılık noktası içinde bir MQGET, MQPUT ya da MQPUT1 çağrısı yayınlanmayın.

#### **Not:**

1. Gönderme ve alma çıkışları genellikle çiftler halinde çalışılır. Örneğin, bir gönderme çıkışı, verileri sıkıştırabilir ve bir alma çıkışı sıkıştırabilir ya da bir gönderme çıkışı verileri şifreleyebilir ve bir alma çıkışı bunu çözebilir. Uygun kanalları tanımladığınızda, kanalın her iki ucu için uyumlu çıkış programlarının adlandırıldığından emin olun.
2. Kanal için sıkıştırma açıksa, çıkışlar sıkıştırılmış veriler iletilir.
3. Kanal gönderme ve alma çıkışları, uygulama verileri (örneğin, durum iletileri) dışındaki ileti kesimleri için çağrılabilir. Bunlar, başlatma iletişim kutusu sırasında çağrılmaz ya da güvenlik denetimi evresidir.
4. İleti kanalları iletileri tek bir yönde gönderse de, kalp atışları ve toplu iş işlemlerinin sonu gibi kanal denetimi verileri her iki yöne doğru akar ve bu çıkışlar her iki yönde de kullanılabilir. Ancak, ilk kanal başlatma veri akışlarından bazıları, çıkışlardan herhangi biri tarafından işlenmekten muaf tutulmakta.
5. Gönderme ve alma çıkışlarının sırayla çağrılacağı durumlar vardır; örneğin, bir dizi çıkış programı çalıştırıyorsanız ya da güvenlik çıkışlarını çalıştırıyorsanız. Daha sonra, alma çıkışı veri işlemek için ilk çağrıldığında, ilgili gönderme çıkıştan geçmemiş veriler alabilir. Alma çıkışı işlemi az önce gerçekleştirdiyse, örneğin, açma işleminin gerekli olduğunu denetlemeden, açma işlemi beklenmeyen bir şekilde sonuçlanabilir.

Gönderme ve alma çıkışlarınızı, alma çıkışı, aldığı verilerin ilgili gönderme çıkışıyla işlenip işlendiğini denetleyebilmesi için çıkış kodlarına gerek duyarsınız. Bunun için önerilen yol, çıkış programlarınızı kodlamak olmalıdır.



- Çıkış gönderme işlemi, işlemi gerçekleştirmeden önce, dokuzuncu bayt veri değerini 0 olarak ayarlar ve tüm verileri 1 bayt boyunca kaydırır. (İlk 8 byte, MCA tarafından kullanılmak üzere ayrılmıştır.)
- Alma çıkışı bayt 9 'da 0 olan verileri alırsa, veri gönderme çıkışından geldiğini bilir. 0 'ı kaldırır, tamamlayıcı işlemi gerçekleştirir ve sonuçta elde edilen verileri 1 bayt geriye kaydırır.
- Alma çıkışı, bayt 9 'da 0 dışında bir değer içeren verileri alırsa, gönderme çıkışının çalıştırılmadığını varsayar ve verileri arayanın geri göndermesine geri gönderir.

Güvenlik çıkışlarını kullanırken, kanal güvenlik çıkışı tarafından sona erdirilirse, ilgili alma çıkışı olmadan bir gönderme çıkışı çağrılabilir. Bu sorunun önüne geçmenin bir yolu da, güvenlik çıkışını MQCD.SecurityUserData ya da MQCD.SendUserData içinde bir işaret ayarlamak için kodlamak; örneğin, çıkış kanalı sona erdirmeye karar verince. Daha sonra, çıkış gönderme işleminin bu alanı denetmesi ve yalnızca işaret ayarlanmadıysa verileri işlemesi gerekir. Bu denetim, verilerin gereksiz yere değiştirilmesine neden olmasını önler ve güvenlik çıkışı değiştirilmiş verileri aldıysa, 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. MQXCP . *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/OS üzerinde desteklenmez.

Gönderme çıkışının ayrılmış alanın tümünü kullanması gerekmez. *ExitSpace* bayttan az ya da iletim arabelleği dolu değilse, çıkış ayrılan miktardan daha fazla kullanılabilir. *ExitSpacedeğerini* ayarlarken, iletim arabelleğinde 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 MQ tarafından 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 ayırıyor.

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 ileti çıkış programları*

Kanal ileti çıkışını kullanarak, şifreleme, gelen kullanıcı kimliklerinin doğrulanması ya da ikamesi, ileti verileri dönüştürme, günlük kaydı ve başvuru ileti işleme gibi görevleri gerçekleştirmek için kullanabilirsiniz. Art arda çalıştırılacak ileti çıkış programlarının bir listesini belirtebilirsiniz.

Kanal ileti çıkış programları MCA ' nın işlem çevriminde aşağıdaki yerlerde çağrılır:

- MCA başlatma ve sonlandırma sırasında
- MCA gönderdikten hemen sonra bir MQGET çağrısı yayınlandıktan sonra
- Alma MCA ' nın bir MQPUT çağrısı yayınlamadan önce


İleti çıkışı, MQXQH iletim kuyruğu üstbilgisini ve kuyruktan alınan uygulama ileti metnini içeren bir aracı arabelleğinden geçirilir. MQXQH biçimi, [MQXQH-Transmission-queue header](#)' da verilmiştir.

Başvuru iletileri (yani, gönderilecek başka bir nesneyi gösteren bir üstbilgi içeren iletiler) kullanıyorsanız, ileti çıkışı üstbilgiyi tanır, MQRMH. Nesneyi tanımlar, uygun şekilde alır, uygun şekilde üstbilgiye ekler ve alma MCA 'ya iletilmesi için MCA' ya iletir. Alıcı MCA ' da, başka bir ileti çıkışı bu iletinin bir başvuru ileti olduğunu algılar, nesneyi çeker ve üstbilgiyi hedef kuyruğa aktarır. Başvuru iletilerine ve bunları işleten bazı örnek ileti çıkışlarına ilişkin ek bilgi için “Başvuru iletileri” sayfa 831 ve “Başvuru İletisi örneklerinin çalıştırılması” sayfa 1138 başlıklı konuya bakın.

İleti çıkışları aşağıdaki yanıtları döndürebilir:

- İletiyi gönderin (GET çıkışı). Çıkış, çıkış tarafından değiştirilmiş olabilir. (Bu, MQXCC\_OK değerini döndürür.)
- İletiyi kuyruğa koyun (PUT çıkışı). Çıkış, çıkış tarafından değiştirilmiş olabilir. (Bu, MQXCC\_OK değerini döndürür.)
- İletiyi işlemeyin. İleti, MCA tarafından gönderilen ileti kuyruğunda (teslim edilmemiş ileti kuyruğu) yerleştirilir.
- Kanalı kapatın.
- MCA ' nın olağandışı sona ermesine neden olan dönüş kodu hatalı.

#### **Not:**

1. İleti, aktarılan her tam ileti için bir kez çağrılır ve ileti parçalara bölünse bile.
2.  If you provide a message exit on UNIX or Linux, the automatic conversion of user IDs to lowercase characters (described [burayı tıklatın](#)) does not operate.
3. Bir çıkış MCA ' nın kendisi ile aynı iş parçacığıda çalışır. Aynı çalışma birimi (UOW) içinde aynı bağlantı tanıtıcısını kullandığı için aynı çalışma birimi (UOW) içinde de çalışır. Bu nedenle, eşitleme noktası altında yapılan tüm aramalar, toplu işin sonundaki kanaldan kesinleştirilir ya da yedeklenir. Örneğin, bir kanal ileti çıkışı programı, başka bir kanala bildirim iletileri gönderebilir ve bu iletiler, yalnızca özgün iletiyi içeren toplu iş kesinleştirildiğinde kuyruğa kesinleşmektedir.

Bu nedenle, bir kanal ileti çıkış programından Sync Point MQI çağrılarını yayınlatabilirsiniz.

### *İleti çıkışı dışında ileti dönüştürme*

İleti çıkışını çağırılmadan önce, alıcı MCA ileti üzerinde bazı dönüştürmeler gerçekleştirir. Bu konuda, dönüştürmeleri gerçekleştirmek için kullanılan algoritmalar açıklanmaktadır.

## **Hangi üstbilgilerin işlendiği**

İleti çıkışı çağırılmadan önce, alıcının MCA 'sında bir dönüştürme yordamı çalıştırılır. Dönüştürme yordamı, iletinin başlangıcındaki MQXQH üstbilgisiyle başlar. Dönüştürme yordamı, MQXQH 'yi izleyen zincirleme üstbilgiler yoluyla, gerektiğinde dönüştürme işlemini gerçekleştirmektedir. Zincirleme üstbilgiler, alıcının ileti çıkışa geçirilen MQCXP verilerinin HeaderLength değiştirilmesinde yer alan görece konumun ötesini genişletebilirler. Aşağıdaki üstbilgiler yerinde dönüştürüldü:

- MQXQH (biçim adı " MQXMIT ")
- MQMD (bu üstbilgi MQXQH 'nin bir parçası ve biçim adı yok)
- MQMDE (biçim adı " MQHMDE ")
- MQDH (biçim adı " MQHDIST ")
- MQWIH (biçim adı " MQHWIH ")

Aşağıdaki üstbilgiler dönüştürülmez, ancak MCA zincirleme üstbilgileri işlemeye devam ettikçe devreye girilir:

- MQDLH (biçim adı " MQDEAD ")
- 'MQH' karakteriyle başlayan üstbilgi adları (örneğin, 'MQH' ile başlayan MQHRF ") bu, başka bir şekilde belirtilmeyen

## **Üstbilgilerin nasıl işlendiği**

Her IBM MQ üstbilgisinin Format parametresi MCA tarafından okunur. Biçim parametresi, üstbilgi içindeki 8 byte 'tır (bir ad içeren 8 tek baytlık karakter).

Daha sonra, MCA, her bir üstbilginin adını belirtilen tipte olarak yorumlayarak verileri yorumlar. Biçim, IBM MQ veri dönüştürme için uygun olan bir üstbilgi tipinin adıdır; dönüştürülmektedir. MQ dışı verileri gösteren başka bir ad ise (örneğin, MQFMT\_NONE ya da MQFMT\_STRING gibi), MCA üstbilgileri işlemeyi durdurur.

## **MQCXP HeaderLength nedir?**

İleti çıkışa sağlanan MQCXP verilerindeki HeaderLength değiştirilmesi, iletinin başlangıcındaki MQXQH (MQMD 'yi içerir), MQMDE ve MQDH üstbilgilerini içeren toplam uzunluktur. Bu üstbilgiler 'Biçim' adları ve uzunlukları kullanılarak zincirlenir.

## **MQWIH**

Zincirleme üstbilgiler, kullanıcı verileri alanına HeaderLength alanının ötesine kadar uzayabilir. MQWIH üstbilgisi (varsa), HeaderLength(HeaderLength) altında görüntülenen üstbilgilerden biridir.

Zincirli üstbilgilerde bir MQWIH üstbilgisi varsa, alıcının ileti çıkışı çağırılmadan önce bu üstbilgi yerine dönüştürülür.

### *Kanal iletisi yeniden deneme çıkış programı*

Kanal iletisi-yeniden deneme çıkışı, hedef kuyruğu açma girişimi başarısız olduğunda çağrılır. Hangi koşullar altında yeniden deneneceğini, kaç kez yeniden deneneceğini ve ne sıklıkta yeniden deneneceğini belirlemek için çıkışı kullanabilirsiniz.

Bu çıkış, MCA başlatma ve sonlandırma sırasında kanalın giriş bitişindeki çağrıdır.

Kanal iletisine yeniden deneme çıkışı, iletim kuyruğu üstbilgisini içeren bir aracı arabelleğinden, MQXQH ve kuyruktan alınan uygulama iletisi metninden geçirilir. MQXQH biçimi [Overview for MQXQH](#)(MQXQH için genel bakış) biçiminde verilir.

Çıkış, tüm neden kodları için çağrılır; çıkış, MCA 'nın kaç kez yeniden denemesini, kaç kez ve hangi aralıklarla yeniden denemesini istediğini belirler. (Kanal tanımlandığında, iletinin yeniden deneme sayısı, MQCD 'deki çıkışa geçirilir, ancak çıkış bu değeri yoksayabilir.)

MQXCP 'deki MsgRetrySayı alanı, çıkış çağrıldığında MCA tarafından artırılır ve çıkış, MQXCP ya da MQXCC\_SUPPRESS\_FUNCTION MsgRetryAralığı alanında bekleme süresiyle MQXCC\_OK değerini döndürür. Çıkış, MQXCP 'nin ExitResponse alanında MQXCC\_SUPPRESS\_FUNCTION döndürünceye kadar süresiz olarak devam eder. Bu tamamlanma kodlarına ilişkin MCA tarafından alınan işlemle ilgili bilgi için [MQXCP](#) başlıklı konuya bakın.

Yeniden denemelerin tümü başarısız olursa, ileti, ölü-mektup kuyruğuna yazılır. Kullanılabilir bir ölü harf kuyruğu yoksa, kanal durur.

Bir kanal için ileti yeniden deneme çıkışı tanımlamadıysanız ve geçici olarak büyük olasılıkla geçici (örneğin MQRC\_Q\_FULL) gerçekleştirilmezse, MCA iletiyi kullanır-yeniden deneme sayısı ve ileti-yeniden deneme aralıkları kanal tanımlandığında ayarlanır. Hata daha kalıcı bir niteliyme ve bunu işlemek için bir çıkış programı tanımlamadıysanız, ileti ölü-mektup kuyruğuna yazılır.

#### *Kanal otomatik tanımlama çıkış programı*

Kanal otomatik tanımlama çıkışı, bir alıcı ya da sunucu bağlantısı kanalı başlatmak için bir istek alındığında kullanılabilir, ancak bu kanal için herhangi bir tanım yoktur ( IBM MQ for z/OS için değil). Ayrıca, tüm altyapılarda, bir kanal yönetim ortamı için tanım değişikliğine izin vermek üzere, küme gönderici ve küme alıcı kanallarına ilişkin tüm altyapılarda da çağrılabilir.

Kanal otomatik tanımlama çıkışı, bir alıcı ya da sunucu bağlantısı kanalını başlatmak için bir istek alındığında z/OS dışındaki tüm altyapılarda çağrılabilir, ancak kanal tanımlaması yok. Bunu, otomatik olarak tanımlanmış bir alıcı ya da sunucu bağlantısı kanalı olan SYSTEM.AUTO.RECEIVERya da SYSTEM.AUTO.SVRCON. Kanal tanımlamalarının otomatik olarak nasıl yaratılabileceğiyle ilgili açıklamalar için [Kanalların hazırlanması](#) başlıklı konuya bakın.

Kanal otomatik tanımlama çıkışı, bir küme gönderici kanalı başlatmak için bir istek alındığında da çağrılabilir. Bu kanal, bu kanal yönetim ortamı için tanımlama değişikliğine izin vermek üzere, küme gönderici ve küme alıcı kanalları için çağrılabilir. Bu durumda, çıkış IBM MQ for z/OS için geçerlidir. Çıkış adlarının farklı platformlarda farklı biçimlerde olması nedeniyle, kanal otomatik tanımlama çıkışının ortak bir kullanımı, ileti çıkışlarının (MSGEXIT, RCPEXIT, SCYEXIT ve SENDEXIT) adlarını değiştirmemektedir. If no channel auto-definition exit is specified, the default behavior on z/OS is to examine a distributed exit name of the form *[path]/libraryname(function)* and take up to eight chars of function, if present, or libraryname. On z/OS, a channel auto-definition exit program must alter the fields addressed by MsgExitPtr, MsgUserDataPtr, SendExitPtr, SendUserDataPtr, ReceiveExitPtr, and ReceiveUserDataPtr, rather than the MsgExit, MsgUserData, SendExit, SendUserData, ReceiveExit and ReceiveUserData fields themselves.

Ek bilgi için [Otomatik tanımlı kanallarla çalışmabaşlıklı](#) konuya bakın.

Diğer kanal çıkışlarında olduğu gibi, parametre listesi şöyle olur:

```
MQ_CHANNEL_AUTO_DEF_EXIT (ChannelExitParms, ChannelDefinition)
```

ChannelExitParms , MQXCP içinde açıklanır. ChannelDefinition , MQCD' de açıklanmaktadır.

MQCD, çıkış tarafından değiştirilmezse, varsayılan kanal tanımında kullanılan değerleri içerir. Çıkış, alanların yalnızca bir alt kümesini değiştirebilir; bkz. [MQ\\_CHANNEL\\_AUTO\\_DEF\\_EXIT](#). Ancak, diğer alanları değiştirme girişimi bir hataya neden olmaz.

Kanal otomatik tanımlama çıkışı, MQXCC\_OK ya da MQXCC\_SUPPRESS\_FUNCTION yanıt verir. Bu yanıtlardan hiçbiri döndürülmezse, MCA işlemi, MQXcc\_suppress\_function döndürüldüğü halde işlenmeye devam eder. Yani, otomatik tanımlama iptal edilir, yeni kanal tanımlaması yaratılamaz ve kanal başlayamaz.

#### **Compiling channel exit programs on Windows, UNIX and Linux systems**

Use the following examples to help you compile channel-exit programs for Windows, UNIX and Linux systems.

## Windows

### Windows

The compiler and linker command for channel-exit programs on Windows:

```
cl.exe /Ic:\mqm\tools\c\include /nologo /c myexit.c  
link.exe /nologo /dll myexit.obj /def:myexit.def /out:myexit.dll
```

## UNIX and Linux sistemleri

### Linux

### UNIX

In these examples `çıkış` is the library name and `ChannelExit` is the function name. On AIX the export file is called `exit.exp`. These names are used by the channel definition to reference the exit program using the format described in [MQCD-kanal tanımlaması](#). Ayrıca, [DEFINE CHANNEL](#) komutunun `MSGEXIT` parametresine de bakın.

### AIX

Sample compiler and linker commands for channel exits on AIX:

```
$ xlc_r -q64 -e MQStart -bE:exit.exp -bM:SRE -o /var/mqm/exits64/exit  
exit.c -I/usr/mqm/inc
```

### Linux

Sample compiler and linker commands for channel-exits on Linux where the queue manager is 32 bit:

```
$ gcc -shared -fPIC -o /var/mqm/exits/exit exit.c -I/opt/mqm/inc
```

### Linux

Kuyruk 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
```

### Solaris

Sample compiler and linker commands for channel exits on Solaris:

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/exit exit.c -I/opt/mqm/inc  
-R/usr/lib/64 -lsocket -lnsl -ldl
```

İstemcide, 32 bit ya da 64 bit çıkışı kullanılabilir. Bu çıkış `mqic_r` ile bağlantılandırılmalıdır.

### AIX

AIX' ta, IBM MQ tarafından çağrılan tüm işlevler dışa aktarılmalıdır. Bu make dosyası için örnek bir dışa aktarma dosyası:

```
#  
!channelExit  
MQStart
```

## Kanal çıkışlarının yapılandırılması

Kanal çıkışını aramak için, kanal tanımını kanal tanımında adlanmanız gerekir.

Kanal çıkışlarında kanal çıkışlarının adlandırılması gerekir. Kanalları ilk kez tanımladığınızda bu adlandırma işlemi yapabilir ya da daha sonra, MQSC komutu `ALTER CHANNEL` 'ı kullanarak bilgileri de ekleyebilirsiniz. Ayrıca, ilgili MQCD kanalı veri yapısında kanal çıkış adlarını da verebilirsiniz. Çıkış adının biçimi IBM MQ altyapınıza bağlıdır; bilgi için [MQCD](#) ya da [MQSC komutları](#) konusuna bakın.

Kanal tanımlaması kullanıcı çıkışı programı adı içermiyorsa, kullanıcı çıkışı çağrılmaz.

Kanal otomatik tanımlama çıkışı, tek kanalda değil, kuyruk yöneticisinin özeliidir. Bu çıkışa çağırılması için, kuyruk yöneticisi tanımlamasında adlandırılması gerekir. Bir kuyruk yöneticisi tanımlamasını değiştirmek için, MQSC komutu ALTER QMGR komutunu kullanın.

## Veri dönüştürme çıkışları yazılıyor

Bu konu derlemi, veri dönüştürme çıkışlarının nasıl yazılacağı ile ilgili bilgileri içerir.

**Not:** MQSeries ' da VSE/ESA için desteklenmez.

Bir MQPUT uyguladığınızda, uygulamanız iletinin ileti tanımlayıcısını (MQMD) yaratır. IBM MQ , üzerinde yaratıldığı altyapıdan bağımsız olarak, MQMD ' nin içeriğini anlayabilmesi için sistem tarafından otomatik olarak dönüştürülmelidir.

Ancak, uygulama verileri otomatik olarak dönüştürülmez. Karakter verileri, CodedCharSetId ve Encoding alanlarının farklı olduğu platformlar arasında değiş tokuş ediliyorsa, örneğin, ASCII-EBCDIC arasında, uygulamanın iletiyi dönüştürmesi için bir düzenleme işlemi ayarlamalıdır. Uygulama verileri dönüştürme, kuyruk yöneticisinin kendisi ya da *veri-dönüştürme çıkışı* olarak adlandırılan bir kullanıcı çıkış programı tarafından gerçekleştirilebilir. Uygulama verileri yerleşik biçimlerden biriye (MQFMT\_STRING gibi), kuyruk yöneticisi yerleşik dönüştürme yordamlarından birini kullanarak veri dönüştürmeyi kendisi gerçekleştirebilir. Bu konuda, uygulama verilerinin yerleşik bir biçimde olmadığı durumlarda IBM MQ ' in sağladığı veri dönüştürme çıkış tesisine ilişkin bilgiler yer alır.

Denetim, bir MQGET çağrısı sırasında veri dönüştürme çıkışa geçirebilir. Bu, son hedefe ulaşmadan önce farklı platformlara dönüştürülmeyi önler. Ancak, son hedef, MQGET üzerinde veri dönüştürmeyi desteklemeyen bir altyapıya, verileri son hedefine gönderen gönderici kanalıyla CONVERT (YES) belirtmelisiniz. This ensures that IBM MQ converts the data during transmission. Bu durumda, veri dönüştürme çıkışınızın, gönderen kanalının tanımlandığı sistemde bulunması gerekir.





MQGET çağrısı doğrudan uygulama tarafından verilir. MQMD ' deki CodedCharSetId ve Encoding alanlarını, gerekli karakter takımı ve kodlamaya ayarlayın. Uygulamanız kuyruk yöneticisi olarak aynı karakter kümesini ve kodlamayı kullanıyorsa, CodedCharSetId seçeneğini, MQCCSI\_Q\_MGR ve Encoding olarak MQENC\_NATIVE olarak ayarlayın. MQGET çağrısının tamamlanmasından sonra, bu alanlar döndürülen ileti verilerine uygun değerlere sahiptir. Bu değerler, dönüştürme başarılı olmadıysa, gereken değerlerden farklı olabilir. Uygulamanızın bu alanları, her MQGET çağrısından önce gerekli olan değerlere sıfırlaması gerekir.

Çağrılacak veri dönüştürme çıkışı için gerekli olan koşullar, [MQGET](#)' de MQGET çağrısı için tanımlanmıştır.

Veri dönüştürme çıkışa ve ayrıntılı kullanım notlarına geçirilen parametrelere ilişkin açıklamalar için, MQ\_DATA\_CONV\_EXIT çağrısına ve MQDXP yapısına ilişkin [Veri dönüştürmesi](#) konusuna bakın.

Farklı makine kodlamaları ve CCSID ' ler arasında uygulama verilerini dönüştüren programlar, IBM MQ veri dönüştürme arabirimine (DCI) uygun olmalıdır.

Çok hedefli istemciler için, bazı iletiler kuyruk yöneticisinden geçemeyebileceğinden, API çıkışlar ve veri dönüştürme çıkışlarının istemci tarafında çalışabilmesi gerekir. Aşağıdaki kitaplıklar istemci paketlerinin yanı sıra sunucu paketleri de içerir:

İşletim sistemi	Kitaplıklar
 AIX	32 bit & 64 bit: libmqm.a & libmqm_r.a
 IBM i	LIBMQM & LIBMQM_R
 Linux	32 bit ve 64 bit: libmqm.so & libmqm_r.so
 Solaris	32 bit ve 64 bit: libmqm.so

Çizelge 147. İstemcide ve sunucu paketlerinde bulunan kitaplıklar (devamı var)

İşletim sistemi	Kitaplıklar
Windows	32 bit & 64 bit: mqm.dll & mqm.pdb

### Veri dönüştürme çıkışı çağrılıyor

Veri dönüştürme çıkışı, bir MQGET çağrısının işlenmesi sırasında denetimi alan, kullanıcı tarafından yazılmış bir çıkıştır.

Çıkış, aşağıdaki deyimler doğruysa çağrılır:

- MQGET çağrısında MQGMO\_CONVERT seçeneği belirtildi.
- İleti verilerinin bazıları ya da tümü istenen karakter kümesinde ya da kodlamada değil.
- İletiyile ilişkilendirilen MQMD yapısındaki *Format* alanı MQFMT\_NONE değil.
- MQGET çağrısında belirtilen *BufferLength* sıfır değil.
- İleti verisi uzunluğu sıfır değil.
- İleti, kullanıcı tanımlı bir biçime sahip verileri içerir. Kullanıcı tanımlı biçim, iletinin tamamını kaplayabilir ya da öncesinde bir ya da daha çok yerleşik biçim olabilir. Örneğin, kullanıcı tanımlı biçimden önce bir MQFMT\_DEAD\_LETTER\_HEADER biçiminden önce olabilir. Çıkış, yalnızca kullanıcı tanımlı biçimi dönüştürmek için çağrılır; kuyruk yöneticisi, kullanıcı tanımlı biçimden önce gelen tüm yerleşik biçimleri dönüştürür.

Yerleşik bir biçimi dönüştürmek için kullanıcı tarafından yazılan bir çıkış da çağrılabilir, ancak bu yalnızca yerleşik dönüştürme yordamları yerleşik biçimi başarılı bir şekilde dönüştüremiyorsa gerçekleşir.

MQ\_DATA\_CONV\_EXIT' ta MQ\_DATA\_CONV\_EXIT çağrısının kullanım notlarında tam olarak açıklanan başka koşullar da vardır.

MQGET çağrısına ilişkin ayrıntılar için MQGET konusuna bakın. Veri dönüştürme çıktıları, MQXCNCV dışındaki MQI çağrılarını kullanamaz.

Bir uygulama kuyruk yöneticisine bağlı uygulamadan bu yana *Format* ' u kullanan ilk iletiyi almayı denediğinde çıkışa ilişkin yeni bir kopya yüklenir. Kuyruk yöneticisi önceden yüklenmiş bir kopyayı atmışsa, başka zamanlarda yeni bir kopya da yüklenebilir.

Veri dönüştürme çıkışı, MQGET çağrısını yayınlayan programdan benzer bir ortamda çalışır. Kullanıcı uygulamalarının yanı sıra, program, ileti dönüştürmeyi desteklemeyen bir hedef kuyruk yöneticisine ileti gönderirken bir MCA (ileti kanalı aracısı) olabilir. Ortam, geçerli olduğu yerlerde, adres alanını ve kullanıcı profilini içerir. Çıkış, kuyruk yöneticisinin ortamında çalışmadığından, kuyruk yöneticisinin bütünlüğünü tehlikeye atmaz.

### z/OSüzerinde veri dönüştürme



z/OS' ta aşağıdakilerin farkında olun:

- Çıkış programları yalnızca montaj dilinde yazılabilir.
- Çıkış programları yeniden girişli olmalı ve depolama alanında herhangi bir yeri çalıştırma yeteneğine sahip olmalıdır.
- Çıkış programları, girişteki bu ortamdaki ortamı geri yüklemelidir ve elde edilen herhangi bir depolama alanını serbest kılmalıdır.
- Çıkış programları BEKLEMELI ya da ESTAE ya da SPEs yayınlamamalıdır.
- Çıkış programları genellikle z/OS LINK tarafından aşağıdaki gibi çağrılır:
  - Yetkili olmayan sorun programı durumu
  - Birincil adres alanı denetim kipi
  - Bellek dışı kip

- Erişim olmayan-kayıt kipi
- 31 bit adresleme kipi
- TCB-PRB kipi
- CICS uygulaması tarafından kullanıldığında çıkış EXEC CICS LINK tarafından çağrılır ve CICS programlama kurallarına uymalıdır. Parametreler, CICS iletişim alanındaki işaretçiler (adresler) tarafından aktarılır (COMMAREA).

Önerilmemesine rağmen, kullanıcı çıkış programları CICS API çağrılarını da kullanabilir; aşağıdaki uyarılarda:

- Sonuçlar MCA tarafından bildirilen iş birimlerini etkileyebileceğinden, eşitleme noktaları yayınlamasın.
- CICS Transaction Server tarafından denetlenenler de içinde olmak üzere, IBM MQ for z/OS'da bir kaynak yöneticisi tarafından denetlenen kaynakları güncellemeyin.

CONVERT = YES olan kanallar için, çıkış, CSQXLIB DD bildirimleriyle gönderme yapılan veri kümesinden yüklenir. MQ-sağlanan, IBM MQ CICS Bridge için CSQCBDCI ve CSQCBDCO 'dan çıkarılır ve SCSQAUTH' da yer alıyor.

### **IBM için veri dönüştürme çıkış programı yazılıyor**

IBM için MQ veri dönüştürme çıkış programları yazıldığında dikkate alınacak adımlar hakkında bilgi.

Aşağıdaki adımları izleyin:

1. İleti biçiminizi adlayın. Adın, MQMD 'nin *Format* alanına sığması gerekir. *Format* adının başında gömülü boşluklar bulunmalı ve sondaki boşluklar yok sayılır. *Format* yalnızca sekiz karakter uzunluğunda olduğu için, nesnenin adı en çok sekiz boşluk karakteri olmayan karakterlere sahip olmalıdır. Bir iletiyi her gönderdiğinizde bu adı kullanmayı unutmayın (örneğin, ad biçimini kullanır).
2. İletinizi göstermek için bir yapı oluşturun. Bir örnek için [Geçerli sözdizimi](#) konusuna bakın.
3. Veri dönüştürme çıkışınız için bir kod parçası yaratmak üzere CVTMQMDTA komutu aracılığıyla bu yapıyı çalıştırın.

CVTMQMDTA komutu tarafından üretilen işlevler QMQM/H (AMQSVMHA) dosyasında verilen makroları kullanır. Bu makrolar, tüm yapıların paketlenmiş olduğu varsayılarak yazılır; dosya bu değilse, bunların değiştirileceği şekilde değiştirilecektir.

4. Sağlanan çatı kaynak dosyasının, QMQMSAMP/QCSRC (AMQSVFC4) bir kopyasını alın ve yeniden adlandırın. (örneğin EXIT\_MOD adını kullanır.)
5. Kaynak dosyada aşağıdaki açıklama kutularını bulun ve anlatıldığı gibi kodu ekleyin:

- a. Kaynak dosyanın sonuna doğru, bir yorum kutusu şununla başlar:

```
/* Insert the functions produced by the data-conversion exit */
```

Burada, "[3](#)" sayfa 1016adımında oluşturulan kod parçasını ekleyin.

- b. Kaynak dosyanın ortasındaki bir yorum kutusu şununla başlar:

```
/* Insert calls to the code fragments to convert the format's */
```

Bunu, ConverttagSTRUCT işlevine ilişkin açıklama satırı yapılan bir çağrıyı izlemektedir.

İşleve ilişkin adı, "[5.a](#)" sayfa 1016adımında eklediğiniz işlevin adına çevirin. İşlevi etkinleştirmek için açıklama karakterlerini kaldırın. Birden çok işlev varsa, bunların her biri için çağrı yaratın.

- c. Kaynak dosyanın başlangıcındaki bir açıklama kutusu şununla başlar:

```
/* Insert the function prototypes for the functions produced by */
```

Here, insert the function prototype statements for the functions added in step "[5.a](#)" sayfa 1016.



İleti karakter verisi içeriyorsa, üretilen kod MQXCNCV ' yi çağırır; bu durum, QMQM/LIBMQM hizmet programı bağ tanımlanarak çözülebilir.

6. EXIT\_MOD adlı kaynak modülünü aşağıdaki gibi derleyin:

```
CRTCMOD MODULE(library/EXIT_MOD) +  
SRCFILE(QCSRC) +  
TERASPACE(*YES *TSIFC)
```

7. Program yarat/bağla.

İş parçacıklı olmayan uygulamalar için aşağıdaki bilgileri kullanın:

```
CRTPGM PGM(library/Format) +  
MODULE(library/EXIT_MOD) +  
BNDSRVPGM(QMQM/LIBMQM) +  
ACTGRP(QMQM) +  
USRPRF(*USER)
```

Temel ortam için veri dönüştürme çıkışının yaratılmasına ek olarak, iş parçacıklı ortamda başka bir veri dönüştürme işlemi gereklidir. Bu yüklenebilir nesne \_R ile izlenmelidir. MQXCNCV çağrılarını çözmek için LIBMQM\_R kitaplığını kullanın. İş parçacıklı bir ortam için yüklenebilir nesnelerin her ikisi de gereklidir.

```
CRTPGM PGM(library/Format_R) +  
MODULE(library/EXIT_MOD) +  
BNDSRVPGM(QMQM/LIBMQM_R) +  
ACTGRP(QMQM) +  
USRPRF(*USER)
```

8. Çıkışı, IBM MQ işi için kitaplık listesine yerleştirin. Üretim için, veri dönüştürme çıkış programlarının QSYS ' de saklanacağı önerilir.

#### Not:

1. CVTMQMDTA paketlenmiş yapıları kullanırsa, tüm IBM MQ uygulamaları \_Packed niteleyicisini kullanmalıdır.
2. Veri dönüştürme çıkış programları yeniden girişli olmalıdır.
3. MQXCNCV, bir veri dönüştürme çıkışından verilebilecek tek MQI çağırısı.
4. Çıkış programını, çıkış programının kullanıcı yetkisiyle çalışabilmesi için, kullanıcı tanıtımı derleyicisi seçeneği olarak \*USER değerine ayarlayın.
5. IBM MQ for IBM i ile tüm kullanıcı çıkışları için Teraspace bellek etkinleştirilmesi gereklidir; CRTCMOD ve CRTBNDC komutlarında TERASPACE (\*YES \*TSIFC) değerini belirleyin.

### **IBM MQ for z/OS için veri dönüştürme çıkış programı yazılıyor**

IBM MQ for z/OS için veri dönüştürme çıkış programları yazılırken dikkate alınacak adımlar hakkında bilgi.

Aşağıdaki adımları izleyin:

1. Take the supplied source skeleton CSQ4BAX9 (for non-CICS environments) or CSQ4CAX9 (for CICS ) as your starting point.
2. CSQUCVX yardımcı programını çalıştırın.
3. CSQUCVX yardımcı programı tarafından oluşturulan yordamları, yapılarının dönüştürmek istediğiniz iletilerde olduğu sırada birleştirmek için CSQ4BAX9 ya da CSQ4CAX9 ' ın öngünlük içindeki yönergeleri izleyin.
4. Yardımcı program veri yapılarının paketlenmediğini, verilerin örtük olarak hizalanmasını ve yapıların bir tam sözcük sınırı üzerinde başlatıldığını ve byte 'lar gerektiği şekilde atlanmasını (örneğin, Geçerli sözdizimi ' daki örnekte olduğu gibi) atlanan byte 'lar olduğunu varsayar. Yapılar paketleniyse, oluşturulan CMQXCALA makrolarını atlayın. Therefore, consider declaring your structures in such a way that all fields are named and no bytes are skipped; in the example in Geçerli sözdizimi, add a field "MQBYTE KUKLA;" between ID and VERSION.

5. Giriş arabelleği dönüştürülecek ileti biçiminden kısaysa, sağlanan çıkış bir hata döndürür. Çıkış, mümkün olduğunca çok sayıda eksiksiz alan döndürse de, hata dönüştürülmeyen bir iletinin uygulamaya döndürülmesine neden olur. Kısmi alanlar da içinde olmak üzere, kısa giriş arabelleklerinin olabildiğince uzağa dönüştürülmesine izin vermek istiyorsanız, CSQXCDF A makrosu üzerindeki TRUNC= değerini YES olarak değiştirin: Bir hata döndürülmez, bu nedenle uygulama dönüştürülmüş bir ileti alır. Uygulamanın kesilmeyi işlemesi gerekir.
6. Gereksinim duyarsanız, diğer özel işlem kodunu ekleyin.
7. Programı veri biçimi adınıza yeniden adlandırın.
8. Programınızı bir toplu iş uygulama programı gibi derleyin ve bağlayın ( CICS uygulamalarıyla birlikte kullanılmadığı sürece). Yardımcı program tarafından oluşturulan koddaki makrolar, **thlqual.SCSQMACS** kitaplığında yer alıyor.  
İleti karakter verisi içeriyorsa, üretilen kod MQXCNCV ' yi çağırır. Çıkışınız bu çağırışı kullanıyorsa, cSQASKICE çıkış kod parçası programını kullanarak bağlantıyı düzenleyin. Sınırlı kod öbeği, dilden bağımsız ve ortamdaki bağımsızdır. Diğer bir seçenek olarak, CSQXCNCV dinamik çağrı adını kullanarak sınırlı kod öbeğini dinamik olarak yükleyebilirsiniz. Ek bilgi için [“IBM MQ sınırlı kod öbeğini devingen olarak çağırma” sayfa 1065](#) başlıklı konuya bakın.  
Bağlantı düzenlenmiş modülü, uygulama yükleme kitaplığınıza ve kanal başlatıcınızın başlattığı görev yordamınızın CSQXLIB DD bildiriminde gönderme yapılan bir veri kümesine yerleştirin.
9. Çıkış, CICS uygulamaları tarafından kullanılırsa, derleme ve bağlantı varsa, gerekiyorsa CSQASOR da içinde olmak üzere, bir CICS uygulama programı gibi düzenleyin. Bunu CICS uygulama programı kitaplığınıza yerleştirin. EXECKEY ( CICS ) belirtilerek, programı tipik şekilde CICS olarak tanımlayın. tanımını kullanın.

**Not:** Although the LE/370 runtime libraries are needed for running the CSQUCVX utility (see step “2” [sayfa 1017](#) ), they are not needed for link-editing or running the data-conversion exit itself (see steps “8” [sayfa 1018](#) and “9” [sayfa 1018](#) ).

IBM MQ - IMS köprüsü içindeki veri dönüştürmeye ilişkin bilgi için bkz. [“IMS köprü uygulamaları yazılıyor” sayfa 70](#) .

## Linux > UNIX **UNIX and Linux sistemlerinde IBM MQ için veri dönüştürme çıkışı yazılıyor**

UNIX and Linux sistemlerinde IBM MQ için veri dönüştürme çıkış programları yazılırken dikkate alınacak adımlar hakkında bilgi.

Aşağıdaki adımları izleyin:

1. İleti biçiminizi adlayın. Ad, MQMD ' nin *Format* alanına sığmalı ve büyük harfli olmalıdır; örneğin, MYFORMAT. *Format* adının başında boşluk bulunmamalıdır. Sondaki boşluklar yok sayılır. *Format* yalnızca sekiz karakter uzunluğunda olduğu için, nesnenin adı en çok sekiz boşluk karakteri olmayan karakterlere sahip olmalıdır. Bir iletiyi her gönderdiğinizde bu adı kullanmayı unutmayın.  
Veri dönüştürme çıkışı bir iş parçacıklı bir ortamda kullanılırsa, yüklenebilir nesnenin yivli bir sürüm olduğunu belirtmek için \_r tarafından izlenmelidir.
2. İletinizi göstermek için bir yapı oluşturun. Bir örnek için [Geçerli sözdizimi](#) konusuna bakın.
3. Veri dönüştürme çıkışınız için bir kod parçası oluşturmak üzere bu yapıyı c1tmqcvx komutu aracılığıyla çalıştırın.  
c1tmqcvx komutu tarafından oluşturulan işlevler, tüm yapıların paketlenmiş olduğunu varsayan makroları kullanır; bu durumda, bu makroların hata durumunda olması gerekir.
4. Belirtilen çatı kaynağı dosyasını kopyalayın, bu dosyayı “1” [sayfa 1018](#) adımıyla ayarladığınız ileti biçiminizin adıyla yeniden adlandırın. İskelet kaynak dosyası ve kopyası salt okunurdur.  
The skeleton source file is called amqsvfc0.c.
5. IBM MQ for AIX üzerinde, amqsvfc.exp adlı bir çatı dışı aktarma dosyası da sağlanır. Bu dosyayı kopyalayın ve MYFORMAT.EXP olarak yeniden adlandırın.

6. İskelet, `MQ_INSTALLATION_PATH/inc` dizininde örnek bir üstbilgi dosyası ( `amqsvmha.h`) içerir; burada `MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder. Bu dosyayı almak için, içirme yolunuzun bu dizini gösterdiğinden emin olun.

`amqsvmha.h` dosyası, `crtmqcvx` komutu tarafından oluşturulan kod tarafından kullanılan makroları içerir. Dönüştürülecek yapı karakter verisi içeriyorsa, bu makrolar `MQXCNCV` ' yi çağırır.

7. Kaynak dosyada aşağıdaki açıklama kutularını bulun ve anlatıldığı gibi kodu ekleyin:
- Kaynak dosyanın sonuna doğru, bir yorum kutusu şununla başlar:

```
/* Insert the functions produced by the data-conversion exit */
```

Burada, “3” sayfa 1018adımında oluşturulan kod parçasını ekleyin.

- Kaynak dosyanın ortasındaki bir yorum kutusu şununla başlar:

```
/* Insert calls to the code fragments to convert the format's */
```

Bunu, `ConverttagSTRUCT` işlevine ilişkin açıklama satırı yapılan bir çağrıyı izlemektedir.

İşleve ilişkin adı, “7.a” sayfa 1019adımında eklediğiniz işlevin adına çevirin. İşlevi etkinleştirmek için açıklama karakterlerini kaldırın. Birden çok işlev varsa, bunların her biri için çağrı yaratın.

- Kaynak dosyanın başlangıcındaki bir açıklama kutusu şununla başlar:

```
/* Insert the function prototypes for the functions produced by */
```

Here, insert the function prototype statements for the functions added in step “3” sayfa 1018.

- Giriş noktası olarak `MQStart` ' ı kullanarak, çıkışınızı paylaşılan kitaplık olarak derleyin. Bunu yapmak için bkz. “UNIX and Linux sistemlerindeki veri dönüştürme çıkışlarının derlenmesi” sayfa 1019.
- Çıkışın çıkış dizinine yerlesin. Varsayılan çıkış dizini, 32 bit sistemler için `/var/mqm/exits` ve 64 bit sistemler için `/var/mqm/exits64` ' dir. Bu dizinleri `qm.ini` ya da `mqclient.ini` dosyasında değiştirebilirsiniz. Bu yol, her bir kuyruk yöneticisi için ayarlanabilir ve çıkış yalnızca o yol ya da yollardaki aralara ararlıdır.

#### Not:

- `crtmqcvx` paketlenmiş yapıları kullanıyorsa, tüm IBM MQ uygulamaları bu şekilde derlenmelidir.
- Veri dönüştürme çıkış programları yeniden girişli olmalıdır.
- `MQXCNCV`, bir veri dönüştürme çıkışından verilebilecek tek MQI çağrısı.

**Linux** **UNIX** *UNIX and Linux sistemlerindeki veri dönüştürme çıkışlarının derlenmesi*  
UNIX and Linux sistemlerinde veri dönüştürme çıkışının nasıl derleneceğini gösteren örnekler.

Tüm altyapılarda, modüle giriş noktası `MQStart` ' tır.

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

## 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
```

## Solaris

### Solaris

Aşağıdaki komutlardan birini vererek çıkış kodunu derleyin:

### 32 bit uygulamalar

#### SPARC altyapısı

```
cc -xarch=v8plus -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \  
MYFORMAT.c -I MQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

#### x86-64 platformu

```
cc -xarch=386 -KPIC -mt -G -o /var/mqm/exits/MYFORMAT \  
MYFORMAT.c -I MQ_INSTALLATION_PATH/inc -R/usr/lib/32 -lsocket -lnsl -ldl
```

### 64 bit uygulamalar

#### SPARC altyapısı

```
cc -xarch=v9 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \  
MYFORMAT.c -I MQ_INSTALLATION_PATH/inc -R/usr/lib/64 -lsocket -lnsl -ldl
```

#### x86-64 platformu

```
cc -xarch=amd64 -KPIC -mt -G -o /var/mqm/exits64/MYFORMAT \  
MYFORMAT.c -I MQ_INSTALLATION_PATH/inc -R/usr/lib/64 -lsocket -lnsl -ldl
```

## Windows

### IBM MQ for Windows için veri dönüştürme çıkışı yazılıyor

IBM MQ for Windows için veri dönüştürme çıkış programları yazılırken dikkate alınacak adımlar hakkında bilgi.

Aşağıdaki adımları izleyin:

1. İleti biçiminizi adlayın. Adın, MQMD ' nin *Format* alanına sığması gerekir. *Format* adının başında boşluk bulunmamalıdır. Sondaki boşluklar yok sayılır. *Format* yalnızca sekiz karakter uzunluğunda olduğu için, nesnenin adı en çok sekiz boşluk karakteri olmayan karakterlere sahip olmalıdır.

A .DEF file called amqsvfcn.def is also supplied in the samples directory, *MQ\_INSTALLATION\_PATH\Tools\C\Samples.MQ\_INSTALLATION\_PATH*, IBM MQ ' in kurulu olduğu dizindir. Bu dosyanın bir kopyasını alın ve yeniden adlandırın; örneğin, MYFORMAT.DEF' a. Yaratılmakta olan DLL adının ve MYFORMAT.DEF değeri aynıdır. Overwrite the name FORMAT1 in MYFORMAT.DEF with the new format name.

Bir iletiyi her gönderdiğinizde bu adı kullanmayı unutmayın.

2. İletinizi göstermek için bir yapı oluşturun. Bir örnek için [Geçerli sözdizimi](#) konusuna bakın.
3. Veri dönüştürme çıkışınız için bir kod parçası oluşturmak üzere bu yapıyı `crtmqcvx` komutu aracılığıyla çalıştırın.

CRTMQCVX komutu tarafından üretilen işlevler, tüm yapıların paketlenmiş olduğu varsayılarak yazılmış makroları kullanır; bu durumda, bu durumda durum bu değilse, bunları sona erdirir.

4. Belirtilen çatı kaynağı dosyasını ( amqsvfc0.c), “1” sayfa 1021adımında ayarladığınız ileti biçiminizin adına yeniden adlandırmanızı sağlar.

amqsvfc0.c , *MQ\_INSTALLATION\_PATH*\Tools\C\Samples dizininde, burada *MQ\_INSTALLATION\_PATH* , IBM MQ ' nin kurulu olduğu dizindir. (Varsayılan kuruluş dizini: C:\Program Files\IBM\MQ.)

The skeleton includes a sample header file amqsvmha.h in the *MQ\_INSTALLATION\_PATH*\Tools\C\include directory. Bu dosyayı almak için, içerme yolunuzun bu dizini gösterdiğinden emin olun.

amqsvmha.h dosyası, CRTMQCVX komutu tarafından oluşturulan kod tarafından kullanılan makroları içerir. Dönüştürülecek yapı karakter verisi içeriyorsa, bu makrolar MQXCNCV ' yi çağırır.

5. Kaynak dosyada aşağıdaki açıklama kutularını bulun ve anlatıldığı gibi kodu ekleyin:
- a. Kaynak dosyanın sonuna doğru, bir yorum kutusu şununla başlar:

```
/* Insert the functions produced by the data-conversion exit */
```

Burada, “3” sayfa 1021adımında oluşturulan kod parçasını ekleyin.

- b. Kaynak dosyanın ortasındaki bir yorum kutusu şununla başlar:

```
/* Insert calls to the code fragments to convert the format's */
```

Bunu, ConverttagSTRUCTişlevine ilişkin açıklama satırı yapılan bir çağrıyı izlemektedir.

İşleve ilişkin adı, “5.a” sayfa 1022adımında eklediğiniz işlevin adına çevirin. İşlevi etkinleştirmek için açıklama karakterlerini kaldırın. Birden çok işlev varsa, bunların her biri için çağrı yaratın.

- c. Kaynak dosyanın başlangıcındaki bir açıklama kutusu şununla başlar:

```
/* Insert the function prototypes for the functions produced by */
```

Here, insert the function prototype statements for the functions added in step “3” sayfa 1021.

6. Aşağıdaki komut dosyasını oluşturun:

```
c1 -I MQ_INSTALLATION_PATH\Tools\C\Include -Tp \
MYFORMAT.C
```

```
MYFORMAT.DEF
```

Burada *MQ\_INSTALLATION\_PATH* , IBM MQ ' in kurulu olduğu dizindir.

7. Çıktınızı bir DLL dosyası olarak derlemek için komut kütüğünü verin.
8. Place the output in the exit subdirectory below the IBM MQ data directory. Çıktılarınızı 32 bit sistemlerle kurmak için kullanılan varsayılan dizin *MQ\_DATA\_PATH*\Exits ve 64 bit sistemler için *MQ\_DATA\_PATH*\Exits64dizindir.

Veri dönüştürme çıktıları aramak için kullanılan yol kayıta verilmiştir. Kayıt dosyası klasörü:

```
HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere
MQ\Installation\MQ_INSTALLATION_NAME\Configuration\ClientExitPath\
```

ve kayıt defteri anahtarı: ExitsDefaultPath. Bu yol, her bir kuyruk yöneticisi için ayarlanabilir ve çıkış yalnızca o yol ya da yollardaki aralara ararlıdır.

#### Not:

1. CRTMQCVX paketlenmiş yapıları kullanıyorsa, tüm IBM MQ uygulamaları bu şekilde derlenmelidir.
2. Veri dönüştürme çıkış programları yeniden girişli olmalıdır.

3. MQXCNCV, bir veri dönüştürme çıkışından verilebilecek tek MQI çağrısı.

## Windows **Windows işletim sistemlerinde yükleme dosyalarını açma ve değiştirme**

IBM WebSphere MQ for Windows 7.5 kuyruk yöneticisi işlemleri 32 bit 'tür. Sonuç olarak, 64 bit kullanan uygulamalar kullanılırken, bazı çıkış tipleri ve XA anahtar yükleme dosyalarının kuyruk yöneticisi tarafından kullanılmak üzere 32 bit sürümünün kullanılabilir olması gerekir. Çıkış ya da XA anahtarı yükleme dosyasının 32 bit sürümü gerekiyorsa ve kullanılamıyorsa, ilgili API çağrısı ya da komutu başarısız olur.

Two attributes are supported in the `qm.ini` file for `ExitPath`. Bunlar `ExitsDefaultPath=MQ_INSTALLATION_PATH\exits` ve `ExitsDefaultPath64=MQ_INSTALLATION_PATH\exits64` dir. `MQ_INSTALLATION_PATH`, IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder. Bunlar, uygun kitaplığın bulunabilmesini sağlar. IBM MQ kümesinde bir çıkış kullanılıyorsa, bu işlem uzak bir sistemdeki uygun kitaplığın bulunabilmesini sağlar.

Aşağıdaki çizelgede, 32 bit ya da 64 bit kullanan uygulamaların kullanılmakta olup olmadığına göre, farklı çıkış tipleri ve anahtar yükleme dosyaları ve 32 bit ya da 64 bit sürümlerinin ya da her ikisinin de gerekli olup olmadığını listelemektedir:

Dosya türleri	32 bit 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 İşlem Yöneticisi AX kitaplıkları	32 bit	64 bit
Bağlantı öncesi çıkış	32 bit	64 bit

## **Bir havuzdaki bağlanma öncesi çıkışı kullanarak bağlantı tanımlarına gönderme yapılması**

IBM MQ MQI clients , bir bağlantı öncesi çıkış kitaplığı kullanarak bağlantı tanımları elde etmek üzere bir havuzu aramak üzere yapılandırılabilir.

### **Giriş**

İstemci uygulaması, istemci kanal tanımlama çizelgelerini (CCDT) kullanarak bir kuyruk yöneticisine bağlanabiliyor. Genel olarak, CCDT dosyası merkezi bir ağ dosya sunucusunda bulunur ve bu dosyaya gönderme yapan istemcilere sahiptir. CCDT dosyasına gönderme yapan çeşitli istemci uygulamalarının yönetilmesi ve yönetilmesi zor olduğundan, esnek bir yaklaşım, istemci tanımlamalarının bir LDAP dizini, bir WebSphere Registry and Repository ya da başka bir havuz gibi genel bir havuzda saklamasıdır. Bir havuzda istemci bağlantısı tanımlarının saklanması, istemci bağlantısı tanımlarının yönetilmesini kolaylaştırır ve uygulamalar, doğru ve en güncel istemci bağlantısı tanımlamalarına erişebilir.

MQCONN/X çağrısı yürütülürken, IBM MQ MQI client , bir uygulamanın önceden bağlanma öncesi çıkış kitaplığını yükler ve bağlantı tanımlamalarını almak için bir çıkış işlevini çağırır. Daha sonra, alınan

bağlantı tanımları bir kuyruk yöneticisiyle bağlantı kurmak için kullanılır. Çağrılacak çıkış kitaplığı ve işlevinin ayrıntıları mqclient.ini yapılandırma dosyasında belirtilir.

## Sözdizimi

```
void MQ_PRECONNECT_EXIT (pExitParms, pQMGrAd, ppConnectOpts, pCompKodu, pReason);
```

## Parametreler

### pExitParms

Tip: PMQNX giriş /çıkış

**PreConnection** çıkış değıştirgesi yapısı.

Yapı, çıkışa ilişkin çağırıcı tarafından ayrılır ve sürdürür.

### pQMGrAdı

Tip: PMQCHAR giriş/çıkış

Kuyruk yöneticisinin adı.

On input, this parameter is the filter string supplied to the MQCONN API call through the **QMGrName** parameter. Bu alan boş, açık ya da belirli genel arama karakterleri içerebilir. Alan, çıkışa göre değıştirilir. Çıkış MQXR\_TERM ile çağrıldığında, parametre NULL (boş değeri) olur.

### ppConnectSeçenekleri

Tip: ppConnectGiriş/çıkış

MQCONNX 'in işlemini denetleyen seçenekler.

Bu, MQCONN API çağırısının işlemini denetleyen MQCNO bağlantı seçenekleri yapısına bir işaretir. Çıkış MQXR\_TERM ile çağrıldığında, parametre NULL (boş değeri) olur. MQI istemcisi, çıkışa her zaman, uygulama tarafından sağlanmamış olsa da, çıkışa bir MQCNO yapısı sağlar. Bir uygulama MQCNO yapısı sağlıyorsa, istemci bunu değıştirdiği yerden çıkışa geçirmek için bir yineleme yapar. İstemci MQCNO ' nun sahipliğini korur.

MQCNO ile gönderme yapılan bir MQCD, dizi aracılığıyla sağlanan bağlantı tanımlarına göre öncelik kazanır. İstemci, kuyruk yöneticisine bağlanmak için MQCNO yapısını kullanıyor ve diğeri yok sayılıyor.

### pCompKodu

Tip: PMQXX\_ENCODE\_CASE\_ONE long giriş/çıkış

Tamamlanma kodu.

Çıkışların tamamlanma kodunu alan bir MQUZE işaretçisi. Bu değeri aşağıdaki değerlerden biri olmalıdır:

- MQCC\_OK -Tamamlama tamamlandı
- MQCC\_UYARI -Uyarı (kısmi tamamlama)
- MQCC\_FAILED -Arama başarısız oldu

### pReason

Tip: PMQXX\_ENCODE\_CASE\_ONE long giriş/çıkış

Neden niteleyici pCompkodu.

Çıkış neden kodunu alan bir MQUZE işaretçisi. Tamamlanma kodu MQCC\_OK ise, tek geçerli değeri şöyledir:

- MQRC\_NONE-(0, x '000') Raporlamak için bir neden yok.

Tamamlanma kodu MQCC\_FAILED ya da MQCC\_UYARI ise, çıkış işlevi neden kodu alanını geçerli bir MQRC\_ \* değerine ayarlayabiliyor.



## C Çağırma

```
void MQ_PRECONNECT_EXIT (&ExitParms, &QMgrName, &pConnectOpts, &CompCode, &Reason);
```

### Parameter

```
PMQNX pExitParms /*PreConnect exit parameter structure*/  
PMQCHAR pQMgrName /*Name of the queue manager*/  
PPMQCNO ppConnectOpts/*Options controlling the action of MQCONN*/  
PMQLONG pCompCode /*Completion code*/  
PMQLONG pReason /*Reason qualifying pCompCode*/
```

## Yayınlama çıkışlarının yazılması ve derlenmesi

Yayınlanan bir iletinin içeriğini aboneler tarafından alınmadan önce değiştirmek için, kuyruk yöneticisinde bir yayınlama çıkışı yapılandırabilirsiniz. Ayrıca, ileti üstbilgisini değiştirebilir ya da iletiyi bir aboneliğe teslim etmemeniz de olabilir.

**Not:** Publish exits are not supported on z/OS.

Abonelere teslim edilen iletileri incelemek ve değiştirmek için yayınlama çıkışını kullanabilirsiniz:

- Her abonede yayınlanan bir iletinin içeriğini inceler
- Her abonede yayınlanan bir iletinin içeriğini değiştirme
- İletinin konacağı kuyruğu değiştir
- Bir iletiyi aboneye teslim etmeyi durdurur

## Yayınlama çıkışı yazılıyor

Çıkışınızı yazmanıza ve derlemenize yardımcı olması için [“Writing exits and installable services on UNIX, Linux and Windows”](#) sayfa 968’indeki adımları kullanın.

Yayınlama çıkışa ilişkin sağlayıcı, çıkışa ilişkin bilgileri tanımlar. Ancak çıkış, [MQPSXP](#)’de tanımlanan kurallara uygun olmalıdır.

IBM MQ , MQ\_PUBLISH\_EXIT giriş noktasının bir somutlamasını sağlamıyor. Bu, bir C dili tipidef bildirimini sağlar. Değiştirgelerin kullanıcı tarafından yazılmış bir çıkışa doğru olarak bildirilmesi için typedef 'i kullanın. Aşağıdaki örnekte, typedef bildiriminin nasıl kullanılacağı gösterilmektedir:

```
#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 */  
}
```

Aşağıdaki işlemlerin bir sonucu olarak, yayınlama çıkışı kuyruk yöneticisi işlemi içinde çalışır:

- Bir iletinin bir ya da daha fazla aboneye teslim edildiği bir yayınlama işlemi
- Bir ya da daha fazla tutulan iletinin teslim edildiği bir abone olma işlemi
- Bir ya da daha fazla tutulan iletinin teslim edildiği bir Abonelik İsteği işlemi

Yayınlama çıkışı bir bağlantı için çağrılırsa, ilk olarak *ExitReason* kodu MQXR\_INIT olarak adlandırılır. Before the connection disconnects after using a publish exit, the exit is called with an *ExitReason* code of MQXR\_TERM.

Yayınlama çıkışı yapılandırıldıysa, ancak kuyruk yöneticisi başlatıldığında, ancak kuyruk yöneticisi için yayınlama/abone olma ileti işlemleri engellendiğinde yüklenemez. Yayınlama/abone olma ileti sistemi

yeniden etkinleştirilmeden önce sorunu düzeltmeniz ya da kuyruk yöneticisini yeniden başlatmanız gerekir.

Yayınlama çıkışı gerektiren her IBM MQ bağlantısı, çıkışı yükleyemeyebilir ya da kullanıma hazırlamayabilir. Çıkış yükleme ya da kullanıma hazırlama işlemi başarısız olursa, yayınlama çıkışı gerektiren abone olma/abone olma işlemleri ilgili bağlantı için geçersiz kılır. İşlemler IBM MQ neden kodu MQRC\_PUBLISH\_EXIT\_ERROR ile başarısız olur.

Yayınlama çıkışının çağrıldığı bağlam, bir uygulama tarafından kuyruk yöneticisine yönelik bağlantıdır. Bir kullanıcı veri alanı, yayınlama işlemleri gerçekleştiren her bir bağlantı için kuyruk yöneticisi tarafından korunur. Çıkış, her bağlantı için kullanıcı verileri alanındaki bilgileri saklayabilir.

Yayınlama çıkışı bazı MQI çağrılarını kullanabilir. Bu, yalnızca ileti özelliklerini yönlendiren bu MQI çağrılarını kullanabilir. Aramalar:

- MQBUFMH
- MQCRTMH
- MQDLTMH
- MQDLTMP
- MQMHBUF
- MQINQMP
- MQSETMP

Yayınlama çıkışı hedef kuyruk yöneticisini ya da kuyruk adını değiştirirse, yeni bir yetki denetimi gerçekleştirilmez.

## Yayınlama çıkışının derlenmesi

Yayınlama çıkışı dinamik olarak yüklenmiş bir kitaptır; kanal çıkışı olarak düşünülebilmektedir. Çıkışların derlenmesiyle ilgili bilgi için bkz. [“Writing exits and installable services on UNIX, Linux and Windows”](#) sayfa 968.

## Örnek yayınlama çıkışı

Örnek çıkış programı amqspse0 . olarak adlandırılır. Çıkış, başlatma, yayınlama ya da sonlandırma işlemleri için çıkışa çağrılıp çağrılmadığına bağlı olarak, günlük dosyasına farklı bir ileti yazar. Ayrıca, çıkış kullanıcı alanı alanının, depolamayı uygun bir şekilde ayırmak ve serbest bir şekilde serbest yapmak için kullanılması da gösterilir.

## Yayınlama çıkışlarının yapılandırılması

Yayınlama çıkışı yapılandırmak için bazı öznitelikleri tanımlamanız gerekir.

Windows ve Linux ' ta, öznitelikleri tanımlamak için IBM MQ gezginini kullanabilirsiniz. Öznitelikler, Publish/Subscribe altında, kuyruk yöneticisi özellikleri sayfasında tanımlanır.


Yayınlama çıkışını UNIX and Linux sistemlerinde qm . ini dosyasında yapılandırmak için PublishSubscribe adlı bir stanza oluşturun. PublishSubscribe Stanza aşağıdaki özniteliklere sahiptir:

### **PublishExitYol = [ yol] |modüle\_adi**

Yayınlama çıkış kodunu içeren modül adı ve yolu. Bu alanın uzunluk üst sınırı MQ\_EXIT\_NAME\_LENGTH' dir. Varsayılan değer yayınlama çıkışıdır.

### **PublishExitİşlevi = function\_name**

İşlev girdisi noktasının adı, yayınlama çıkış kodunu içeren modüle işaret eder. Bu alanın uzunluk üst sınırı MQ\_EXIT\_NAME\_LENGTH' dir.

 On IBM i, if a program is used, omit PublishExitFunction.

### **PublishExitVeri = dizgi**

Kuyruk yöneticisi bir yayınlama çıkışı çağırıyorsa, giriş olarak bir MQPSXP yapısını geçirir. The data specified using the **PublishExitData** attribute is provided in the *ExitData* field of the structure. Dize, MQ\_EXIT\_DATA\_LENGTH karakterlerine kadar uzunluğa kadar çıkabilmektedir. 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. İletilerin yönlendirilmesi sırasında, günün farklı zamanlarında bir kanalı ya da ileti içeriğini kullanarak bir kanalı kullanma maliyetini de kullanabilirsiniz. Bunlar, standart iş yükü yönetimi algoritmasıyla dikkate alınmayan etkenlerdir.

Çoğu durumda, iş yükü yönetimi algoritması gereksinimleriniz için yeterli olur. Ancak, iş yükü yönetimini uyarlamak için kendi kullanıcı çıkışı programınızı sağlayabilmeniz için, IBM MQ bir kullanıcı çıkışı, küme iş yükü çıkışı içerir.

Ağınızla ya da iş yükü dengelemesini etkilemek için kullanabileceğiniz iletilerinizle ilgili bazı bilgilere sahip olabilirsiniz. Yüksek kapasiteli kanallar ya da ucuz ağ rotaları hangileridir, ya da iletileri içeriğine bağlı olarak yönlendirmek isteyebilirsiniz. Bir küme iş yükü çıkış programı yazmaya karar verebilir ya da bir üçüncü kişi tarafından sağlanan bir program kullanabilirsiniz.

Küme iş yükü çıkışı bir küme kuyruğuna erişilirken çağrılır. It is called by MQOPEN, MQPUT1 and MQPUT.

The target queue manager selected at MQOPEN time is fixed if MQ00\_BIND\_ON\_OPEN is specified. Bu durumda, çıkış yalnızca bir kez çalıştırılır.

If the target queue manager is not fixed at MQOPEN time, the target queue manager is chosen at the time of the MQPUT call. Hedef kuyruk yöneticisi kullanılabilir değilse ya da ileti hala iletim kuyruğunda olduğunda başarısız olursa, çıkış yeniden çağrılır. Yeni bir hedef kuyruk yöneticisi seçildi. İleti aktarılırken ileti kanalı başarısız olursa ve ileti geriletirse, yeni bir hedef kuyruk yöneticisi seçilir.

**Multi** Multiplatformsüzerinde, kuyruk yöneticisi yeni küme iş yükü çıkışını kuyruk yöneticisinin bir sonraki başlatışındaki çıkışını yükler.

Kuyruk yöneticisi tanımlaması küme iş yükü çıkış programı adı içermiyorsa, küme iş yükü çıkışı çağrılmaz.

Çıkış parametresi yapısındaki bir küme iş yükü çıkışa çeşitli veriler iletilir, MQWXP:

- İleti tanımlaması yapısı ( MQMD).
- İleti uzunluğu parametresi.
- İletin bir kopyası ya da iletinin bir parçası.

z/OS dışı platformlarda, CLWLMode=FASTkullanıyorsanız, her bir işletim sistemi işlemi çıkışa ait kendi kopyasını yükler. Kuyruk yöneticisine yapılan farklı bağlantılar, çıkışa ilişkin farklı kopyaların çağrılmasına neden olabilir. Çıkış varsayılan güvenli kipte çalıştırılırsa, CLWLMode=SAFE, çıkışa ait tek bir kopya kendi ayrı işleminde çalışır.

## **Küme iş yükü çıkışları yazılıyor**

**z/OS** Küme iş yükü çıkışlarını z/OSiçin yazmakla ilgili bilgi için bkz. [“IBM MQ for z/OSiçin küme iş yükü çıkış programlaması” sayfa 1029.](#)

**Multi** Çoklu platformlar için, küme iş yükü çıkışlarının MQI çağrıları kullanılmaması gerekir. diğer açıdan, küme iş yükü çıkış programlarını yazma ve derlemeye yönelik kurallar, kanal çıkış programları için geçerli olan kurallar gibidir. Follow the steps in [“Writing exits and installable services on UNIX, Linux and Windows” sayfa 968,](#) and use the sample program, [“Örnek küme iş yükü çıkışı” sayfa 1028](#) to help write and compile your exit.

Kanal çıkışlarına ilişkin daha fazla bilgi için bkz. [“Kanal-çıkış programları yazılıyor” sayfa 996.](#)

## Küme iş yükü çıkışlarının yapılandırılması

You name cluster workload exits in the queue manager definition by specifying the cluster workload exit attribute on the ALTER QMGR command. 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 içinde ve C içinde sağlanır. Çevirici sürümü CSQ4BAF1 adını verilir ve th1qua1 .SCSQASMSkitaplığında bulunabilir. C sürümüne CSQ4BCF1 adı verilir ve th1qua1 .SCSQC37Sürümü,foundkitaplığında bulunabilir. th1qua1 , kurulumunuzdaki IBM MQ veri kümeleri için hedef kitaplık üst düzey niteleyicidir.

#### Multi IBM MQ for Multiplatforms

Örnek küme iş yükü çıkış programı C ' de sağlanır ve adı amqsw1m0 . olarak adlandırılır. Bu öge aşağıdaki yerde bulunabilir:

Platform	filePath
 AIX	MQ_INSTALLATION_PATH/samp
 Solaris	
 Windows	MQ_INSTALLATION_PATH \Tools\c\Samples
 IBM i	qmqm kitaplığı

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Bu örnek çıkış, kuyruk yöneticisi kullanılamaz duruma gelmediği sürece, tüm iletileri belirli bir kuyruk yöneticisine yönlendirir. Bu işlem, iletileri başka bir kuyruk yöneticisine yönelterek kuyruk yöneticisinin başarısızlığa uğramasını sağlar.

İletilerin hangi kuyruk yöneticisini göndermesini istediğinizi belirtin. Kuyruk yöneticisi tanımlamasındaki CLWLDATA özniteindeki küme alıcısı kanalının adını belirtin. Örneğin:

```
ALTER QMGR CLWLDATA(' my-cluster-name. my-queue-manager ')
```

Çıkışı etkinleştirmek için, CLXLEXIT özniteisinde tam yolunu ve adını belirtin:

  UNIX and Linux'ta:

```
ALTER QMGR CLWLEXIT(' path /amqsw1m(cwlFunction)')
```

**Windows** Windows'ta:

```
ALTER QMGR CLWLEXIT(' path \amqswlm(c1wlFunction)')
```

**z/OS** z/OS'ta:

```
ALTER QMGR CLWLEXIT(CSQ4BxF1)
```

Burada x , kullanmakta olduğunuz sürümün programlama diline bağlı olarak 'A' ya da 'C' olur.

**IBM i** IBM i' ta aşağıdaki komutlardan birini kullanın:

- MQSC komutunu kullanın:

```
ALTER QMGR CLWLEXIT('AMQSWLM library')
```

Hem program adı, hem de kitaplık adı 10 karakter kaplar ve gerekirse boşlukla doldurulmuyorlar.

- CL komutunu kullanın:

```
CHGMQM MQMNAME( qmgrname ) CLWLEXIT(' library /AMQSWLM')
```

Şimdi, sağlanan iş yükü yönetimi algoritmasını kullanmak yerine, IBM MQ bu çıkışı, tüm iletileri seçtiğiniz kuyruk yöneticinize yönleltmek için 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 katı programlama kuralına tabidir. Bekleme içeren en çok SVC komutunu kullanmaktan kaçınin ya da bir iş yükü çıkışında STAE ya da ESTAE kullanılmasını öner.

Küme iş yükü çıkışları, bir z/OS **LINK** tarafından aşağıdaki gibi çağrılır:

- Yetkili olmayan sorun programı durumu
- Birincil adres alanı denetim kipi
- Çapraz olmayan bellek kipi
- Erişim olmayan kayıt kipi
- 31 bit adresleme kipi
- Depolama anahtarı 8
- Program Anahtar Maskesi 8
- TCB anahtarı 8

**V9.1.0** Bağlantı düzenlenmiş modülleri, kanal başlatıcısının başlatılan görev yordamında belirtilen CSQXLIB DD deyimi tarafından belirtilen veri kümesine yerleştirin. Yükleme birimlerinin adları, kuyruk yöneticisi tanımlamasındaki iş yükü çıkış adları olarak belirtilir.

IBM MQ for z/OS için iş yükü çıkışları yazıldığında aşağıdaki kurallar geçerlidir:

- Çevirici ya da C ' de çıkış yazmanız gerekir. If you use C, it must conform to the C systems programming environment for system exits, described in the z/OS C/C++ Programlama Kılavuzu, SC09-4765.
- If using the MQXCLWLN call, link edit with CSQMFCWL, supplied in *thlqua1*. SCSQLOAD.
- Çıkışlar, CSQXLIB DD deyimi tarafından tanımlanan yetkili olmayan kitaplıklardan yüklenir. Providing CSQXLIB has DISP=SHR, exits can be updated while the queue manager is running, with the new version used in the next MQCONN thread the queue manager starts.
- Çıkışlar yeniden girişli olmalı ve sanal saklama alanı içinde herhangi bir yerde çalıştırma yeteneğine sahip olmalıdır.

- Çıkışlar, girişteki bu ortama geri dönmeli olarak ilk durumuna getirilmelidir.
- Çıkışlar, elde edilen herhangi bir depolamayı serbest bırakmalı ya da sonraki bir çıkış çağrısıyla depolamanın serbest bırakıldığından emin olmalıdır.
- MQI çağrılarına izin verilmez.
- Bir bekleme durumu kuyruk yöneticisinin başarımını ciddi şekilde geçersiz kıldığından, çıkışlar bekleme nedeniyle neden olabilen sistem hizmetlerini kullanmamalıdır. Genel olarak, bir SVC, kişisel bilgisayar ya da G/Ç 'yi önlemekten kaçının.
- Çıkışların, bağlı oldukları alt görevler dışında ESTAE ya da SPIE yayınlamaması gerekir.

**Not:** Bir çıkışta neler yapabileceğiyle ilgili mutlak kısıtlama yoktur. Ancak, en çok SVC 'ler bekleme işlemlerini içerir, bu nedenle aşağıdaki komutlar hariç olmak üzere bu işlemleri önlemekten kaçının:

- **GETMAIN / FREEMAIN**
- **LOAD / DELETE**

Hata işleme, IBM MQ tarafından gerçekleştirilen hata işleme engeline müdahale edebileceğinden, ESTAS ve ESPEE'leri kullanmayın. IBM MQ , bir hatadan kurtarılamayabilir ya da çıkış programınız tüm hata bilgilerini alamayabilir.

The system parameter EXITLIM limits the amount of time an exit might run for. EXITLIM için varsayılan değer 30 saniyedir. If you see the return code MQRC\_CLUSTER\_EXIT\_ERROR, 2266 X '8DA ' your exit might be looping. Çıkışta 30 saniyeden uzun bir süre gerekeceğini düşünüyorsanız, EXITLIM değerini artırın.

## Yordamsal uygulama oluşturulması

Bir IBM MQ uygulamasını birden çok yordamsal dilden birine yazabilir ve uygulamayı farklı platformlarda çalıştırabilirsiniz.

### **AIX** Building your procedural application on AIX

AIX yayınları, yazdığınız programlardan yürütülebilir uygulamaların nasıl oluşturulacağını açıklar.

Bu konuda, AIX altında çalışmak üzere IBM MQ for AIX uygulamaları oluşturulurken gerçekleştirmeniz gereken ek görevler ve standart görevlerde yapılan değişiklikler ele alınmıştır. C, C++ ve COBOL desteklenmektedir. C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. [Using C++](#).

IBM MQ for AIX kullanarak yürütülebilir bir uygulama yaratmak için gerçekleştirmeniz gereken görevler, kaynak kodunuzun yazıldığı programlama diline göre değişiklik gösterir. In addition to coding the MQI calls in your source code, you must add the appropriate language statements to include the IBM MQ for AIX include files for the language that you are using. Bu dosyaların içeriğini kendinize tanıdık bir hale getiriniz. Tam açıklama için ["IBM MQ veri tanımlama dosyaları"](#) sayfa 756 'e bakın.

İş parçacıklı sunucu ya da iş parçacıklı istemci uygulamaları çalıştırdığınızda, AIXTHREAD\_SCOPE = S. ortam değişkenini ayarlayın.

### **AIX** C programlarını AIX içinde hazırlama

This topic contains information about linking libraries necessary to prepare C programs on AIX.

Önderlenmiş C programları, `MQ_INSTALLATION_PATH/samp/bin` dizininde sağlanır. ANSI derleyicisini kullanın ve aşağıdaki komutları çalıştırın. 64 bit kullanan uygulamalar hakkında daha fazla bilgi için bkz. [64 bit altyapılarda koşma standartları](#).

`MQ_INSTALLATION_PATH` , IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder.

32 bit uygulamalar için:

```
$ xlc_r -o amqspu32 amqspu0.c -I MQ_INSTALLATION_PATH/inc -LMQ_INSTALLATION_PATH/lib -lmqm
```

Burada amqspu0 , 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.

C++ programları için VisualAge C/C++ derleyicisi kullanıyorsanız, kitaplıkları bağlarken çözümlenen tüm IBM MQ simgelerini almak için -q namemangling=v5 seçeneğini eklemelisiniz.

Programları yalnızca IBM MQ MQI client for AIX kurulu bir makinede kullanmak istiyorsanız, bunları istemci kitaplığıyla (-lmqic) bağlamak için programları yeniden derleyin.

## Kitaplıkların bağlanması

Aşağıdaki kitaplıklara gereksinim duyarsınız:

- Programlarınızı, IBM MQ tarafından sağlanan uygun kitaplıkla bağlantılayın.

İş parçacıklı olmayan bir ortamda, aşağıdaki kitaplıklardan birine bağlayın:

Kitaplık dosyası	Program/çıkış tipi
libmqm.a	C sunucusu için sunucu
libmqic.a & libmqm.a	C İçin İstemci

İş parçacıklı bir ortamda, aşağıdaki kitaplıklardan birine bağlantı:

Kitaplık dosyası	Program/çıkış tipi
libmqm_r.a	C sunucusu için 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.

Programları yalnızca IBM MQ MQI client for AIX kurulu bir makinede kullanmak istiyorsanız, bunları istemci kitaplığıyla (-lmqic) bağlamak için programları yeniden derleyin.

### Not:

1. Birden çok kitaplığa bağlanamazsınız. Yani, aynı anda hem yivli hem de iş parçacıklı bir kitaplığa bağlanamazsınız.
2. Kurulabilir bir hizmet yazıyorsanız (ek bilgi için [Administering](#) başlıklı konuya bakın), iş parçacıklı olmayan bir uygulamadaki libmqmzf.a kitaplığına ve iş parçacıklı bir uygulamadaki libmqmzf\_r.a kitaplığına bağlanmanız gerekir.
3. If you are producing an application for external coordination by an XA-compliant transaction manager such as IBM TXSeries, Encina, or BEA Tuxedo, you need to link to the libmqmxa.a (or libmqmxa64.a if your transaction manager treats the 'long' type as 64 bit) and libmqz.a libraries

in a non-threaded application and to the libmqmxa\_r.a (or libmqmxa64\_r.a) and libmqz\_r.a libraries in a threaded application.

4. Güvenilir uygulamaları iş parçacıklı IBM MQ kitaplıklarına bağlamaya gerek duyarsınız. However, only one thread in a trusted application on IBM MQ on UNIX and Linux systems can be connected at a time.
5. Diğer ürün kitaplıklarından önce IBM MQ kitaplıklarını bağlamanız gerekir.

### **AIX** *Preparing COBOL programs in AIX*

Use this information when preparing COBOL programs in AIX using IBM COBOL Set and Micro Focus COBOL.

`MQ_INSTALLATION_PATH`, IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

- 32 bit COBOL kopya kitapları aşağıdaki dizine takılır:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

ve simgesel bağlantılar aşağıdaki yerde yaratılır:

```
MQ_INSTALLATION_PATH/inc
```

- 64 bit COBOL kopya kitapları aşağıdaki dizine takılır:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

Aşağıdaki örneklerde, **COBCPY** ortam değişkeni şu şekilde ayarlanır:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

32 bit uygulamalar için ve:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

64 bit uygulamalar için.

Programınızı aşağıdaki kitaplık dosyalarından biriyle bağlamaya gereksinim duyarsınız:

<b>Kitaplık dosyası</b>	<b>Program/çıkış tipi</b>
libmqmcb.a	COBOL için sunucu (iş parçacıklı uygulama)
libmqmcb_r.a	COBOL için sunucu (iş parçacıklı uygulama)
libmqicb.a	COBOL için İstemci (iş parçacıklı uygulama)
libmqicb_r.a	COBOL için İstemci (iş parçacıklı uygulama)

You can use the IBM COBOL Set compiler or Micro Focus COBOL compiler depending on the program:

- amqm ' un başlangıç programları Micro Focus COBOL derleyicisi için uygundur ve
- amq0 başlangıç programları, her iki derleyici için uygundur.

### **Preparing COBOL programs using IBM COBOL Set for AIX**

Örnek COBOL programları IBM MQ ile sağlanır. Böyle bir programı derlemek için, aşağıdaki listeden uygun komutu girin:



### 32 bit yivli olmayan sunucu uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcbl -qLIB \  
-ICOBPCPY_VALUE
```

### 32 bit yivli olmayan istemci uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -L MQ_INSTALLATION_PATH/lib -lmqicbl -qLIB \  
-ICOBPCPY_VALUE
```

### 32 bit yivli sunucu uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \  
-lmqmcbl_r -qLIB -ICOBPCPY_VALUE
```

### 32 bit yivli istemci uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -qTHREAD -L MQ_INSTALLATION_PATH/lib \  
-lmqicbl_r -qLIB -ICOBPCPY_VALUE
```

### 64 bit yivli olmayan sunucu uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib -lmqmcbl \  
-qLIB -ICOBPCPY_VALUE
```

### 64 bit yivli olmayan istemci uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -q64 -L MQ_INSTALLATION_PATH/lib -lmqicbl \  
-qLIB -ICOBPCPY_VALUE
```

### 64 bit yivli sunucu uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \  
-lmqmcbl_r -qLIB -ICOBPCPY_VALUE
```

### 64 bit yivli istemci uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -q64 -qTHREAD -L MQ_INSTALLATION_PATH/lib \  
-lmqicbl_r -qLIB -ICOBPCPY_VALUE
```

## COBOL programlarının Micro Focus COBOL

Programınızı derlemeden önce ortam değişkenlerini aşağıdaki gibi ayarlayın:

```
export COBPCPY=COBPCPY_VALUE  
export LIBPATH=MQ_INSTALLATION_PATH/lib:$LIBPATH
```

Bir 32 bit COBOL programını Micro Focus COBOL kullanarak derlemek için şunu girin:

- COBOL için sunucu

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcbl
```

- COBOL için İstemci

```
$ cob32 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicbl
```

- COBOL İçin İş Y

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqmcbr
```

- COBOL için İş parçacığı

```
$ cob32 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib -lmqicbr
```

Bir 64 bit COBOL programını Micro Focus COBOL kullanarak derlemek için şunu girin:

- COBOL için sunucu

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcbr
```

- COBOL için İstemci

```
$ cob64 -xvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicbr
```

- COBOL İçin İş Y

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmcbr
```

- COBOL için İş parçacığı

```
$ cob64 -xtvP amqminqx.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqicbr
```

Burada amqminqx , örnek bir programdır

Ayarlamanız gereken ortam değişkenlerine ilişkin açıklamalar için Micro Focus COBOL belgelerine bakın.

### **AIX** *Preparing CICS application programs in AIX*

Use this information when preparing CICS programs in AIX.

CICS bağlantısını IBM MQ ile bağlamak için *XA anahtarı* modüllerini kullanın. XA anahtar yapısına ilişkin ek bilgi için *XA anahtarı yapıları* başlıklı konuya bakın.

Diğer hareket iletilerine ilişkin XA anahtarlarını geliştirmenizi sağlamak için örnek kaynak kod dosyası sağlanmıştır. Sağlanan anahtar yükleme modülünün adı *Çizelge 149 sayfa 1034* listesinde yer alıyor.

<i>Çizelge 149. AIX üzerindeki CICS uygulama programları için temel kod: XA kullanıma hazırlama yordamı</i>		
<b>Tanım</b>	<b>C (kaynak)</b>	<b>C (exec)-sitenize ekle XAD.Stanza</b>
XA kullanıma hazırlama yordamı	amqzsc.c	amqzsc - AIX için CICS

Use the prebuilt version of the IBM MQ switch load file *amqzsc*, which is provided with the product.

C işlemlerinizi her zaman iş parçacığı korumalı IBM MQ kitaplığınızla (*libmqm\_r.a*) *bağlayın*. and your COBOL transactions with the COBOL library *libmqmcb\_r.a*.

You can find more information about supporting CICS transactions in the *Administering IBM MQ System Administration Guide*.

### **AIX** *TXSeries CICS desteği*

AIX üzerinde IBM MQ , XA arabirimini kullanarak TXSeries CICS ' ı destekler. CICS uygulamalarının, IBM MQ kitaplıklarının iş parçacıklı sürümüyle bağlantılı olduğundan emin olun.

CICS program programlarını IBM COBOL Set for AIX ya da Micro Focus COBOL kullanarak çalıştırabilirsiniz. The following sections describe the difference between running CICS programs on IBM COBOL Set for AIX and Micro Focus COBOL.

C ya da COBOL ' de aynı CICS bölgesine yüklenen IBM MQ programlarını yazma. Aynı CICS bölgesine C ve COBOL MQİ çağrılarının birleşiminden oluşan bir birleşim yapamazsınız. Most MQİ calls in the second language used fail with a reason code of MQRC\_HOBJ\_ERROR.

## Preparing CICS COBOL programs using IBM COBOL Set for AIX

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

IBM COBOL olanağını kullanmak için aşağıdaki adımları izleyin:

1. Aşağıdaki ortam değişkenini dışa aktarın:

```
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 'dir.

2. Programı çevirerek, derleyin ve bağlayın:

```
cicstcl -l IBMCOB yourprog.ccp
```

## Preparing CICS COBOL programs using Micro Focus COBOL

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Micro Focus COBOL olanağını kullanmak için aşağıdaki adımları izleyin:

1. Aşağıdaki komutu kullanarak, IBM MQ COBOL yürütme ortamı kitaplık modülünü yürütme ortamı kitaplığına ekleyin:

```
cicsmkcobol -L/usr/lib/dce -L MQ_INSTALLATION_PATH/lib \  
MQ_INSTALLATION_PATH/lib/libmqmcbt.o -lmqe_r
```

**Not:** `cicsmkcobol` ile, IBM MQ , C programlama dilinde COBOL uygulamanızın MQİ çağrılarını gerçekleştirmenize izin vermez.

Var olan uygulamalarınızda bu tür çağrılar varsa, bu işlevleri COBOL uygulamalarından kendi kitaplığınıza taşımanız önerilir; örneğin, `myMQ . so`. After moving the functions, do not include the IBM MQ library `libmqmcbt.o` when building the COBOL application for CICS.

Ayrıca, COBOL uygulamanızın COBOL MQİ çağrısı yapmazsa, `libmqmz_r` ile `cicsmkcobol` bağlantısını bağlamayın.

Bu, Micro Focus COBOL dil yöntemi dosyasını oluşturur ve CICS Runtime COBOL kitaplığını UNIX and Linux sistemlerinde IBM MQ ' u aramasına olanak sağlar.

**Not:** `cicsmkcobol` komutunu yalnızca aşağıdaki ürünlerden birini kurduğunuzda çalıştırın:

- Micro Focus COBOL ' in yeni sürümü veya
- New version or release of CICS for AIX
- Desteklenen herhangi bir veritabanı ürününün yeni sürümü ya da yayın düzeyi (yalnızca COBOL işlemleri için)
- New version or release of IBM MQ

2. Aşağıdaki ortam değişkenini dışa aktarın:

```
COBCPY= MQ_INSTALLATION_PATH/inc export COBCPY
```

3. Programı çevirerek, derleyin ve bağlayın:

```
cicstcl -l COBOL -e yourprog.ccp
```

## CICS C programlarının hazırlanması

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Standart CICS olanaklarını kullanarak CICS C programlarını oluşturun:

1. Aşağıdaki ortam değişkenlerinin **bir** ögesini dışa aktarın:
  - `LDflags = "-L/ MQ_INSTALLATION_PATH lib -lmqm_r" LDFLAGS` dışa aktar
  - `USERLIB = "-L MQ_INSTALLATION_PATH lib -lmqm_r" export USERLIB`
2. Programı çevirerek, derleyin ve bağlayın:

```
cicstcl -l C amqscic0.ccs
```

## CICS C örnek hareketi

Bir AIX IBM MQ işlemi için Örnek C kaynağı, `AMQSCIC0.CCS`. Hareket, `SYSTEM.SAMPLE.CICS`iletim kuyruğundan ileti okur.Varsayılan kuyruk yöneticisinde `WORKQUEUE` ve bunları, iletinin iletim üstbilgisinde yer alan bir kuyruk adıyla yerel kuyruğa yerleştirir. `SYSTEM.SAMPLE.CICS`kuyruğuna herhangi bir hata gönderilsin.DQ. Örnek MQSC komut kütüğünü ( `AMQSCIC0.TST` , bu kuyrukları ve örnek giriş kuyruklarını yaratmanızı sağlar.

IBM i

## Building your procedural application on IBM i

The IBM i publications describe how to build executable applications from the programs that you write, to run with IBM i on iSeries or System i systems.

Bu konuda, IBM i sistemlerinde çalıştırılacak IBM MQ for IBM i yordamsal uygulamaları oluştururken gerçekleştirmeniz gereken ek görevleri ve standart görevlerdeki değişiklikleri ele alan bilgiler yer alır. COBOL, C, C + +, Java ve RPG programlama dilleri desteklenmektedir. C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. [Using C++](#). Java programlarınızı hazırlamaya ilişkin bilgi için bkz. [IBM MQ classes for Java](#).

Yürütülebilir bir IBM MQ for IBM i uygulaması oluşturmak için gerçekleştirmeniz gereken görevler, kaynak kodun yazıldığı programlama diline bağlıdır. Kaynak kodunuzda, MQI çağrılarının kodlamaya ek olarak, kullanmakta olduğunuz dile ilişkin IBM MQ for IBM i veri tanımlama dosyalarının içerileceği uygun dil deyimlerini eklemelisiniz. Bu dosyaların içeriğini kendinize tanıdık bir hale getiriniz. Tam açıklama için "[IBM MQ veri tanımlama dosyaları](#)" sayfa 756 ' e bakın.

IBM i

## C programlarını IBM içinde hazırlama

IBM MQ for IBM i , 100 MB ' ye kadar olan iletileri destekler. ILE C içinde yazılan uygulama programları, 16 MB ' den büyük IBM MQ iletilerini destekleyen uygulama programları, bu iletiler için yeterli bellek ayırmak üzere *Teraspace* derleyici seçeneğini kullanmamız gerekir.

C derleyici seçenekleri hakkında daha fazla bilgi için *WebSphere Development Studio ILE C/C++ Programmer's Guide* adlı yayına bakın.

To compile a C module, you can use the IBM i command, `CRTCMOD`. Derlediğinizde, include files (`QMQM`) içeren kitaplığın kitaplık listesinde yer aldığından emin olun.

Daha sonra, `CRTPGM` komutunu kullanarak derleyicinin çıkışını hizmet programıyla bağlamanız gerekir.

Çizelge 150. İş parçacıklı ortamdaki CRTPGM örneği	
Komut	Program/çıkış tipi
<pre>CRTPGM PGM( pgmname ) MODULE( pgmname ) BNDSRVPGM(QMQM/LIBMQM)</pre>	C için sunucu ya da istemci

Burada *pgmname* , programınızın adıdır.

İş parçacıklı bir ortama ilişkin komut örneği:

Çizelge 151. İş parçacıklı ortamdaki CRTPGM örneği	
Komut	Program/çıkış tipi
<pre>CRTPGM PGM( pgmname ) MODULE( pgmname ) BNDSRVPGM(QMQM/LIBMQM_R)</pre>	C için sunucu ya da istemci

Burada *pgmname* , programınızın adıdır.

Aşağıdaki çizelgelerde, iş parçacıklı bir ortamda ve iş parçacıklı bir ortamda IBM i 'de C programları hazırlanırken gereken kitaplıklar listelenmiştir.

Çizelge 152. İş parçacıklı olmayan ortam	
Kitaplık dosyası	Program/çıkış tipi
LIBMQM	C sunucusu için sunucu
LIBMQIC & LIBMQM	C İçin İstemci

Çizelge 153. Işıklı ortam	
Kitaplık dosyası	Program/çıkış tipi
LIBMQM_R	C sunucusu için sunucu
LIBMQIC_R & LIBMQM_R	C İçin İstemci

## IBM i **Preparing COBOL programs in IBM i**

COBOL programlarını IBM i 'ta hazırlama ve COBOL programı içinden MQI' ye erişme yöntemi hakkında bilgi edinin.

### Bu görev hakkında

COBOL programlarından MQI ' ye erişmek için IBM MQ for IBM i , hizmet programları tarafından sağlanan bir sınır yordamsal çağrı arabirimi sağlar. Bu, IBM MQ for IBM i içindeki tüm MQI işlevlerine ve iş parçacıklı uygulamalar için destek sağlar. Bu arabirim yalnızca ILE COBOL derleyicisiyle birlikte kullanılabilir.

Standart COBOL CALL sözdizimi, MQI işlevlerine erişmek için kullanılır.

MQI ile kullanılmak üzere adlandırılmış değişmezleri ve yapı tanımlamalarını içeren COBOL kopya dosyaları, QMQM/QCBLLSRC kaynak fiziksel dosyasında yer alır.

COBOL kopya dosyaları, dizgi sınırlayıcı olarak tek tırnak işareti (') kullanır. IBM i COBOL derleyicileri, sınırlayıcının tırnak işareti (") olduğunu varsaymaktadır. Derleyicilerin uyarı iletileri oluşturmasını önlemek için, **CRTCBLPGM**, **CRBNDCBL** ya da **CRTCBLMOD** komutlarında **OPTION (\*APOST)** seçeneğini belirleyin.

Derleyiciyi, COBOL kopya dosyalarındaki dizgi sınırlayıcı olarak tek tırnak işareti (') işaretini kabul etmek için, \APOST derleyici seçeneğini kullanın.

**Not:** Dinamik çağrı arabirimi, IBM MQ 9.0 ya da daha sonraki bir yayın düzeyiyle birlikte sağlanmaz. Bağlı yordam çağırma arabirimini kullanmak için aşağıdaki adımları tamamlayın.

## Yordam

1. Parametreyi belirterek **CRTCBLMOD** derleyicisini kullanarak bir modül yaratın:

```
LINKLIT(*PRC)
```

2. Uygun değiştirgeyi belirterek program nesnesini yaratmak için **CRTPGM** komutunu kullanın:  
İş parçacıklı olmayan uygulamalar için:

```
BNDSRVPGM(QMQM/AMQOSTUB)      Server for COBOL for non-threaded applications
BNDSRVPGM(QMQM/AMQCSTUB)      Client for COBOL for non-threaded applications
```

İş parçacıklı uygulamalar için:

```
BNDSRVPGM(QMQM/AMQOSTUB_R)     Server for COBOL for threaded applications
BNDSRVPGM(QMQM/AMQCSTUB_R)     Client for COBOL for threaded applications
```

**Not:** V4R4 ILE COBOL derleyicisini kullanan ve PROCESS deyiminde THREAD (SERIALIZE) seçeneğini içeren programlar dışında, COBOL programları iş parçacıklı IBM MQ kitaplıklarını kullanmamalıdır. Bir COBOL programı bu şekilde iş parçacığı güvenliğini sağlasa bile, uygulamayı tasarladığınızda dikkatli olun; çünkü, iş parçacığı (SERIALIZE), COBOL yordamlarının modül düzeyinde diziselleştirilmesini zorlar ve genel başarımı etkileyebilir.

Ek bilgi için *WebSphere Development Studio: ILE COBOL Programmer's Guide* ve *WebSphere Development Studio: ILE COBOL Reference* adlı yayına bakın.

Bir CICS uygulamasını derlemeye ilişkin ek bilgi için *CICS for IBM i Application Programming Guide*, SC41-5454 adlı yayına bakın.

## IBM i **Preparing CICS programs in IBM i**

IBM i' ta CICS programlarını hazırlarken gerekli olan adımları öğrenin.

EXEC CICS deyimlerini ve MQI çağrılarını içeren bir program yaratmak için aşağıdaki adımları gerçekleştirin:

1. Gerekirse, CRTICSCMAP komutunu kullanarak eşlemleri hazırlayın.
2. EXEC CICS komutlarını yerel dil deyimlerine çevirin. C programı için CRTICSC komutunu kullanın. Bir COBOL programı için CRTICSCBL komutunu kullanın.  
  
CCRTICSC ya da CRTICSCBL komutuna CICSOPT(\*NOGEN) ekleyin. Bu işlem, uygun CICS ve IBM MQ hizmet programlarını eklemenize olanak sağlamak için bu işleme devam eder. Bu komut, kodu varsayılan olarak QTEMP/QACYCICSiçine yerleştirir.
3. CRTCMOD komutunu (C programı için) ya da CRTCBLMOD komutunu (COBOL programı için) kullanarak kaynak kodu derleyin.
4. Derlenmiş kodu uygun CICS ve IBM MQ hizmet programlarıyla ilişkilendirmek için CRTPGM ' yi kullanın. Bu, yürütülebilir programı oluşturur.

Aşağıdaki gibi bir kod örneği aşağıdaki gibidir (sevk edilen CICS örnek programını derler):

```
CRTICSC 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 **RPG programlarının IBM içinde hazırlanması**

IBM MQ for IBM ikullanıyorsanız, uygulamalarınızı RPG ' de yazabilirsiniz.

Ek bilgi için “Coding IBM MQ programs in RPG (IBM i only)” sayfa 1091 konusuna bakın ve [IBM i Application Programming Reference \(ILE/RPG\)](#) belgesine bakın.

## IBM i **IBM için SQL programlama konuları**

SQL kullanarak IBM i üzerinde bir uygulama oluştururken gereken adımlar hakkında bilgi edinin.

Programınızda EXEC SQL deyimleri ve MQI çağrıları varsa, aşağıdaki adımları gerçekleştirin:

1. EXEC SQL komutlarını yerel dil deyimlerine çevirin. CRTSQLCI komutunu C programı için kullanın. COBOL programı için CRTSQLCBLI (CRTSQLCBLI) komutunu kullanın.  
CRTSQLCI ya da CRTSQLCBLI komutuna OPTION (\*NOGEN) ögesini ekleyin. Bu işlem, uygun IBM MQ hizmet programlarını eklemenize olanak sağlamak için bu işleme devam eder. Bu komut, kodu varsayılan olarak QTEMP/QSQLTEMP kodlarına koyar.
2. CRTCMOD komutunu (C programı için) ya da CRTCLMOD komutunu (COBOL programı için) kullanarak kaynak kodu derleyin.
3. Derlenmiş kodu uygun IBM MQ hizmet programlarıyla ilişkilendirmek için CRTPGM ' yi kullanın. Bu, yürütülebilir programı oluşturur.

Bu kodun bir örneği aşağıdaki gibidir (bir programı sıkıştırır, SQLTEST, kitaplık, SQLUSER):

```
CRTSQLCI OBJ(MQTEST/SQLTEST) SRCFILE(SQLUSER/QCSRC) +
          SRCMBR(SQLTEST) OUTPUT(*PRINT) OPTION(*NOGEN)
CRTCMOD  MODULE(MQTEST/SQLTEST) +
          SRCFILE(QTEMP/QSQLTEMP) OUTPUT(*PRINT)
CRTPGM   PGM(MQTEST/SQLTEST) +
          BNDSRVPGM(QMQM/LIBMQIC)
```

## Linux **Building your procedural application on Linux**

This information describes the additional tasks, and the changes to the standard tasks, that you must perform when building IBM MQ for Linux applications to run.

C ve C++ desteklenir. C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. [Using C++](#).

## Linux **C programlarını Linux içinde hazırlama**

Önderlenmiş C programları, `MQ_INSTALLATION_PATH/samp/bin` dizininde sağlanır. Kaynak koddan bir örnek oluşturmak için gcc derleyicisini kullanın.

`MQ_INSTALLATION_PATH`, IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Normal ortamınızda çalışın. 64 bit kullanan uygulamalar hakkında daha fazla bilgi için bkz. [64 bit altyapılarda koşma standartları](#).

## Kitaplıkların bağlanması

The following tables lists the libraries that are needed when preparing C programs on Linux.

- Programlarınızı, IBM MQ tarafından sağlanan uygun kitaplıkla ilişkilendirmeniz gerekir.

İş parçacıklı olmayan bir ortamda, aşağıdaki kitaplıklardan yalnızca birine bağlayın:

Kitaplık dosyası	Program/çıkış tipi
libmqm.so	C sunucusu için sunucu
libmqic.so & libmqm.so	C İçin İstemci

İş parçacıklı bir ortamda, aşağıdaki kitaplıklardan yalnızca birine bağlayın:

Kitaplık dosyası	Program/çıkış tipi
libmqm_r.so	C sunucusu için sunucu
libmqic_r.so & libmqm_r.so	C İçin İstemci

**Not:**

1. Birden çok kitaplığa bağlanamazsınız. Yani, aynı anda hem yivli hem de iş parçacıklı bir kitaplığa bağlanamazsınız.
2. Kurulabilir bir hizmet yazıyorsanız (ek bilgi için [Administering](#) başlıklı konuya bakın), libmqmzf.so kitaplığına bağlanmanız gerekir.
3. If you are producing an application for external coordination by an XA-compliant transaction manager such as IBM TXSeries Encina, or BEA Tuxedo, you need to link to the libmqmxa.so (or libmqmxa64.so if your transaction manager treats the 'long' type as 64 bit) and libmqz.so libraries in a non-threaded application and to the libmqmxa\_r.so (or libmqmxa64\_r.so) and libmqz\_r.so libraries in a threaded application.
4. Diğer ürün kitaplıklarından önce IBM MQ kitaplıklarını bağlamanız gerekir.

**Linux** 31 bit uygulamalar oluşturuluyor

Bu konu, çeşitli ortamlarda 31 bit programları oluşturmak için kullanılan komutlara ilişkin örnekleri içerir.

MQ\_INSTALLATION\_PATH, IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder.

**C istemci uygulaması, 31-bit, yivsiz**

```
gcc -m31 -o famqspu32 amqspu0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqic
```

**C istemcisi uygulaması, 31-bit, yivli**

```
gcc -m31 -o amqspu32_r amqspu0.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 uygulaması, 31-bit, iş parçacıklı olmayan**

```
gcc -m31 -o amqspu32 amqspu0.c -I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -Wl,-rpath=MQ_INSTALLATION_PATH/lib -Wl,-rpath=/usr/lib -lmqm
```

**C sunucusu uygulaması, 31-bit, iş parçacığı**

```
gcc -m31 -o amqspu32_r amqspu0.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 parçacıklı olmayan**

```
g++ -m31 -fsigned-char -o imqspu32 imqspu.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 imqspu32_r imqspu.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 parçacıklı olmayan

```
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, yivsiz

```
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 istemcisi çıkışı, 31-bit, yivli

```
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 olmayan

```
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 sunucusu 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 bitlik uygulamalar oluşturuluyor

Bu konu, çeşitli ortamlarda 32 bit programları oluşturmak için kullanılan komutlara ilişkin örnekleri içerir.

`MQ_INSTALLATION_PATH`, IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder.

## C istemcisi uygulaması, 32 bit, iş parçacıklı olmayan

```
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, yivli

```
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 sunucusu uygulaması, 32 bitlik, iş parçacıklı olmayan

```
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 sunucusu uygulaması, 32 bit, yivli

```
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 olmayan

```
g++ -m32 -fsigned-char -o imqsputc_32 imqsputc.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, yivli iş parçacığı

```
g++ -m32 -fsigned-char -o imqsputc_32_r imqsputc.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 bitlik, iş parçacıklı olmayan

```
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, yivli 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, yivsiz olmayan

```
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 istemcisi çıkışı, 32 bit, yivli 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 sunucusu çıkışı, 32 bitlik, iş parçacıklı olmayan

```
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 sunucusu çıkışı, 32 bit, yivli iş parçacığı

```
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 programları oluşturmak için kullanılan komutlara ilişkin örnekler içerir. MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

### C istemcisi uygulaması, 64 bitlik, iş parçacıklı olmayan

```
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 istemcisi uygulaması, 64 bitlik, iş parçacıklı

```
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 sunucusu uygulaması, 64 bitlik, iş parçacıklı olmayan

```
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 sunucusu uygulaması, 64 bitlik, iş parçacıklı

```
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 bitlik, iş parçacıklı olmayan

```
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 bitlik, iş parçacıklı

```
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 bitlik, iş parçacıklı olmayan

```
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 bitlik, iş parçacıklı

```
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 bitlik, iş parçacıklı olmayan

```
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 istemcisi çıkışı, 64 bit, yivli

```
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 sunucusu çıkışı, 64 bitlik, iş parçacıklı olmayan

```
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 sunucusu çıkışı, 64 bitlik, iş parçacıklı

```
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 **Preparing COBOL programs in Linux**

Learn about preparing COBOL programs in Linux and preparing COBOL programs using IBM COBOL for Linux on x86 and Micro Focus COBOL.

`MQ_INSTALLATION_PATH`, IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder.

1. 32 bit COBOL kopya kitapları aşağıdaki dizine takılır:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

ve simgesel bağlantılar aşağıdaki yerde yaratılır:

```
MQ_INSTALLATION_PATH/inc
```

2. 64 bit altyapılarda, 64 bit COBOL kopya kitapları aşağıdaki dizine kurulur:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

3. Aşağıdaki örneklerde COBCPY to:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

32 bit uygulamalar için ve:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

64 bit uygulamalar için.

Programınızı aşağıdaki bağlantılardan biriyle ilişkilendirmeniz gerekir:

Kitaplık dosyası	Program/çıkış tipi
libmqmcb.so	COBOL için sunucu
libmqicb.so	COBOL için İstemci
libmqmcb_r.so	COBOL için sunucu (iş parçacıklı uygulama)

Kitaplık dosyası	Program/çıkış tipi
libmqicb_r.so	COBOL için İstemci (iş parçacıklı uygulama)

## Preparing COBOL programs using IBM COBOL for Linux on x86

Örnek COBOL programları, IBM MQ ile sağlanır. Böyle bir programı derlemek için, aşağıdaki listeden uygun komutu girin:

### 32 bitlik iş parçacıklı olmayan sunucu uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-L MQ_INSTALLATION_PATH/lib -lmqmc -ICOBPCY_VALUE
```

### 32 bitlik iş parçacıklı istemci uygulaması

```
$ cob2 -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-L MQ_INSTALLATION_PATH/lib -lmqicb -ICOBPCY_VALUE
```

### 32 bit yivli sunucu uygulaması

```
$ cob2_r -o amq0put0 amq0put0.cbl -q"BINARY(BE)" -q"FLOAT(BE)" -q"UTF16(BE)"
-qTHREAD -L MQ_INSTALLATION_PATH/lib -lmqmc_r -ICOBPCY_VALUE
```

### 32 bit yivli 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 -ICOBPCY_VALUE
```

## COBOL programlarının Micro Focus COBOL

Programınızı derlemeden önce ortam değişkenlerini aşağıdaki gibi ayarlayın:

```
export COBPCY=COBPCY_VALUE
export LIB= MQ_INSTALLATION_PATH lib:$LIB
```

Desteklenen, Micro Focus COBOL kullanan 32 bit COBOL programını derlemek için şunu girin:

```
$ cob32 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqmc 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 -lmqmc_r Threaded Server for COBOL
$ cob32 -xtvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib -lmqicb_r Threaded Client for COBOL
```

Bir 64 bit COBOL programını Micro Focus COBOL kullanarak derlemek için şunu girin:

```
$ cob64 -xvP amqsput.cbl -L MQ_INSTALLATION_PATH/lib64 -lmqmc 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 -lmqmc_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 değişkenlerine ilişkin açıklamalar için Micro Focus COBOL belgelerine bakın.

## Building your procedural application on Solaris

This information describes the additional tasks, and the changes to the standard tasks, that you must perform when building IBM MQ for Solaris applications to run under Solaris.

COBOL, C ve C++ programlama dilleri desteklenmektedir. C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. [Using C++](#).

Kaynak kodunuzda MQI çağrılarını kodlamaya ek olarak, uygun içerme dosyalarını eklemelisiniz. Bu dosyaların içeriğini kendinize tanıdık bir hale getiriniz. Tam açıklama için "[IBM MQ veri tanımlama dosyaları](#)" sayfa 756 ' e bakın.

Bu konu boyunca ters eğik çizgi (\) karakteri, uzun komutları birden çok satıra bölmek için kullanılır. Bu karakteri girmeyin, her komutu tek bir satır olarak girin.

## **Solaris** C programlarını Solarisiçinde hazırlama

Önderlenmiş C programları, `MQ_INSTALLATION_PATH/samp/bin` dizininde sağlanır.

`MQ_INSTALLATION_PATH`, IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

64 bit kullanan uygulamalar hakkında daha fazla bilgi için bkz. [64 bit altyapılarda koşma standartları](#).

Programları yalnızca IBM MQ MQI client for Solaris kurulu olan bir makinede kullanmak istiyorsanız, bunları istemci kitaplığıyla ( `-lmqic`) bağlamak için bu programları derleyin.

If you use the unsupported compiler `/usr/ucb/cc`, your application might compile and link successfully. Ancak, uygulamayı çalıştırdığınızda, kuyruk yöneticisine bağlanma girişiminde bulunduğu başarısız olur.

**Not:** Intel sistemlerinde çalışırken, FIPS 140-2 uyumlu işlem için yapılandırılan 32 bit Solaris x86 SSL ve TLS istemcileri başarısız olur. Bu hata, FIPS 140-2 uyumlu GSKit-Crypto Solaris x86 32 bit kitaplık dosyası Intel yonga setine yüklenmediğinden ortaya çıkar. Etkilenen sistemlerde, istemci hata günlüğünde AMQ9655 hatası bildirildi. Bu sorunu çözmek için, 64 bitlik kod etkilenmediğinden, FIPS 140-2 uyumluluğunu geçersiz kılın ya da istemci uygulamasını 64 bitlik olarak yeniden derleyin.

## Kitaplıkların bağlanması

Uygulama tipiniz için uygun olan IBM MQ kitaplıklarıyla bağlantı vermelisiniz:

Kitaplık dosyaları	Program/çıkış tipi
<code>libmqm.so</code>	C sunucusu için sunucu
<code>libmqic.so &amp; libmqm.so</code>	C İçin İstemci

### Not:

1. Kurulabilir bir hizmet yazıyorsanız (daha fazla bilgi için bkz. [Administering](#)), `libmqmzf.so` kitaplığına bağlantı.
2. If you are producing an application for external coordination by an XA-compliant transaction manager such as IBM TXSeries Encina, or BEA Tuxedo, you must link to the `libmqmxa.so` (or `libmqmxa64.so` if your transaction manager treats the 'long' type as 64 bit) and `libmqz.so` libraries.
3. Diğer ürün kitaplıklarından önce IBM MQ kitaplıklarını bağlamanız gerekir.

## **Solaris** Solaris x86-64üzerinde uygulama oluşturma

Bu konuda, Solaris x86-64 platformunda çeşitli ortamlarda programlar oluşturmak için kullanılan komutlara ilişkin örnekler yer alır.

`MQ_INSTALLATION_PATH`, IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

### C istemci uygulaması, 32 bit

```
cc -xarch=386 -mt -o amqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

### C istemci uygulaması, 64 bit

```
cc -xarch=amd64 -mt -o amqsputc_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic -lsocket
-lnsl -ldl
```

### C sunucusu uygulaması, 32 bit

```
cc -xarch=386 -mt -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

### C sunucusu uygulaması, 64 bit

```
cc -xarch=amd64 -mt -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm -lsocket
-lnsl -ldl
```

### C++ istemci uygulaması, 32 bit

```
CC -xarch=386 -mt -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as -lmqic -lsocket -lnsl -ldl
```

### C++ istemci uygulaması, 64 bit

```
CC -xarch=amd64 -mt -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as
-limb23as
-lmqic -lsocket -lnsl -ldl
```

### C++ sunucu uygulaması, 32 bit

```
CC -xarch=386 -mt -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc -L
MQ_INSTALLATION_PATH/lib
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm
-lsocket -lnsl -ldl
```

### C++ sunucu uygulaması, 64 bit

```
CC -xarch=amd64 -mt -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
-limb23as -lmqm
-lsocket -lnsl -ldl
```

### C istemcisi çıkışı, 32 bit

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib
-R/usr/lib/32
-lmqic -lsocket -lnsl -ldl
```

### C istemci çıkışı, 64 bit

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64
-R/usr/lib/64
-lmqic -lsocket -lnsl -ldl
```

### C sunucusu çıkışı, 32 bit

```
cc -xarch=386 -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib
-R/usr/lib/32
-lmqm -lsocket -lnsl -ldl
```

## C sunucusu çıkışı, 64 bit

```
cc -xarch=amd64 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c  
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64  
-R/usr/lib/64  
-lmqm -lsocket -lnsl -ldl
```

### *Building applications on Solaris SPARC*

Bu konuda, Solaris SPARC platformunda çeşitli ortamlarda programlar oluşturmak için kullanılan komutlara ilişkin örnekler yer alır.

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

## C istemci uygulaması, 32 bit

```
cc -xarch=v8plus -mt -o amqsputc_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L  
MQ_INSTALLATION_PATH/lib  
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqic -lsocket -lnsl -ldl
```

## C istemci uygulaması, 64 bit

```
cc -xarch=v9 -mt -o amqsputc_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqic  
-lsocket -lnsl -ldl
```

## C sunucusu uygulaması, 32 bit

```
cc -xarch=v8plus -mt -o amqsput_32 amqsput0.c -I MQ_INSTALLATION_PATH/inc -L  
MQ_INSTALLATION_PATH/lib  
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -lmqm -lsocket -lnsl -ldl
```

## C sunucusu uygulaması, 64 bit

```
cc -xarch=v9 -mt -o amqsput_64 amqsput0.c -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm  
-lsocket -lnsl -ldl
```

## C++ istemci uygulaması, 32 bit

```
CC -xarch=v8plus -mt -o imqsputc_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqc23as -limqb23as  
-lmqic  
-lsocket -lnsl -ldl
```

## C++ istemci uygulaması, 64 bit

```
CC -xarch=v9 -mt -o imqsputc_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqc23as  
-limqb23as  
-lmqic -lsocket -lnsl -ldl
```

## C++ sunucu uygulaması, 32 bit

```
CC -xarch=v8plus -mt -o imqsput_32 imqsput.cpp -I MQ_INSTALLATION_PATH/inc -L  
MQ_INSTALLATION_PATH/lib  
-R MQ_INSTALLATION_PATH/lib -R/usr/lib/32 -limqs23as -limqb23as -lmqm  
-lsocket -lnsl -ldl
```

## C++ sunucu uygulaması, 64 bit

```
CC -xarch=v9 -mt -o imqsput_64 imqsput.cpp -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -limqs23as
```



```
-limqb23as -lmqm  
-lsocket -lnsl -ldl
```

### C istemcisi çıkışı, 32 bit

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/cliexit_32 cliexit.c  
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib  
-R/usr/lib/32  
-lmqic -lsocket -lnsl -ldl
```

### C istemci çıkışı, 64 bit

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/cliexit_64 cliexit.c  
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64  
-R/usr/lib/64  
-lmqic -lsocket -lnsl -ldl
```

### C sunucusu çıkışı, 32 bit

```
cc -xarch=v8plus -mt -G -KPIC -o /var/mqm/exits/srvexit_32 srvexit.c  
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib -R MQ_INSTALLATION_PATH/lib  
-R/usr/lib/32  
-lmqm -lsocket -lnsl -ldl
```

### C sunucusu çıkışı, 64 bit

```
cc -xarch=v9 -mt -G -KPIC -o /var/mqm/exits64/srvexit_64 srvexit.c  
-I MQ_INSTALLATION_PATH/inc -L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64  
-R/usr/lib/64  
-lmqm -lsocket -lnsl -ldl
```

## **Preparing COBOL programs in Solaris**

Learn about preparing COBOL programs in Solaris.

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

1. 32 bit COBOL kopya kitapları aşağıdaki dizine takılır:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

ve simgesel bağlantılar aşağıdaki yerde yaratılır:

```
MQ_INSTALLATION_PATH/inc
```

2. 64 bit COBOL kopya kitapları aşağıdaki dizine takılır:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

3. Aşağıdaki örneklerde COBCPY to:

```
MQ_INSTALLATION_PATH/inc/cobcpy32
```

32 bit uygulamalar için ve:

```
MQ_INSTALLATION_PATH/inc/cobcpy64
```

64 bit uygulamalar için.

Micro Focus derleyicisini kullanarak programları derleyin. Yapıları beyan eden kopya dosyaları `MQ_INSTALLATION_PATH/inc`' ta bulunur:

```
$ export LIB= MQ_INSTALLATION_PATH/lib:$LIB
$ export COBCPY="COBCPY_VALUE"
```

32 bit program derleniyor:

- \$ cob32 -xv amqs0put0.cbl -L MQ\_INSTALLATION\_PATH/lib -lmqmc  
COBOL için sunucu
- \$ cob32 -xv amqs0put0.cbl -L MQ\_INSTALLATION\_PATH/lib -lmqic  
COBOL için İstemci
- \$ cob32 -xtv amqs0put0.cbl -L MQ\_INSTALLATION\_PATH/lib -lmqmc\_r  
COBOL İçin İş Y
- \$ cob32 -xtv amqs0put0.cbl -L MQ\_INSTALLATION\_PATH/lib -lmqic\_r  
COBOL için İş parçacığı

64 bit programlar derleniyor:

- \$ cob64 -xv amqs0put0.cbl -L MQ\_INSTALLATION\_PATH/lib64 -lmqmc  
COBOL için sunucu
- \$ cob64 -xv amqs0put0.cbl -L MQ\_INSTALLATION\_PATH/lib64 -lmqic  
COBOL için İstemci
- \$ cob64 -xtv amqs0put0.cbl -L MQ\_INSTALLATION\_PATH/lib64 -lmqmc\_r  
COBOL İçin İş Y
- \$ cob64 -xtv amqs0put0.cbl -L MQ\_INSTALLATION\_PATH/lib64 -lmqic\_r  
COBOL için İş parçacığı

Burada amqs0put0.cbl örnek bir programdır.

Programınızı aşağıdakilerden biriyle bağlamanız gerekir:

- libmqmc.so  
COBOL için sunucu
- libmqic.so  
COBOL için İstemci

### **Solaris** *Preparing CICS programs in Solaris*

Learn about preparing CICS programs in Solaris.

An XA switch module is provided to enable you to link CICS with IBM MQ:

Çizelge 154. CICS uygulamaları için temel kod (Solaris)		
Tanım	C (kaynak)	C (exec)
XA kullanıma hazırlama yordamı	amqzscix.c	amqzsc- TXSeries - Solaris

İşlemlerinizi her zaman iş parçacığı güvenli IBM MQ kitaplığı libmqm.so ile bağlantılayın.

You can find more information about supporting CICS transactions in the [Administering](#).

### **Solaris** *TXSeries CICS desteği*

IBM MQ for Solaris , XA arabirimini kullanarak TXSeries CICS ' ı destekler.

C ya da COBOL ' de aynı CICS bölgesine yüklenen IBM MQ programlarını yazma. Aynı CICS bölgesine C ve COBOL MQI çağrılarının birleşiminden oluşan bir birleşim yapamazsınız. Most MQI calls in the second language used fail with a reason code of MQRC\_HOBJ\_ERROR.

## Preparing CICS COBOL programs using Micro Focus COBOL

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Micro Focus COBOL olanağını kullanmak için aşağıdaki adımları izleyin:

1. Aşağıdaki komutu kullanarak, IBM MQ COBOL yürütme ortamı kitaplık modülünü yürütme ortamı kitaplığına ekleyin:

```
cicsmkcobol -L/usr/lib/dce -L MQ_INSTALLATION_PATH/lib \  
MQ_INSTALLATION_PATH/lib/libmqmcbirt.o -lmqe
```

**Not:** `cicsmkcobol` ile, IBM MQ , C programlama dilinde COBOL uygulamanızın MQI çağrılarını gerçekleştirmenize izin vermez.

Var olan uygulamalarınızda bu tür çağrılar varsa, bu işlevleri COBOL uygulamalarından kendi kitaplığınıza taşıyın; örneğin, `myMQ.so`. After moving these functions do not include the IBM MQ library `libmqmcbirt.o` when building the COBOL application for CICS.

Ayrıca, COBOL uygulamanızın COBOL MQI çağrısı yapmazsa, `libmqmz_r` ile `cicsmkcobol` bağlantısını bağlamayın.

Bu, Micro Focus COBOL dil yöntemi dosyasını oluşturur ve CICS Runtime COBOL kitaplığını UNIX and Linux sistemlerinde IBM MQ ' u aramasına olanak sağlar.

**Not:** `cicsmkcobol` komutunu yalnızca aşağıdaki ürünlerden birini kurduğunuzda çalıştırın:

- Micro Focus COBOL ' in yeni sürümü veya
- New version or release of TXSeries for Solaris
- Desteklenen herhangi bir veritabanı ürününün yeni sürümü ya da yayın düzeyi (yalnızca COBOL işlemleri için)
- New version or release of IBM MQ

2. Aşağıdaki ortam değişkenini dışa aktarın:

```
COBCPY= MQ_INSTALLATION_PATH/inc export COBCPY
```

3. Programı çevirerek, derleyin ve bağlayın:

```
cicstcl -l COBOL -e yourprog.ccp
```

## CICS C programlarının hazırlanması

Standart CICS olanaklarını kullanarak CICS C programlarını oluşturun:

1. Aşağıdaki ortam değişkenlerinin **bir** ögesini dışa aktarın:
  - `LDFLAGS = "-L MQ_INSTALLATION_PATH ALIND LIB -LMQM_R" LDFLAGS` dışa aktar
  - `USERLIB = "-L MQ_INSTALLATION_PATH add lib -lmqm_r" export USERLIB`
2. Programı çevirerek, derleyin ve bağlayın:

```
cicstcl -l C amqscic0.ccs
```

## CICS C örnek hareketi

Bir CICS IBM MQ işlemi için Örnek C kaynağı, AMQSCIC0.CCS. Hareket, SYSTEM.SAMPLE.CICSiletim kuyruğundan ileti okur.Varsayılan kuyruk yöneticisinde WORKQUEUE ve bunları, iletinin iletim üstbilgisinde yer alan bir kuyruk adıyla yerel kuyruğa yerleştirir. SYSTEM.SAMPLE.CICSkuyruğuna herhangi bir hata gönderilsin.DQ. Örnek MQSC komut kütüğünü ( AMQSCIC0.TST , bu kuyrukları ve örnek giriş kuyruklarını yaratmanızı sağlar.

## Windows Building your procedural application on Windows

Windows sistem yayınları, yazdığınız programlardan yürütülebilir uygulamaların nasıl oluşturulacağını açıklar.

Bu konuda, Windows sistemleri altında çalışmak üzere IBM MQ for Windows uygulamalarını oluştururken gerçekleştirmeniz gereken ek görevleri ve standart görevlerde yapılan değişiklikleri ele alır. ActiveX, C, C++, COBOL ve Visual Basic programlama dilleri desteklenmektedir. ActiveX programlarınızı hazırlamaya ilişkin bilgi için bkz. [Component Object Model Interface \(WebSphere MQ Automation Classes for ActiveX\)](#). C++ programlarınızı hazırlamaya ilişkin bilgi için bkz. [Using C++](#).

IBM MQ for Windows kullanarak yürütülebilir bir uygulama yaratmak için gerçekleştirmeniz gereken görevler, kaynak kodunuzun yazıldığı programlama diline göre değişiklik gösterir. In addition to coding the MQI calls in your source code, you must add the appropriate language statements to include the IBM MQ for Windows include files for the language that you are using. Bu dosyaların içeriğini kendinize tanıdık bir hale getiriniz. Tam açıklama için ["IBM MQ veri tanımlama dosyaları" sayfa 756](#) ' e bakın.

## Windows Windowsüzerinde 64 bit uygulamalar oluşturma

Hem 32 bitlik, hem de 64 bitlik uygulamalar IBM MQ for Windows' ta desteklenir. IBM MQ yürütülür dosyası ve kitaplık dosyaları hem 32 bit, hem de 64 bit biçimlerde sağlanır, çalıştığınız uygulamaya bağlı olarak uygun sürümü kullanın.

## Yürütülür dosyalar ve kitaplıklar

IBM MQ kitaplıklarının 32 bit ve 64 bit sürümleri aşağıdaki konumlarda sağlanır:

Çizelge 155. IBM MQ kitaplıklarının yeri	
Kitaplık sürümü	Kitaplık dosyalarını içeren dizin
32 bit	<code>MQ_INSTALLATION_PATH\Tools\Lib</code>
64 bit	<code>MQ_INSTALLATION_PATH\Tools\Lib64</code>

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

32 bit uygulamalar, geçiş işleminden sonra olağan şekilde çalışmaya devam eder. 32 bitlik dosyalar, ürünün önceki sürümleriyle aynı dizinde bulunur.

If you want to create 64-bit version you must ensure that your environment is configured to use the library files in `MQ_INSTALLATION_PATH\Tools\Lib64`. LIB ortam değişkeninin, 32 bit kitaplıkları içeren klasörü aramak üzere ayarlanmadığından emin olun.

## Windows C programlarını Windowsiçinde hazırlama

Tipik Windows ortamınızda çalışın; IBM MQ for Windows özel bir şey gerektirmez.

64 bit uygulamaların programlanması hakkında daha fazla bilgi için [64 bit platformlarda Coding standartları](#) başlıklı konuya bakın.

- Programlarınızı IBM MQ tarafından sağlanan uygun kitaplıklarla bağlantılayın:

Kitaplık dosyası	Program/çıkış tipi
<i>MQ_INSTALLATION_PATH</i> TH \Tools\Lib\mqm.lib	32 bit C için sunucu
<i>MQ_INSTALLATION_PATH</i> TH \Tools\Lib\mqic.lib	32 bit C için istemci
<i>MQ_INSTALLATION_PATH</i> TH \Tools\Lib\mqicxa.lib	İşlem eşgüdümü olan 32 bitlik C için istemci
<i>MQ_INSTALLATION_PATH</i> TH \Tools\Lib64\mqm.lib	64 bit C için sunucu
<i>MQ_INSTALLATION_PATH</i> TH \Tools\Lib64\mqic.lib	64 bit C için istemci
<i>MQ_INSTALLATION_PATH</i> TH \Tools\Lib64\mqicxa.lib	İşlem eşgüdümü olan 64 bit C için istemci

*MQ\_INSTALLATION\_PATH* , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Aşağıdaki komut, amqsget0 örnek programının ( Microsoft Visual C++ derleyicisi kullanılarak) derlenmesi için bir örnek verir.

32 bit uygulamalar için:

```
c1 -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib\mqm.lib
```

64 bit uygulamalar için:

```
c1 -MD amqsget0.c -Feamqsget.exe MQ_INSTALLATION_PATH\Tools\Lib64\mqm.lib
```

#### Not:

- Kurulabilir bir hizmet yazıyorsanız (ek bilgi için [Administering](#) başlıklı konuya bakın), mqmzf.lib kitaplığına bağlanmanız gerekir.
- IBM TXSeries Encina ya da BEA Tuxedo gibi bir XA uyumlu hareket yöneticisi tarafından dış eşgüdümü için bir uygulama üretiyorsanız, mqmxa.lib ya da mqmxa.lib kitaplığına bağlanmanız gerekir.
- Bir CICS çıkışı yazıyorsanız, mqmcics4.lib kitaplığına bağlantı girin.
- Diğer ürün kitaplıklarından önce IBM MQ kitaplıklarını bağlamanız gerekir.
- DLL ' ler belirttiğiniz yolda (PATH) yer almalıdır.
- Mümkün olduğunda küçük harfli karakterler kullanırsanız, küçük harf kullanımının gerekli olduğu UNIX and Linux sistemlerinde IBM MQ for Windows 'dan IBM MQ ' a hareket edebilirsiniz.

### CICS ve Transaction Server programlarının hazırlanması

Bir CICS IBM MQ işlemi için Örnek C kaynağı, AMQSCIC0.CCS. Bunu, standart CICS olanaklarını kullanarak oluşturursunuz. For example, for TXSeries for Windows 2000:

1. Ortam deęişkenini ayarlayın (bir satıra ařaęıdaki kodu girin):

```
set CICS_IBMC_FLAGS=-I MQ_INSTALLATION_PATH\Tools\C\Include;  
%CICS_IBMC_FLAGS%
```

2. USERLIB ortam deęişkenini ayarlayın:

```
set USERLIB=MQM.LIB;%USERLIB%
```

3. Örnek programı çevirin, derleyin ve baęlayın:

```
cicstcl -l IBMC amqscic0.ccs
```

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduęu üst düzey dizini temsil eder.

Bu, *Transaction Server for Windows NT Application Programming Guide ( CICS ) adlı belgede açıklanmaktadır. V4.*

You can find more information about supporting CICS transactions in the [Administering](#).

### **Windows** *Preparing COBOL programs in Windows*

Use this information to learn to prepare COBOL programs in Windows, and preparing CICS and Transaction Server programs.

1. 32 bit COBOL kopya kitapları řu dizine kurulur: MQ\_INSTALLATION\_PATH \Tools\cobol\CopyBook.
2. 64 bit COBOL kopya kitapları řu dizine kurulur: MQ\_INSTALLATION\_PATH \Tools\cobol\CopyBook64
3. Ařaęıdaki örneklerde CopyBook to:

```
CopyBook
```

32 bit uygulamalar için ve:

```
CopyBook64
```

64 bit uygulamalar için.

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduęu üst düzey dizini temsil eder.

COBOL programlarını Windows sistemlerinde hazırlamak için, programınızı IBM MQ tarafından sağlanan ařaęıdaki kitaplıklardan birine baęlayın:

<b>Kitaplık dosyası</b>	<b>Program ya da çıkış tipi</b>
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

Bir programı MQI istemcisi ortamında çalıştırırken, DOSCALLS kitaplığının COBOL ya da IBM MQ kitaplığından önce görünmesine dikkat edin.

### **COBOL programlarının Micro Focus COBOL**

Var olan 32 bit IBM MQ Micro Focus COBOL programlarını, mqmcb ve mqiccb kitaplıklarını deęil, mqmcb.lib ya da mqiccb.lib kullanarak yeniden baęlayın.

To compile, for example, the sample program amq0put0, using Micro Focus COBOL:

1. COBCPY ortam değişkenini IBM MQ COBOL copybook kitaplarını gösterecek şekilde ayarlayın (bir satıra aşağıdaki kodu girin):

```
set COBCPY= MQ_INSTALLATION_PATH\  
Tools\Cobol\Copybook
```

2. Size bir nesne dosyası vermek için programı derleyin:

```
cobol amq0put0 LITLINK
```

3. Nesne dosyasını çalıştırma zamanı sistemine bağlayın.

- LIB ortam değişkenini, derleyici COBOL kitaplıklarını işaret edecek şekilde ayarlayın.
- Nesne dosyasını IBM MQ sunucusunda kullanılmak üzere bağlayın:

```
cbllink amq0put0.obj mqmcb.lib
```

- Ya da nesne dosyasını IBM MQ istemcisinde kullanılmak üzere bağlantılayın:

```
cbllink amq0put0.obj mqiccb.lib
```

## CICS ve Transaction Server programlarının hazırlanması

To compile and link a TXSeries for Windows NT, V5.1 program using IBM VisualAge COBOL:

1. Ortam değişkenini ayarlayın (bir satıra 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 bağlayın:

```
cicstcl -l IBMCOB myprog.ccp
```

Bu, *Transaction Server for Windows NT, V4 Application Programming Guide* adlı kılavuzda 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  
cbl1link -D -Mcicsmq00 -Ocicsmq00.cbmfmt cicsmq00.obj  
%CICSLIB%\cicsprCBMFNT.lib user32.lib msvcrt.lib kernel32.lib mqmcb.lib
```

## **Preparing Visual Basic programs in Windows**

Windows üzerinde Microsoft Visual Basic programlarını kullanırken dikkate alınması gereken bilgiler.

IBM MQ 9.0' tan Microsoft Visual Basic 6.0 için IBM MQ desteği kullanımdan kaldırılmıştır. IBM MQ classes for .NET are the recommended replacement technology. Daha fazla bilgi için bkz [Developing .NET applications](#).

**Not:** 64-bit versions of the Visual Basic module files are not supplied.

Visual Basic programını Windows üzerinde hazırlamak için:

1. Yeni bir proje yaratır.
2. Sağlanan modül dosyasını ( CMQB.BAS, projeye.
3. Gereksinim duyarsanız, sağlanan diğer modül dosyalarını ekleyin:
  - CMQBB.BAS: MQAI desteği
  - CMQCFB.BAS: PCF desteği
  - CMQXB.BAS: Kanal çıkış desteği
  - CMQPSB.BAS: Yayınla/abone ol

See [“Visual Basic içinde kodlama” sayfa 1087](#) for information about using the MQCONNXAny call from within Visual Basic.

Proje kodunda MQI çağrılarını yapmadan önce MQ\_SETDEFAULTS yordamını çağırın. Bu yordam, MQI çağrılarının gerektirdiği varsayılan yapıları ayarlar.

Specify whether you are creating an IBM MQ server or client, before you compile or run the project, by setting the conditional compilation variable *MqType*. Set *MqType* in a Visual Basic project to 1 for a server or 2 for a client as follows:

1. Proje menüsünü seçin.
2. Select *Name* Properties (where *Name* is the name of the current project).
3. İletişim kutusunda Make (Make) etiketini seçin.
4. Koşullu Derleme Bağımsız Değişkenleri alanında, bu değeri bir sunucu için girin:

```
MqType=1
```

ya da bir istemci için:

```
MqType=2
```

### **İlgili kavramlar**

[“Visual Basic içinde kodlama” sayfa 1087](#)

Microsoft Visual Basic' taki IBM MQ programlarını kodlarken dikkate alınacak bilgiler. Visual Basic yalnızca Windows üzerinde desteklenir.

### **İlgili başvurular**

[“Visual Basic uygulamalarını IBM MQ MQI client koduyla bağlantılandırma” sayfa 952](#)

Microsoft Visual Basic uygulamalarını Windows üzerindeki IBM MQ MQI client kodlarıyla bağlantılayabilirsiniz.



## Windows SSPI güvenlik çıkışı

IBM MQ for Windows , hem IBM MQ MQI client hem de IBM MQ sunucusu için bir güvenlik çıkışı sağlar. Bu program, Security Services Programming Interface (SSPI) olanağını 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'inden yüklenir. Bu DLL 'ler işletim sisteminiz ile birlikte sağlanır.

NTLM kimlik doğrulama hizmetleri kullanılarak tek yönlü kimlik doğrulaması sağlanır. Kerberos kimlik doğrulama hizmetleri kullanılarak iki yönlü kimlik doğrulaması sağlanır.

Güvenlik çıkış programı kaynak ve nesne biçiminde sağlanır. Nesne kodunu olduğu gibi kullanabilir ya da kaynak kodu, kendi kullanıcı çıkış programlarınızı yaratmak için başlangıç noktası olarak kullanabilirsiniz.

Ayrıca bkz. [“Windows üzerindeki SSPI güvenlik çıkışının kullanılması” sayfa 1167.](#)

## Güvenlik çıkışlarına giriş

Bir güvenlik çıkışı, iki güvenlik çıkış programı arasında güvenli bir bağlantı oluşturur; burada bir program, ileti kanalı aracısı (MCA) için bir programdır ve bir program MCA 'yı almak içindir.

Güvenli bağlantıyı başlatan program (örneğin, MCA oturumu kurulduktan sonra denetimi alan ilk program olan) *bağlam başlatıcısı* olarak bilinir. İş ortağı programı, *bağlam kabul edici* olarak bilinir.

Aşağıdaki çizelge, bağlam kullanıma hazırlayıcıları ve ilişkili bağlam kabul edenleri olan bazı kanal tiplerini gösterir.

Çizelge 156. Bağlam başlatıcıları ve bunların ilişkili bağlam kabul edicileri	
Bağlam Başlatıcısı	Bağlam Kabul Edicisi
MQCHT_CLNTCONN	MQCHT_SVRCONN
MQCHT_RECEI	MQCHT_SENDER
MQCHT_CLAUSRCVR	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 doğrulamasına olanak sağlar. İstemcilerin bir sunucunun kimliğini doğrulamasına izin vermez ya da başka bir sunucunun kimliğini doğrulamak için bir sunucu doğrulamaz. NTLM kimlik doğrulaması, sunucuların orijinal olduğu varsayıldığı bir ağ ortamı için tasarlandı.

### • SCY\_KERBEROS

Bu, Kerberos karşılıklı kimlik doğrulama hizmetlerini kullanır. Kerberos protokolü, bir ağ ortamındaki sunucuların gerçek olduğunu varsaymaz. 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 çıkıştan çıkılıyor

Bu konuda, SSPI kanal çıkış programlarının ne yaptığı açıklanmaktadır.

Sağlanan kanal çıkış programları, bir oturum kurulurken bir ortak sistemin tek yönlü ya da iki yönlü (karşılıklı) kimlik doğrulamasını sağlar. Belirli bir kanal için, her çıkış programının ilişkili bir *birincil kullanıcı* (kullanıcı kimliği ile benzer şekilde, bkz. [“IBM MQ erişim denetimi ve Windows asıl adları” sayfa](#)

1058 ) vardır. İki çıkış programı arasında bir bağlantı, iki birincil kullanıcı arasında bir ilişkilendirmeye sahip olur.

Temeldeki oturum oluşturulduktan sonra iki güvenlik çıkış programı arasında güvenli bir bağlantı (MCA 'nın gönderilmesi için biri ve alıcı MCA için bir program) kurulur. İşlem sırası aşağıdaki gibidir:

1. Her program, belirli bir birincil kullanıcıyla ilişkilendirilir; örneğin, açık bir oturum açma işleminin sonucu olarak.
2. Bağlam başlatıcı, güvenlik paketindeki ortakla güvenli bir bağlantı ( Kerberos, adı ortak) ister ve bir simge alır ( token1olarak adlandırılır). Simge, önceden kurulmuş olan temel oturumu ortak programa kullanarak gönderilir.
3. İş ortağı programı (bağlam kabul edici), bağlam başlatıcısının özgün olduğunu doğrulayan token1 ' i güvenlik paketine iletir. NTLM için şu anda bağlantı kurulur.
4. Kerberostarafından sağlanan güvenlik çıkışı (karşılıklı kimlik doğrulama için) için, güvenlik paketi, bağlam kabul cısının temeldeki oturumu kullanarak bağlam başlatıcısına döndürdüğü ikinci bir belirteç ( token2olarak adlandırılır) oluşturur.
5. Bağlam başlatıcısı, bağlam kabul edilenin otantik olduğunu doğrulamak için token2 ' yi kullanır.
6. Bu aşamada, her iki uygulama da iş ortağının simginden memnunsa, güvenli (doğrulanmış) bağlantı kurulur.

## IBM MQ erişim denetimi ve Windows asıl adları

IBM MQ ' in sağladığı erişim denetimi, kullanıcıya ve gruba dayalı olur. Windows ' in sağladığı kimlik doğrulaması, kullanıcı ve servicePrincipalAd (SPN) gibi birincil kullanıcıları temel alır. servicePrincipal(servicePrincipal) adı durumunda, tek bir kullanıcıyla ilişkilendirilmiş olan birçok öge olabilir.

SSPI güvenlik çıkışı, kimlik doğrulaması için ilgili Windows birincil kullanıcılarını kullanır. Windows kimlik doğrulaması başarılı olursa, çıkış, erişim denetimi için Windows asıl adıyla IBM MQ ile ilişkili kullanıcı kimliğini iletir.

Kimlik doğrulaması için ilgili olan Windows asıl adları, kullanılan kimlik doğrulama tipine bağlı olarak değişir.

- For NTLM authentication, the Windows principal for Context Initiator is the user ID associated with the process that is running. Bu kimlik doğrulaması bir yol olduğu için, Context Acceptor ile ilişkili asıl adın ilgisiz olduğu bir yöntem.
- Kerberos kimlik doğrulaması için, CLNTCONN kanallarında, Windows asıl adı, çalışmakta olan süreçle ilişkilendirilmiş olan kullanıcı kimliğidir. Otherwise, the Windows principal is the servicePrincipalName that is formed by adding the following prefix to the QueueManagerName.

```
ibmMQSeries/
```

## Building your procedural application on z/OS

CICS, IMSve z/OS yayınları, bu ortamlarda çalışan uygulamaların nasıl oluşturulacağını açıklar.

Bu konu derlemi, ek görevleri ve standart görevlerde yapılan değişiklikleri, bu ortamlar için IBM MQ for z/OS uygulamaları oluştururken gerçekleştirmeniz gereken değişiklikleri açıklar. COBOL, C, C + +, Assembler ve PL/I programlama dilleri desteklenir. (C++ uygulamaları oluşturma hakkında bilgi için bkz. [Using C++](#).)

Yürütülebilir bir IBM MQ for z/OS uygulaması oluşturmak için gerçekleştirmeniz gereken görevler, hem programın yazıldığı programlama diline, hem de uygulamanın çalışacağı ortama bağlıdır.

Programınızdaki MQI çağrılarını kodlamaya ek olarak, kullanmakta olduğunuz dile ilişkin IBM MQ for z/OS veri tanımlama dosyasını eklemek için uygun dil deyimlerini ekleyin. Bu dosyaların içeriğini kendinize tanıdık bir hale getiriniz. Tam açıklama için [“IBM MQ veri tanımlama dosyaları” sayfa 756 ' e](#) bakın.

## Not

The name **thlqual** is the high-level qualifier of the installation library on z/OS.

### z/OS Programınızı çalıştırmak üzere hazırlama

After you have written the program for your IBM MQ application to create an executable application, you have to compile or assemble it, then link-edit the resulting object code with the stub program that IBM MQ for z/OS supplies for each environment that it supports.

Programınızı nasıl hazırladığınız, uygulamanın çalıştığı ortama (toplu iş, CICS, IMS(BMP ya da MPP), Linux ya da UNIX System hizmetlerine) ve z/OS kurulumunuzda veri kümelerinin yapılarına bağlıdır.

“IBM MQ sınırlı kod öbeğini devingen olarak çağırma” sayfa 1065 , bir IBM MQ sınırlı kod öbeğini bağlanmanıza gerek kalmaması için, programlarınızda MQI çağrılarının yapılması için alternatif bir yöntem tanımlıyor. Bu yöntem, tüm diller ve ortamlar için kullanılamaz.

Daha yüksek bir sınırlı kod öbeği programını, programınızın çalıştığı IBM MQ for z/OS sürümünden bu sürümden bağlamayın. Örneğin, OS/390 için MQSeries üzerinde çalışan bir program olan V5.2 , IBM MQ for z/OS V7 ile birlikte verilen bir sınırlı kod öbeği programıyla bağlantı düzenlenmemelidir.

### z/OS Bina 64 bit C uygulamaları

z/OS' ta, 64 bit C uygulamaları, LP64 derleyicisi ve bağ tanımlama seçenekleri kullanılarak oluşturulur. IBM MQ for z/OS *cmqc.h* üstbilgi dosyası, derleyiciye bu seçenek sağlandığında verilir ve 64 bit işlemi için uygun IBM MQ veri tiplerini ve yapıları oluşturur.

Bu seçenekle oluşturulan C kodu, gerekli koordinasyon semantik için uygun dinamik bağlantı kitaplıklarını (DLL ' ler) kullanmak üzere oluşturulmalıdır. Derlenmiş kodu, [Her eşgüdümleme semantik için gerekli yan deste adı](#) içinde tanımlanan uygun yüz deste ile bağlamanın, gerekli olan DLL ' in gösterilmesine neden olduğunu gösterir.

Çizelge 157. Her eşgüdümleme semantik için gerekli yan deste adı	
coordination	Yan güverte adı
Tek aşamalı kesinleştirme MQI	CSQBMQ2X
RRS eşgüdümü ile iki aşamalı kesinleştirme, RRS fillerini kullanma	CSQBRR2X
RSC eşgüdümü ile iki aşamalı kesinleştirme, MQI fillerinin kullanılması	CSQBRI2X

Toplu iş olarak tek aşamalı kesinleştirme IBM MQ programı oluşturmak için z/OS XL C/C++ ile birlikte verilen EDCQCB JCL yordamını kullanın:

```
//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 Services olanağında bir RRS eşgüdümlü programı oluşturmak için, aşağıdaki gibi derleme 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 toplu iş uygulamalarının nasıl oluşturulacağı ve bunu yaparken göz önünde bulundurulacak adımların nasıl oluşturulacağı hakkında bilgi edinin.

z/OS toplu işi altında çalışan IBM MQ for z/OS için bir uygulama oluşturmak için, bu görevleri gerçekleştiren iş denetim dili (JCL) oluşturun:

1. Nesne kodunu üretmek için programı derleyin (ya da birleştirin). Derleyiciye ilişkin JCL, derleyicinin kullanabileceği ürün verileri tanımlama dosyalarını oluşturan SYSLIB deyimlerini içermelidir. Veri tanımları 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. C uygulaması için, “1” sayfa 1060adımında yaratılan nesne kodunu ön bağlantıyla ilişkilendirin.
3. PL/I uygulamaları için, EXTRN (SHORT) derleyici seçeneğini kullanın.
4. Link-edit the object code created in step “1” sayfa 1060 (or step “2” sayfa 1060 for a C application) to produce a load module. Kodu bağlarken, kodu düzenlediğinizde, IBM MQ for z/OS toplu sınırlı kod parçası programlarından birini (CSQBKOD ya da RRS kod parçası programlarından biri: CSQBRRSI ya da CSQBRSTB) eklemelisiniz.

#### **CSQBKOD**

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, ATRSCSS ile uygulamanızı SYS1.CSSLIB' den ATRSCSS ile de düzenlemeniz gerekir. Şekil 118 sayfa 1060 and Şekil 119 sayfa 1061 show fragments of JCL to do this. Sınırlı kod öbekleri dilden bağımsızdır ve **thlqual.SCSQLOAD** kitaplığı olarak sağlanır.
  - b. Uygulamanız Dil Ortamı altında çalışıyorsa, “Building z/OS batch applications using Language Environment” sayfa 1061 içinde açıklandığı şekilde, Language Environment DLL ile bağlantı düzenlediğinizden emin olmalısınız.
5. Yükleme modülünü bir uygulama yükleme kitaplığında saklayın.

```
..
/*
/* WEBSHERE MQ FOR Z/OS LIBRARY CONTAINING BATCH STUB
/*
/*CSQSTUB DD DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
/*
..
//SYSIN DD *
INCLUDE CSQSTUB(CSQBSTUB)
..
/*
```

Şekil 118. Toplu iş ortamında nesne modülünü düzenlemek için JCL parçaları, tek aşamalı kesinleştirmeyi kullanarak nesne modülünü değiştirin

```

:
/*
/** WEBSPPHERE 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 119. İki aşamalı kesinleştirme kullanılarak, nesne modülünü toplu iş ortamında bağlamak için JCL parçaları parçaları

Bir toplu ya da RRS programını çalıştırmak için, STEPLIB ya da JOBLIB veri kümesi bitişirmesine ilişkin **thlqual.SCSQAUTH** ve **thlqual.SCSQLOAD** kitaplıklarını eklemelisiniz.

TSO programını çalıştırmak için, TSO oturumu tarafından kullanılan STEPLIB içinde **thlqual.SCSQAUTH** ve **thlqual.SCSQLOAD** kitaplıklarını eklemelisiniz.

To run a UNIX System Services batch program from the UNIX System Services shell, add the libraries **thlqual.SCSQAUTH** and **thlqual.SCSQLOAD** to the STEPLIB specification in your \$HOME?.profile like this:

```

STEPLIB= thlqual.SCSQAUTH: thlqual.SCSQLOAD
export STEPLIB

```

#### z/OS Building z/OS batch applications using Language Environment

IBM MQ for z/OS , uygulamalarınızı bağlarken kullanılması gereken bir dizi dinamik bağlantı kitaplığı (DLL) sağlar.

Uygulamanın aşağıdaki çağırın arabirimlerden birini kullanmasına izin veren iki kitaplık değişkeni vardır:

- 31 bitlik Dil Ortamı arabirimi çağırılıyor.
- 31 bit XPLINK arama arabirimi. z/OS XPLINK, C uygulamaları için kullanılabilen yüksek performanslı bir arama sözleşmesidir.

DDL ' leri kullanmak için uygulama, önceki sürümlerle sağlanan sınırlı kod öbekleri yerine, *sidedestead* verilen şekilde bağlanır ya da bağlanır. Kenar desteleri SCSQDEFS kitaplığında bulunur (SCSQLOAD kitaplığı yerine).

Çizelge 158. Dinamik bağlantı kitaplıklarının çeşitlemeleri			
Kesinleştir	31 bit Dil Ortamı DLL 'si	31 bit XPLINK DLL	Eşdeğer kod parçası adı
1 evre kesinleştirme MQI kitaplıkları	CSQBMQ1	CSQBMQ1X	CSQBKOD
RRS işlem-denetim fiilleri kullanılarak RRS eşgüdümü ile 2 aşamalı kesinleştirme	CSQBRR1	CSQBRR1X	CSQBRSTB
MQI hareket-denetim fiillerini kullanarak RRS eşgüdümü ile 2 aşamalı kesinleştirme	CSQBRI1	CSQBRI1X	CSQBRRSI

**Not:** Tüm yardımcı birimler, daha önce CSQASKOR da içinde olmak üzere, veri dönüştürme giriş noktasının, MQXCNVC ' nin bir tanımını içerir.

Ortak sorunlar:

- Uygulamanızın zamanuyumsuz ileti tüketmesi (MQCB, MQCTL ya da MQSUB çağrıları) kullanılıyorsa ve önceki DLL arabirimi kullanılmıyorsa, iş günlüğünde aşağıdaki ileti görüntülenir:

CSQB001E Language environment programs running in z/OS batch or USS must use the DLL interface to IBM MQ

Çözüm: Daha önce ayrıntılı olarak sınırlı kod öbekleri yerine kaldırımları kullanarak uygulamanızı yeniden oluşturun.

- Program oluşturma sırasında aşağıdaki ileti görüntülenir:

IEW2469E *kodunuz* bölümünden MQAPI-NAME ögesine yönelik başvuruların öznitelikleri, bu özniteliklerin öznitelikleriyle eşleşmiyor.  
hedef simge

Reason: This means that you have compiled your XPLINK program with V701 (or later) version of cmqc.h, but are not binding with sidedecks.

Çözüm: Programınızın oluşturma dosyasını, SCSQLOAD ' dan bir sınırlı kod öbeği yerine, uygun yardımcı güverteye (SCSQDEFS) bağlamak için değiştirin.

Aşağıdaki örnek JCL, bir C programını, 31 bit Dil Ortamı DLL arama arabirimini kullanmak için nasıl derleyebileceğinin ve nasıl düzenleyebileceğinin gösterilmiştir:

```
//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=h1q.SCSQC370
//COMPILE.SYSLIBV DD DISP=SHR,DSN=h1q.BASE.H
/*
//BIND.SYSOBJ DD DISP=SHR,DSN=CEE.SCEE0BJ
// DD DISP=SHR,DSN=h1q.SCSQDEFS
//BIND.SYSLMOD DD DISP=SHR,DSN=h1q.LOAD(MYPROGAM)
//BIND.SYSIN DD *
ENTRY CEESTART
INCLUDE SYSOBJ(CSQBMQ1)
NAME MYPROGAM(R)
//
```

**Not:** Derleme, **DLL** seçeneğini kullanır. Bağlantı düzenleme işlevi, **DYNAM=DLL** seçeneği ve **CSQBMQ1** kitaplığının başvurularını kullanır.

Aşağıdaki örnek JCL, 31 bit XPLINK DLL arama arabirimini kullanmak için bir C programını nasıl derleyebileceğinden ve nasıl düzenleyebileceğinin gösterilmiştir:

```
//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=h1q.SCSQC370
//COMPILE.SYSLIBV DD DISP=SHR,DSN=h1q.BASE.H
/*
//BIND.SYSOBJ DD DISP=SHR,DSN=CEE.SCEE0BJ
// DD DISP=SHR,DSN=h1q.SCSQDEFS
//BIND.SYSLMOD DD DISP=SHR,DSN=h1q.LOAD(MYPROGAM)
//BIND.SYSIN DD *
ENTRY CEESTART
INCLUDE SYSOBJ(CSQBMQ1X)
NAME MYPROGAM(R)
//
```

**Not:** Derleme, **XPLINK** ve **DLL** seçeneklerini kullanır. Bağlantı düzenleme işlemi **DYNAM=DLL** seçeneğini kullanır ve **CSQBMQ1X** kitaplığına gönderme yapar.

Derleme seçeneği DLL 'sini modüldeki her bir programa eklediğinizden emin olun. IEW2456E 9207 SIMGESI CSQ1BAK SYNRESH gibi iletiler, tüm programların DLL seçeneğiyle derlenmiş olduğunu denetlemeniz gerektiğini gösteren bir gösterebilir.

## z/OS z/OSiçinde CICS uygulamaları oluşturma

z/OS' ta CICS uygulamaları oluşturulurken bu bilgileri kullanın.

CICSaltında çalışan IBM MQ for z/OS için bir uygulama oluşturmak için aşağıdakileri yapmak gerekir:

- Programınızdaki CICS komutlarını, programınızın geri kalanının yazıldığı dilde çevirmesini sağlar.
- Nesne kodunu üretmek için çevirmenden çıkışı derleyin ya da birleştirin.
  - PL/I programlarında, EXTRN (SHORT) derleyici seçeneğini kullanın.
  - C uygulamaları için, uygulama XPLINK kullanmıyorsa, derleyici seçeneğini DEFINE kullanın (MQ\_OS\_LINKAGE=1).
- Bağlantı-bir yükleme modülü yaratmak için nesne kodunu düzenleyin.

CICS , desteklediği programlama dillerinin her biri için bu adımları sırayla yürütmek için bir yordam sağlar.

- For CICS Transaction Server for z/OS, the *CICS Transaction Server for z/OS System Definition Guide* describes how to use these procedures and the *CICS/ESA Application Programming Guide* gives more information on the translation process.

Şunları eklemelisiniz:

- Derleme (ya da derleme) aşamasındaki SYSLIB deyiminde, ürün verileri tanımlama dosyalarını derleyici tarafından kullanılabilir duruma getirebilecek deyimler. Veri tanımları 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**
- Bağlantı düzenleme JCL ' nizde, IBM MQ for z/OS CICS sınırlı kod öbeği programı (CSQCKOD). [Şekil 120 sayfa 1063](#) , bu işlemi yapmak için JCL kodunun parçalarını gösterir. Sınırlı kod öbeği dilden bağımsızdır ve **thlqual.SCSQLOAD** kitaplığı olarak sağlanır.

```

:
/*
/* WEBSPPHERE MQ FOR Z/OS LIBRARY CONTAINING CICS STUB
/*
/*CSQSTUB DD DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
/*
:
//LKED.SYSIN DD *
INCLUDE CSQSTUB(CSQSTUB)
:
/*
```

Şekil 120. Fragments of JCL to link-edit the object module in the CICS environment

- For CICS versions later than CICS TS 3.2, or, if you want to use IBM MQ message property APIs, or IBM MQ APIs MQCB, MQCTL, MQSTAT, MQSUB or MQSUBR, you must linkedit your object code with the CICS supplied stub, DFHMOSTB and not the IBM MQ supplied CSQSTUB. CICS için IBM MQ programlarının oluşturulması hakkında daha fazla bilgi için CICS ürün belgelerindeki [IBM MQ MQI çağrılarında erişmek için API kod parçası programı](#) belgesine bakın.

Bu adımları tamamladığınızda, yükleme modülünü bir uygulama yükleme kitaplığında saklayın ve programı olağan şekilde CICS ' e tanımlayın.

Bir CICS programını çalıştırmadan önce, sistem denetimcinizin bunu IBM MQ programı ve işlemi olarak CICS olarak tanımlamalı ve bunu tipik bir şekilde çalıştırabilmeniz gerekir.

## **z/OS** Building IMS (BMP or MPP) applications

IMS (BMP ya da MPP) uygulamalarını oluştururken bu bilgileri kullanın.

Toplu DL/I programları oluşturuyorsanız, bkz. [“z/OS toplu iş uygulamaları oluşturma” sayfa 1060](#). IMS altında çalışan (BMP ya da MPP olarak) başka uygulamalar oluşturmak için, bu görevleri gerçekleştiren JCL oluşturun:

1. Nesne kodunu üretmek için programı derleyin (ya da birleştirin). Derleyiciye ilişkin JCL, derleyicinin kullanabileceği ürün verileri tanımlama dosyalarını oluşturan SYSLIB deyimlerini içermelidir. Veri tanımları 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. C uygulaması için, [“1” sayfa 1064](#) adımıyla yaratılan nesne modülünü ön bağlantıyla ilişkilendirin.
3. PL/I programlarında, EXTRN (SHORT) derleyici seçeneğini kullanın.
4. C uygulaması için, uygulama XPLINK kullanmıyorsa, derleyici seçeneğini DEFINE kullanın (MQ\_OS\_LINKAGE=1).
5. Yükleme modülü üretmek için, [“1” sayfa 1064](#) adımıyla (ya da C/370 uygulaması için [“2” sayfa 1064](#) adımıyla) oluşturulan nesne kodunu düzenleyin:
  - a. IMS dil arabirimi modülünü ekleyin (DFSLI000).
  - b. IBM MQ for z/OS IMS sınırlı kod öbeği programını (CSQQSTUB) ekleyin. [Şekil 121 sayfa 1064](#), bu işlemi yapmak için JCL 'nin parçalarını gösterir. Sınırlı kod öbeği dili bağımsızdır ve **thlqual.SCSQLOAD** kitaplığında sağlanır.

**Not:** If you are using COBOL, select the NODYNAM compiler option to enable the linkage editor to resolve references to CSQQSTUB unless you intend to use dynamic linking as described in [“IBM MQ sınırlı kod öbeğini devingen olarak çağırma” sayfa 1065](#).
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(CSQSTUB)
..
/*
```

Şekil 121. Fragments of JCL to link-edit the object module in the IMS environment

Bir IMS programını çalıştırmadan önce, sistem denetimcinizin bunu bir IBM MQ programı ve işlem olarak IMS olarak tanımlamalı olması gerekir: daha sonra, bunu tipik bir şekilde çalıştırabilirsiniz.

## **z/OS** Building z/OS UNIX System Services applications

z/OS UNIX System Services uygulamalarını oluştururken bu bilgileri kullanın.

To build a C application for IBM MQ for z/OS that runs under UNIX System Services, compile and link your application as follows:

```
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, .profile dosyanızı aşağıdaki dizine eklemeniz gerekir; bu, kök dizininizde olmalıdır:

```
STEPLIB= thlqual.SCSQANLE:thlqual.SCSQAUTH: STEPLIB
```

Note that you need to exit from UNIX System Services, and enter UNIX System Services again, for the change to be recognized.

Birden çok kabuk çalıştırmak istiyorsanız, satır başına sözcük dışı aktarma sözcüğünü ekleyin. Bu değer:

```
export STEPLIB= thlqual.SCSQANLE:thlqual.SCSQAUTH: STEPLIB
```

Bu işlem başarıyla tamamlandığında CSQBSTUB ' i bağlayabilir ve IBM MQ çağrılarını yayınlayabilirsiniz.

“IBM MQ sınırlı kod öbeğini devingen olarak çağırma” sayfa 1065 , bir IBM MQ sınırlı kod öbeğini bağlanmanıza gerek kalmaması için, programlarınızda MQI çağrılarının yapılması için alternatif bir yöntem tanımlıyor. Bu yöntem, tüm diller ve ortamlar için kullanılamaz.

Daha yüksek bir sınırlı kod öbeği programını, programınızın çalıştığı IBM MQ for z/OS sürümünden bu sürümden bağlamayın. Örneğin, IBM WebSphere MQ for z/OS 7.1 üzerinde çalışan bir program, IBM MQ for z/OS 8.0 ile birlikte verilen bir sınırlı kod öbeği programıyla bağlantı düzenlenmemelidir.

### **z/OS IBM MQ sınırlı kod öbeğini devingen olarak çağırma**

IBM MQ sınırlı kod öbeği programını, nesne kodunuzla bağlamak yerine, sınırlı kod öbeğini programınızın içinden dinamik olarak çağırabilirsiniz.

Bunu toplu iş, IMS ve CICS ortamlarında yapabilirsiniz. Bu olanak, RRS ortamında desteklenmez. Uygulama programınız güncellemeleri koordine etmek için RRS kullanıyorsa, bkz. “RRS Dikkate Alınması” sayfa 1069.

Ancak, bu yöntem:

- Programlarınızın karmaşıklığını artırır
- Yürütme sırasında programlarınızın gerektirdiği depolama alanını artırır
- Programlarınızın performansını azaltır
- Diğer ortamlarda aynı programları kullanamayamazsınız.

Sınırlı kod öbeğini devingen olarak çağırırsanız, uygun kod öbeği programı ve diğer adları yürütme sırasında kullanılabilir olmalıdır. Bunu sağlamak için, IBM MQ for z/OS veri kümesi SCSQLOAD ' u ekleyin:

- Toplu iş ve IMS için, JCL ' nin STEPLIB bitişirme işlemi.
- CICS için, CICS DFHRPL birleştirmesinde.

For IMS, ensure that the library containing the dynamic stub (built as described in the information about installing the IMS adapter in [IMS bağdaştırıcısının ayarlanması](#)) is ahead of the data set SCSQLOAD in the STEPLIB concatenation of the region JCL.

Sınırlı kod öbeğini devingen olarak çağırduğunuzda, Çizelge 159 sayfa 1065 içinde gösterilen adları kullanın. PL/I içinde, yalnızca programınızda kullanılan arama adlarını bildirebilirsiniz.

<i>Çizelge 159. Dinamik bağlantı oluşturma için arama adları</i>			
<b>MQI çağrısı</b>	<b>Toplu iş (RRS olmayan) dinamik arama adları</b>	<b>CICS dinamik arama adları</b>	<b>IMS dinamik arama adları</b>
<b>MQBACK</b>	CSQBBACK	desteklenmiyor	Desteklenmiyor
<b>MQBUFMH</b>	CSQBFBMH	CSQCBFMH <sup>1</sup>	MQBUFMH
<b>MQCB</b>	CSQBBB	CSQCCB <sup>1</sup>	Desteklenmiyor
<b>MQCLOSE</b>	CSQBCLOS	CSQCCLOS	MQCLOSE

Çizelge 159. Dinamik bağlantı oluşturma için arama adları (devamı var)

<b>MQI çağırısı</b>	<b>Toplu iş (RRS olmayan) dinamik arama adları</b>	<b>CICS dinamik arama adları</b>	<b>IMS dinamik arama adları</b>
<b>MQCMIT</b>	CSQBCOMM	desteklenmiyor	Desteklenmiyor
<b>MQCONN</b>	CSQBCONN	CSQCCONN	MQCONN
<b>MQCONNX</b>	CSQBCONX	CSQCCONX	MQCONNX
<b>MQCRTMH</b>	CSQBCTMH	CSQCCTMH <sup>1</sup>	MQCRTMH
<b>MQCTL</b>	CSQBCTL	CSQCCTL <sup>1</sup>	Desteklenmiyor
<b>MQDISC</b>	CSQBISC	CSQCDISC	MQDISC
<b>MQDLTMH</b>	CSQBDMH	CSQCDTMH <sup>1</sup>	MQDLTMH
<b>MQDLTMP</b>	CSQBDMPT	CSQCDTMP <sup>1</sup>	MQDLTMP
<b>MQGet</b>	CSQBGET	CSQCGET	MQGet
<b>MQINQ</b>	CSQBINQ	CSQCINQ	MQINQ
<b>MQINQMP</b>	CSQBIMPT	CSQCIQMP <sup>1</sup>	MQINQMP
<b>MQMHBUF</b>	CSQBMHBF	CSQCMHBF <sup>1</sup>	MQMHBUF
<b>MQOPEN</b>	CSQBOPEN	CSQCOPEN	MQOPEN
<b>MQPUT</b>	CSQBPUT	CSQCPUT	MQPUT
<b>MQPUT1</b>	CSQBPUT1	CSQCPUT1	MQPUT1
<b>MQSET</b>	CSQBSET	CSQCSET	MQSET
<b>MQSETMP</b>	CSQBSTMP	CSQCSTMP <sup>1</sup>	MQSETMP
<b>MQSTAT</b>	CSQBSTAT	CSQCSTAT <sup>1</sup>	MQSTAT
<b>MQSUB</b>	CSQB SUB	CSQCSUB <sup>1</sup>	MQSUB
<b>MQSUBRQ</b>	CSQB SUBR	CSQCSUBR <sup>1</sup>	MQSUBRQ

**Not:** 1. Bu API çağrıları yalnızca CICS TS 3.2 ya da sonraki bir sürümü kullanıldığında ve CICS ile birlikte gönderilen CSQCSTUB kullanılmalı olduğunda kullanılabilir. CICS TS 3.2 için APAR PK66866 uygulanmalıdır. CICS TS 4.1 için APAR PK89844 uygulanmalıdır.

Bu tekniğin nasıl kullanılacağını gösteren örnekler için aşağıdaki şekillere bakın:

- Toplu ve COBOL: bkz. [Şekil 122 sayfa 1067](#)
- CICS ve COBOL: bkz. [Şekil 123 sayfa 1067](#)
- IMS ve COBOL: bkz. [Şekil 124 sayfa 1067](#)
- Toplu iş ve çevirici: bkz. [Şekil 125 sayfa 1068](#)
- CICS ve çevirici: bkz. [Şekil 126 sayfa 1068](#)
- IMS ve çevirici: bkz. [Şekil 127 sayfa 1068](#)
- Toplu İş ve C: [Şekil 128 sayfa 1068](#)
- CICS ve C: bkz. [Şekil 129 sayfa 1068](#)
- IMS ve C: bkz. [Şekil 130 sayfa 1069](#)
- Toplu İş ve PL/I: bkz. [Şekil 131 sayfa 1069](#)
- IMS ve PL/I: bkz. [Şekil 132 sayfa 1069](#)

```

...
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 122. COBOL ile toplu iş ortamında 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 123. COBOL kullanarak CICS ortamında 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 124. COBOL kullanarak IMS ortamında dinamik bağlantı oluşturma

```

...      LOAD    EP=CSQBOPEN
...      CALL   (15), (HCONN, MQOD, OPTIONS, HOBJ, COMPCODE, REASON), VL
...      DELETE EP=CSQBOPEN
...

```

*Şekil 125. Toplu iş ortamında montaj 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 126. Dynamic linking using assembly language in the CICS environment*

```

...      LOAD    EP=MQOPEN
...      CALL   (15), (HCONN, MQOD, OPTIONS, HOBJ, COMPCODE, REASON), VL
...      DELETE EP=MQOPEN
...

```

*Şekil 127. Dynamic linking using assembly language in the IMS environment*

```

...
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 128. 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 129. 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 130. 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 131. Toplu iş ortamında PL/I kullanarak dinamik bağlantı oluşturma

```

...
DCL MQOPEN ENTRY EXT OPTIONS(ASSEMBLER INTER);
...
FETCH MQOPEN;

CALL MQOPEN(HQM,
            MQOD,
            OPTIONS,
            HOBJ,
            COMPCODE,
            REASON);

RELEASE MQOPEN;

```

Şekil 132. IMS ortamında PL/I kullanarak dinamik bağlantı oluşturma

#### RRS Dikkate Alınması

Uygulama programınız güncelleştirmeleri koordine etmek için RRS 'yi kullanıyorsa bu bilgileri kullanın.

IBM MQ , RRS koordinasyonuna gereksinim duyan toplu iş programları için iki farklı sınırlı kod öbeği sağlar- "[RRS toplu iş bağdaştırıcısı](#)" sayfa 925. Daha sonraki API çağrılarının davranışındaki fark, toplu iş bağdaştırıcısı tarafından MQCONN ya da MQCONNX API 'deki sınırlı kod öbeği yordamında geçirilen bilgilerden, MQCONN' de belirlenir. This means that dynamic API calls are available for batch programs which need RRS coordination, provided that the initial connection to IBM MQ was done by using the appropriate stub. 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 CSQBRSI 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 ayıklanması**

Bu bilgileri, hata ayıklama TSO ve CICS programlarıyla ve CICS izleme ile ilgili bir kavrayış hakkında bilgi edinmek için kullanın.

IBM MQ for z/OS uygulama programlarında hata ayıklamaya yardımcı olan ana yardımlar, her API çağırısı tarafından döndürülen neden kodlarıdır. Düzeltme işlemine ilişkin fikirler 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 [Messages and reason codes](#)

Bu konuda ayrıca, belirli ortamlarda kullanılmak üzere diğer hata ayıklama araçları da önerilmektedir.

## **TSO programlarında hata ayıklama**

TSO programları için aşağıdaki etkileşimli hata ayıklama araçları kullanılabilir:

- TEST aracı
- VS COBOL II etkileşimli hata ayıklama aracı
- C ve PL/I programları için etkileşimli hata ayıklama aracını inceleyin

## **CICS programlarında hata ayıklama**

You can use the CICS Execution Diagnostic Facility (CEDF) to test your CICS programs interactively without having to modify the program or program-preparation procedure.

EDF hakkında daha fazla bilgi için *CICS Transaction Server for z/OS CICS Application Programming Guide* adlı belgeye bakın.

## **CICS İz**

Ayrıca, CICS izleme etkinliğini denetlemek için CICS İzleme Denetimi hareketinin (CEETR) kullanılmasının yararlı olacağını da bulacaksınız.

CEETR hakkında daha fazla bilgi için bkz. *CICS Transaction Server for z/OS CICS-Supplied Transactions* el ile.

CICS izleme işleminin etkin olup olmadığını belirlemek için, CKQC panosunu kullanarak bağlantı durumunu görüntüleyin. Bu pano, izleme numarasını da gösterir.

CICS izleme girdilerini yorumlamak için bkz. [Çizelge 160 sayfa 1071](#).

Bu değerlere ilişkin CICS izleme girişi AP0 xxx biçimidir (burada xxx, CICS bağdaştırıcısı etkinleştirildiğinde belirtilen izleme numarasıdır). CSQCTEST dışındaki tüm izleme girişleri CSQCTRUE tarafından verilir. CSQCTEST, CSQCRST ve CSQCDSR tarafından verilir.

Çizelge 160. CICS bağdaştırıcı izleme girişleri

Ad	Tanım	İzleme sırası	İzleme verileri
CSQCABNT	Olağandışı sona erdirme	Before issuing END_THREAD ABNORMAL to IBM MQ. Bunun nedeni, görevin sona ermesinden kaynaklanır ve uygulama tarafından örtük bir geri alma gerçekleştirilebilecektir. Bu durumda, END_THREAD çağrısına bir ROLLBACK isteği dahil edilir.	İş birimi bilgileri. Bu bilgileri, işin durumu hakkında bilgi bulunca kullanabilirsiniz. (Örneğin, DISPLAY THREAD komutu ya da IBM MQ for z/OS günlük yazdırma yardımcı programı tarafından üretilen çıkışa göre doğrulanabilir.)
CSQCBACK	Syncpoint geri alma	BACKUT 'u IBM MQ for z/OS' a vermeden önce. Bunun nedeni, uygulamadaki belirtik bir geri alma isteğidir.	İş birimi bilgileri.
CSQCCRC	Tamamlanma kodu ve neden kodu	API çağrısından başarısız döndükten sonra.	Tamamlanma kodu ve neden kodu.
CSQCCOMM	Syncpoint kesinleştirme	Before issuing COMMIT to IBM MQ for z/OS. Bunun nedeni, tek aşamalı kesinleştirme isteğinden ya da iki aşamalı kesinleştirme isteğinin ikinci evresinden kaynaklanabilir. İstek, uygulamadaki belirtik bir eşitleme noktası isteğinden kaynaklanır.	İş birimi bilgileri.
CSQCEXER	Çözümleme yürüt	EXECUTE_RESOLVE IBM MQ for z/OS' u yayınlamadan önce.	EXECUTE_RESOLVE birimini veren iş biriminin iş bilgileri birimi. Bu, yeniden eşzamanlama sürecindeki en son belirsiz iş birimidir.
CSQCGETW	-Bekle.	CICS beklemeden önce bekleyin.	Beklenecek ECB ' nin adresi.
CSQCGMGD	GET iletisi verileri	MQGET ' den başarıyla döndükten sonra.	İleti verisinin en çok 40 baytı.
CSQCGMGH	GET ileti tanıtıcısı	MQGET 'yi IBM MQ for z/OS' e yayınlamadan önce.	Nesne tanıtıcısı.
CSQCGMGI	İleti tanıtıcısını al	MQGET ' den başarıyla döndükten sonra.	İletinin ileti tanıtıcısı ve ilinti tanıtıcısı.
CSQCINDL	Belirsiz liste	İkinci INQUIRE_INDOUBT ögesinden başarıyla döndükten sonra.	Belirsiz iş birimleri listesi.
CSQCINDO	YalnızcaIBM kullanın		
CSQCINDS	Belirsiz liste boyutu	İlk INQUIRE_INDOUBT ve belirsiz listeden başarıyla döndükten sonra boş değil.	Listenin uzunluğu. 64 'e bölünen, belirsiz iş birimlerinin sayısını gösterir.
CSQCINQH	INQ tutamacı	MQINQ 'yi IBM MQ for z/OS' e yayınlamadan önce.	Nesne tanıtıcısı.

Çizelge 160. CICS bağdaştırıcı izleme girişleri (devamı var)

Ad	Tanım	İzleme sırası	İzleme verileri
CSQCLOSH	KAPALI KAL	MQCLOSE 'yi IBM MQ for z/OS' e vermeden önce.	Nesne tanıtıcısı.
CSQCLOST	Yok etme kaybı	Yeniden eşzamanlama işlemi sırasında, CICS yeniden başlatılmakta olan iş birimine ilişkin herhangi bir yok etme bilgisi olmadığından, bağdaştırıcıya yeniden başlatıldığını bildirir.	Yeniden eşzamanlanmakta olan iş birimi için CICS olarak bilinen iş birimi tanıtıcısı.
CSQCNIND	Yok etme belirsiz	Yeniden eşzamanlama işlemi sırasında, CICS , bağdaştırıcıya, yeniden eşzamanlanmakta olan iş biriminin belirsiz kalmaması gerektiğini bildirir (yani, hala çalışır durumda olduğunu).	Yeniden eşzamanlanmakta olan iş birimi için CICS olarak bilinen iş birimi tanıtıcısı.
CSQCNORT	Olağan sonlandırma	END_THREAD NORMAL IBM MQ for z/OS' ı yayınlamadan önce. Bunun nedeni görevin sona ermesi ve uygulamanın örtük bir eşitleme noktası kesinleştirmesi gerçekleştirmesine neden olabilir. Bu durumda END_THREAD çağrısına bir COMMIT isteği dahil edilir.	İş birimi bilgileri.
CSQCOPNH	OPEN TAN	MQOPER ' tan başarıyla döndükten sonra.	Nesne tanıtıcısı.
CSQCOPNO	Nesneyi Aç	Before issuing MQOPEN to IBM MQ for z/OS.	Nesne adı.
CSQCPMGD	GUT iletisi verileri	MQPUT ' i IBM MQ for z/ OSolarak yayınlamadan önce.	İleti verisinin en çok 40 baytı.
CSQCPMGH	PUT ileti tanıtıcısı	MQPUT ' i IBM MQ for z/ OSolarak yayınlamadan önce.	Nesne tanıtıcısı.
CSQCPMGI	PUT ileti tanıtıcısı	IBM MQ for z/OS' tan başarıyla MQPUT işlemi tamamlandıktan sonra.	İletinin ileti tanıtıcısı ve ilinti tanıtıcısı.
CSQCPREP	Eşitleme noktası hazırlama	İki aşamalı kesinleştirme işleminin birinci aşamasında PREPARE IBM MQ for z/OS ' u yayınlamadan önce. Bu çağrı, dağıtılmış kuyruğa alma bileşeninden bir API çağrısı olarak da verilebilir.	İş birimi bilgileri.
CSQCP1MD	PUTONE ileti verileri	MQPUT1 ögesini IBM MQ for z/ OSolarak yayınlamadan önce.	İletinin 40 bayt 'a kadar veri verisi.
CSQCP1MI	PUTONE ileti tanıtıcısı	MQPUT1' tan başarıyla döndükten sonra.	İletinin ileti tanıtıcısı ve ilinti tanıtıcısı.



Çizelge 160. CICS bağdaştırıcı izleme girişleri (devamı var)			
Ad	Tanım	İzleme sırası	İzleme verileri
CSQCP1ON	PUTONE nesne adı	MQPUT1 ögesini IBM MQ for z/OSolarak yayınlamadan önce.	Nesne adı.
CSQCRBAK	Çözömlenen geri alma	Before issuing RESOLVE_ROLLBACK to IBM MQ for z/OS.	İş birimi bilgileri.
CSQCRMT	Çözölmüş kesinleştirme	Before issuing RESOLVE_COMMIT to IBM MQ for z/OS.	İş birimi bilgileri.
CSQCRMIR	RMI yanıtı	CICS RMI ' ye (kaynak yöneticisi arabirimi) belirli bir çağrıdan dönmeden önce.	Archipted RMI yanıt değeri. Anlamı, çağrının tipine bağlıdır. Bu değeler, <i>CICS Transaction Server for z/OS Customization Guide</i> adlı belgede belgelenir. Çağırma tipini saptamak için, CICS RMI bileşeni tarafından üretilen önceki izleme girişlerine bakın.
CSQCRSYN	Yeniden eşzamanlama	Yeniden eşzamanlama işlemi, görev için başlatılmadan önce.	Yeniden eşzamanlanmakta olan iş birimi için CICS olarak bilinen iş birimi tanıtıcısı.
CSQCSETH	SET tanıtıcısı	MQSET ' i IBM MQ for z/OSolarak yayınlamadan önce.	Nesne tanıtıcısı.
CSQCTASE	YalnızcaIBM kullanın		
CSQCTEST	İzleme sınaması	EXEC CICS ENTER TRACE çağrısında kullanılan, kullanıcı tarafından sağlanan izleme numarasını ya da bağlantının izleme durumunu doğrulamak için kullanılır.	Veri yok.
CSQCDCFF	YalnızcaIBM kullanın		

## Yordamsal program hatalarının işlenmesi

Bu bilgiler, bir çağrı yaparken ya da iletisi son hedefine teslim edildiğinde, uygulama MQI çağrılarınızla ilişkili hataları açıklar.

Mümkün olduğunda, kuyruk yöneticisi bir MQI çağrısı yapıldığında hata döndürür. Bunlar, *yerel olarak belirlenen hatalardır*.

Uzak bir kuyruğa ileti gönderirken, MQI çağrısı yapıldığında hatalar görünmeyebilir. Bu durumda, hata raporlarını tanımlayan kuyruk yöneticisi, kaynak programa başka bir ileti göndererek bunları bildirir. Bunlar *uzaktan belirlenen hatalardır*.

### Yerel olarak belirlenen hatalar

Yerel olarak saptanan hatalara ilişkin bilgiler: Bir MQI çağrısında hata, sistem kesintileri ve yanlış veri içeren iletiler.

Kuyruk yöneticisinin hemen rapor verebileceği en sık rastlanan üç hata nedeni:

- Bir MQI çağrısının başarısız olması; örneğin, kuyruğun dolu olduğu için

- Uygulamanızın bağlı olduğu sistemin bir kısmının çalıştırılmasına ilişkin bir kesinti; örneğin, kuyruk yöneticisi
- Başarıyla işlenemeyen verileri içeren iletiler


Zamanuyumsuz put olanağını kullanıyorsanız, hatalar hemen bildirilmez. Önceki zamanuyumsuz koyma işlemlerine ilişkin durum bilgilerini almak için MQSTAT çağrısını kullanın.

## Bir MQI çağrısının başarısız olması

Kuyruk yöneticisi, bir MQI çağrısının kodlamasındaki hataları hemen bildirebilir. Bu, önceden tanımlanmış bir dönüş kodu kümesini kullanarak yapar. Bunlar, tamamlanma kodlarına ve neden kodlarına bölünüyorlar.

Bir çağrı başarılı olup olmadığını göstermek için, kuyruk yöneticisi arama tamamlandığında bir *tamamlama kodu* döndürür. Başarılı, kısmi tamamlama ve çağrı başarısızlığı belirten üç tamamlama kodu vardır. Kuyruk yöneticisi ayrıca, kısmi tamamlama ya da çağrı hatasının nedenini belirten bir *neden kodu* değerini de döndürür.

Her bir çağrıya ilişkin tamamlanma ve neden kodları, [Dönüş kodları](#)' ta bu çağrıya ilişkin açıklamayla listelenir. Düzeltici eylemle ilgili fikirler de içinde olmak üzere daha ayrıntılı bilgi için bkz:

-  [IBM MQ for z/OS iletileri, tamamlama, ve neden kodları - IBM MQ for z/OS](#)
- Diğer tüm IBM MQ platformları için [Messages and reason codes](#)

Her çağrıdan oluşabilecek tüm dönüş kodlarını işlemek için programlarınızı tasarlayın.

## System i İnteruptları

Bağlı olduğu kuyruk yöneticisinin bir sistem hatasından kurtulması durumunda, uygulamanız herhangi bir kesintiden habersiz olabilir. Ancak, bu tür bir kesinti oluştuğunda verilerinizin kaybedilmediğinden emin olmak için uygulamanızı tasarlamamız gerekir.

Verilerinizin tutarlı olduğundan emin olmak için kullanabileceğiniz yöntemler, kuyruk yöneticinizin çalıştığı altyapıya bağlıdır:

### z/OS

In the CICS and IMS environments, you can make MQPUT and MQGET calls within units of work that are managed by CICS or IMS. Toplu iş ortamında, MQPUT ve MQGET çağrılarını aynı şekilde yapabilirsiniz; ancak, aşağıdakileri kullanarak eşitleme noktalarını bildirmeniz gerekir:

- IBM MQ for z/OS MQCMIT ve MQBACK çağrıları (bkz. [“İş birimlerinin kesinleştirilmesi ve yedeklenmesi”](#) sayfa 888 ) ya da
- İki aşamalı eşitleme noktası desteği sağlamak için z/OS Transaction Management ve Recoverable Resource Manager Services (RRS). RRS, IBM MQ ve diğer RRS etkinleştirilen ürün kaynaklarını ( Db2 saklanmış yordam kaynakları gibi) tek bir mantıksal iş birimi içinde güncellenmeye olanak sağlar. RRS eşitleme noktası desteği hakkında bilgi için bkz. [“Hareket yönetimi ve kurtarılabilir kaynak yöneticisi hizmetleri”](#) sayfa 892.

### IBM i

You can make your MQPUT and MQGET calls within global units of work that are managed by IBM i commitment control. 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 yönetmektedir.

## UNIX, Linux, and Windows sistemleri

Bu ortamlarda, MQPUT ve MQGET çağrılarınızı olağan biçimde yapabilirsiniz, ancak MQCMIT ve MQBACK çağrılarını kullanarak eşitleme noktalarını bildirmeniz gerekir (bkz. [“İş birimlerinin kesinleştirilmesi ve yedeklenmesi”](#) sayfa 888 ). CICS ortamında, MQCMIT ve MQBACK komutları, CICStarafından yönetilen iş birimleri içinde MQPUT ve MQGET çağrılarınızı yapabildiğiniz için devre dışı bırakılır.

Kaybetmeyi göze alamadığınız tüm verileri taşımak için kalıcı iletiler kullanın. Kuyruk yöneticisi bir hatadan kurtarılması gerekiyorsa, kalıcı iletiler kuyruklara geri dönmektedir. **ULW** With IBM MQ on UNIX, Linux, and Windows, an MQGET or MQPUT call within your application will fail at the point of filling all the log files, with the message MQRC\_RESOURCE\_PROBLEM. UNIX, Linux, and Windows' ta günlük dosyaları hakkında daha fazla bilgi için bkz. [Administering z/OS](#) z/OS için bkz. [Planning on z/OS](#).

Bir uygulama çalışırken kuyruk yöneticisi bir işletmen tarafından durdurulursa, susturma seçeneği genellikle kullanılır. Kuyruk yöneticisi, uygulamaların çalışmaya devam edebileceği bir susturma durumuna girer, ancak en kısa zamanda sona ermelidir. Küçük, hızlı uygulamalar, büyük olasılıkla susturulmuş durumu yoksayabilir ve normal olarak sona erdirilinceye kadar devam edebilir. Uygulamaların daha uzun çalıştırılması ya da iletilerin gelmesi beklenenler, MQOPEN, MQPUT, MQPUT1ve MQGET çağrılarını kullandıklarında *durdurulursa başarısız olur* seçeneğini kullanmalıdır. Bu seçenekler, kuyruk yöneticisi sustururken çağrılarının başarısız olduğu anlamına gelir; ancak, uygulamanın susturulmuş durumdan yoksayma çağrılarını yayınlayarak temizleme işlemini sonlandırma zamanı hala olabilir. Bu tür uygulamalar, gerçekleştirdikleri değişiklikleri de kesinleştirebilir ya da geri alabilir ve daha sonra sona erdirebilir.

Kuyruk yöneticisi durdurulmaya zorlandıysa (yani, susturulmuş durumdan dur), 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, UNIX, Linux, and Windows sistemlerinde bir MQDISC çağrısı yayınlayın.

## Yanlış veri içeren iletiler

Uygulamadaki iş birimlerini kullandığınızda, bir program kuyruktan aldığı bir iletiyi başarılı bir şekilde işleyemezse, MQGET çağrısını geri alır.

Kuyruk yöneticisi bir sayıyı (ileti tanımlayıcısının *BackoutCount* (Geri Sayıma Sayısı) alanında) tutar (ileti tanımlayıcısının). Bu sayı, etkilenen her iletinin tanımlayıcısında bu sayıyı korur. Bu sayı, bir uygulamanın verimliliğiyle ilgili değerli bilgiler sağlayabilir. Zaman içinde artan geri sayım olan iletiler sürekli olarak reddedilir; uygulamanızı tasarlayın, böylece bu tür iletileri buna göre işler ve bu tür iletileri işler.

**z/OS** IBM MQ for z/OS' ta, kuyruk yöneticisinin geri alma sayımının yeniden başlatılmasını sağlamak için, **HardenGetBackout** özneteliğini MQQA\_BACOUT HARDEDEDI; olarak ayarlayın; tersi durumda, kuyruk yöneticisi yeniden başlatmak zorunda kalıyorsa, her ileti için doğru bir geri alma sayısı tutmaz. Özneteliğin bu şekilde ayarlanması, fazladan işleme penaltısını ekler.

IBM MQ ' ta **IBM i** IBM i, Windows, UNIX and Linux sistemleri için, geriletme sayısı her zaman kuyruk yöneticisi yeniden başlatılmasına neden olur.

**z/OS** Ayrıca, IBM MQ for z/OS' ta, bir iş birimi içindeki bir kuyruktan iletileri kaldırdığınızda, iş birimi uygulama tarafından yedeklendiyse, bir iletiyi yeniden kullanılabilir kılınmayacak şekilde işaretleyebilirsiniz. İşaretsiz ileti, yeni bir iş birimi altında alınmış gibi işlem görür. İletiyi, MQGMO\_MARK\_SKIP\_BACKOUT seçeneğini kullanarak geri alma işlemini atlayacak şekilde işaretliyorsunuz.(MQGMO yapısında), MQGET çağrısını kullanın. Bu teknik ile ilgili daha fazla bilgi için bkz. "[Geri alma işlemi atlanıyor](#)" sayfa 835 .

## Sorun belirleme için rapor iletilerini kullanma

Uzak kuyruk yöneticisi, bir MQI aramanızı yaparken bir iletiyi kuyruğa koyamamanız gibi hataları raporlayamaz, ancak iletiyi nasıl işlediğini bildiren bir rapor iletileri gönderebilir.

Uygulamanızın içinde, (MQPUT) rapor iletilerini ve bunları alma seçeneğini belirleyebilirsiniz (bu durumda başka bir uygulama ya da bir kuyruk yöneticisi tarafından gönderilirler).

## Rapor iletileri oluşturma

Rapor iletileri, bir uygulamanın başka bir uygulamaya, gönderilen iletiyle baş edemeyeceğini bildirmesini sağlar.

Ancak, iletiyi gönderen uygulamanın herhangi bir sorun hakkında bilgilendirilmekle ilgilenip ilgilenmediğini belirlemek için *Report* alanının ilk olarak analiz edilmesi gerekir. Bir rapor iletinin gerekli olduğunu belirlemiş olmak, aşağıdakine karar vermeniz gerekir:

- Özgün iletinin tamamını, yalnızca ilk 100 baytlık verileri içermek isteyip istemediğinizi ya da özgün iletinin hiçbirini içermeyeceğini.
- Özgün iletiyle ne yapılır? Bunu atabilir ya da ölü-mektup kuyruğuna gönderebilirsiniz.
- *MsgId* ve *CorrelId* alanlarının içindekilerin de gerekli olup olmadığını.

Oluşturulan rapor iletinin nedenini belirtmek için *Feedback* alanını kullanın. Rapor iletilerinizi bir uygulamanın yanıtlama kuyruğuna yerleştirin. Ek bilgi için [Geribildirim](#) başlıklı konuya bakın.

## İsteme ve alma (MQGET) rapor iletileri

Başka bir uygulamaya ileti gönderdiğinizde, gereksinim duyduğunuz geri bildirim belirtmek üzere *Report* alanını tamamladığınız sürece herhangi bir sorun hakkında bilgilendirilmediğiniz anlamına gelir. Kullanılabilir seçenekler için bkz. [Rapor alanı yapısı](#).

Kuyruk yöneticileri her zaman bir uygulamanın yanıtlama kuyruğuna rapor iletileri koyar ve kendi uygulamalarının aynı şekilde yapılması önerilir. Rapor iletileri olanağını kullandığınızda, iletinizin ileti tanımlayıcısında yanıtlanacak yanıtınızın adını belirtin; aksi takdirde, MQPUT çağrısı başarısız olur.

Uygulamanızın yanıt kuyruğunuzu izleyen yordamları içermesi ve ona gelen iletileri işlemeniz gerekir. Bir rapor iletinin, özgün iletinin ilk 100 baytı, özgün iletinin ilk 100 baytı içerebileceğini ya da özgün iletinin hiçbirini içermediğini unutmayın.

Kuyruk yöneticisi, hata nedenini belirtmek için rapor iletinin *Feedback* alanını ayarlar; örneğin, hedef kuyruk yok. Programlarınız aynı şeyi yapmalıdır.

Rapor iletilerine ilişkin daha fazla bilgi için bkz. [“Rapor iletileri” sayfa 18](#).

## Uzaktan saptanan hatalar

Uzak bir kuyruğa ileti gönderdiğinizde, yerel kuyruk yöneticisi bir hata bulmadan MQI aramanızı işlese bile, diğer etmenler iletinizin uzak bir kuyruk yöneticisi tarafından nasıl işleneceğini etkileyebilir.

Örneğin, hedeflediğiniz kuyruk dolu olabilir ya da kuyruk yok olabilir. İletiniz, hedef kuyruğa yönlendirilecek diğer ara kuyruk yöneticileri tarafından işlenecekse, bunların herhangi biri bir hata bulabilir.

## İleti teslim edilmesi sorunları

Bir MQPUT çağrısı başarısız olduğunda, iletiyi yeniden kuyruğa yerleştirmeyi, gönderene geri döndürmeyi ya da bu iletiyi ölü harf kuyruğuna koymayı deneyebilirsiniz.

Her seçeneğin merits değeri vardır, ancak MQPUT ' un başarısız olmasının nedeni hedef kuyruğun dolu olduğu için bir ileti yeniden yerleştirmeyi denemek istemeyebilirsiniz. Bu örnekte, bunu ölü harf kuyruğuna koymak, bunu daha sonra doğru hedef kuyruğa ulaştırmanızı sağlar.

### İleti sağlamayı yeniden dene

Before the message is put on a dead-letter queue, a remote queue manager attempts to put the message on the queue again if the attributes *MsgRetryCount* and *MsgRetryInterval* have been set for the channel, or if there is a retry exit program for it to use (the name of which is held in the channel attribute *MsgRetryExitId* field).

*MsgRetryExitId* alanı boş bırakılırsa, *MsgRetryCount* ve *MsgRetryInterval* özniteliklerindeki değerler kullanılır.

*MsgRetryExitId* alanı boş değilse, bu adın çıkış programı çalıştırılır. Kendi çıkış programlarınızı kullanma hakkında daha fazla bilgi için bkz. [“İleti alışverişi kanallarına ilişkin kanal çıkışı programları” sayfa 993](#).

## İletiyi gönderene geri ver

Özgün iletinin tümünü içermek üzere bir rapor iletisi oluşturulmasını isteyerek gönderene bir ileti dönmenizi sağlar.

Rapor iletisi seçeneklerine ilişkin ayrıntılar için bkz. [“Rapor iletileri” sayfa 18](#).

## Ölü harf (teslim edilmemiş ileti) kuyruğunun kullanılması

Kuyruk yöneticisi bir iletiyi teslim edemediğinde, iletiyi ölüme mektup kuyruğuna yerleştirmeyi dener. Kuyruk yöneticisi kurulu olduğunda bu kuyruk tanımlanmalıdır.

Programlarınız, kuyruğun kullandığı kuyruğu, kuyruk yöneticisinin kullandığı aynı şekilde kullanabilir. Kuyruk yöneticisi nesnesini (MQOPEN çağrısını kullanarak) ve **DeadLetterQName** özniteliğini sorgulayarak (MQINQ çağrısını kullanarak), ölü-mektup kuyruğunun adını bulabilirsiniz.

Kuyruk yöneticisi bu kuyruğa bir ileti yerleştirdiğinde, iletiye üstbilgi ekler; bu biçim, ölü-mektup üstbilgisi (MQDLH) yapısı tarafından tanımlanır; bkz. MQDLH-Dead-letter header. Bu üstbilgi, hedef kuyruğun adını ve iletinin ölü harf kuyruğuna konmasının nedenini içerir. İleti, ileti istenen kuyruğa konmadan önce kaldırılmalı ve sorunun çözülmesi gerekir. Ayrıca, kuyruk yöneticisi, iletinin bir MQDLH yapısı içerdiğini göstermek için ileti tanımlayıcısının (MQMD) *Format* alanını değiştirir.

## MQDLH yapısı

Ölü-mektup kuyruğuna yerleştirdiğiniz tüm iletilere bir MQDLH yapısı eklemeniz önerilir; ancak, bazı IBM MQ ürünleri tarafından sağlanan harf kullanımı işleyicisini kullanmak istiyorsanız, iletilerinize bir MQDLH yapısı eklemelisiniz.

İletinin üstbilgisinin eklenmesi, iletiyi ölü harf kuyruğu için çok uzun bir hale getirebileceğinden, iletilerinizin, en azından MQ\_MSG\_HEADER\_LENGTH değişiminin değeri tarafından, ölü-harfli kuyruk için izin verilen büyüklük üst sınırından daha kısa olmasına dikkat edin. Bir kuyruğa izin verilen ileti boyutu üst sınırı, kuyruğun **MaxMsgLength** özniteliğinin değerine göre belirlenir. Ölü-mektup kuyruğu için, bu özniteliğin kuyruk yöneticisi tarafından izin verilen üst sınıra ayarlandığından emin olun. Uygulamanız bir iletiyi teslim edemiyorsa ve ileti, ölü-mektup kuyruğuna konmak için çok uzun olursa, MQDLH yapısının tanımında verilen öneriyi izleyin.

Ölü harf kuyruğunun izlendiğinden ve bu kuyruğun üzerine gelen iletilerin işlendiğinden emin olun. Ölü-harfli kuyruk işleyicisi bir toplu iş yardımcı programı olarak çalışır ve seçilen iletiler üzerinde, seçilen iletiler kuyruğunda çeşitli işlemler gerçekleştirmek için kullanılabilir. Daha fazla ayrıntı için bkz. [“Kuyruk-kuyruğun kuyruğunda işlenmesi” sayfa 1077](#).

Veri dönüştürme gerekliyse, MQGET çağrısında MQGMO\_CONVERT seçeneğini kullandığınızda, kuyruk yöneticisi üstbilgi bilgisini dönüştürür. İletiyi koyulan işlem bir MCA ise, üstbilgi, özgün iletinin tüm metni tarafından takip edilir.

Bu kuyruk için çok uzunsa, ölü-mektup kuyruğuna gönderilen iletiler kesilmiş olabilir. Bu durumun olası bir göstergesi, kuyruktaki iletiler, kuyruğun **MaxMsgLength** özniteliğinin değeriyle aynı uzunlukta olan iletilerdir.

### *Kuyruk-kuyruğun kuyruğunda işlenmesi*

Bu bilgiler, kuyruk-harf kuyruğunda işlem yaparken genel kullanıma açık programlama arabirimi bilgilerini içerir.

Çıkmaz mektup kuyruğu işleme, yerel sistem gereksinimlerine bağlıdır, ancak belirtimi çizdiğinizde aşağıdaki şeyleri göz önünde bulundurun:

- MQMD 'deki biçim alanının değeri MQFMT\_DEAD\_LETTER\_HEADER olduğu için, ileti, ölü-mektup kuyruğu üstbilgisine sahip olduğu tanımlanabilir.
- On IBM MQ for z/OS using CICS, if an MCA puts this message to the dead-letter queue, the *PutAppLType* field is MQAT\_CICS, and the *PutAppLName* field is the *AppLId* of the CICS system followed by the transaction name of the MCA.
- İletinin ölü-mektup kuyruğuna yönltilmesinin nedeni, ölü-harfli kuyruk üstbilgisinin *Reason* alanında yer alır.

- Ölü-harfli kuyruk üstbilgisi, hedef kuyruk adı ve kuyruk yöneticisi adına ilişkin ayrıntıları içerir.
- Ölü-mektup kuyruğu üstbilgisi, ileti hedef kuyruğuna konmadan önce ileti tanımlayıcısında yeniden yürürlüğe konması gereken alanları içerir. Bu bilgiler şunlardır:
  1. *Encoding*
  2. *CodedCharSetId*
  3. *Format*
- İleti tanımlayıcısı, gösterilen üç alan (Encoding, CodedCharSetId ve Format) dışında, özgün uygulama tarafından PUT ile ayıdır.

Ölü harf kuyruğu uygulamanızın aşağıdaki şeylerden birini ya da birkaçını yapması gerekir:

- *Reason* alanını inceleyin. Aşağıdaki nedenlerle bir ileti MCA tarafından konulmuş olabilir:
  - İleti, kanala ilişkin ileti boyutu üst sınırından daha uzun  
Neden: MQRC\_MSG\_TOO\_BIG\_FOR\_CHANNEL
  - İleti hedef kuyruğuna konamadı.  
Neden, bir MQPUT işlemi tarafından döndürülebilen herhangi bir MQRC\_ \* neden koddur.
  - Bir kullanıcı çıkışı bu işlemi istedi  
Neden kodu kullanıcı çıkışıyla ya da varsayılan MQRC\_SUPPRESSED\_BY\_EXIT tarafından sağlanır.
- İletiyi amaçlanan hedefe iletmeye çalışın, bu mümkün olabilir.
- Şaşırtma nedenini saptmadan önce belirli bir süre boyunca iletiyi alıkoyma, ancak hemen düzeltilebilir değil.
- Yöneticilere, bunların belirlendiği yerlerde doğru sorunları çözmeleri için yönergeler verin.
- Bozuk ya da uygulanamaz olan iletileri atın.

Ölü harf kuyruğundan kurtardığınız iletilerle başa çıkmak için iki yol vardır:

#### 1. İleti yerel bir kuyruksa:

- Uygulama verilerini çıkarmak için gereken kod çevirilerini gerçekleştirme
- Bu yerel bir işlevse, bu verilerde kod dönüştürmeleri gerçekleştirilmeyi sağlar
- Sonuç iletilerini yerel kuyruğa, geri yüklenen ileti tanımlayıcısının tüm ayrıntılarıyla birlikte yerel kuyruğa koyun

#### 2. İleti uzak bir kuyruk için gönderilmişse, iletiyi kuyruğa koyun.

Teslim edilmeyen iletilerin dağıtılmış bir kuyruğa alma ortamında nasıl işlendiği hakkında bilgi için bkz. [Bir ileti teslim edilemezse ne olur?](#).

## Çoklu yayın 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, mümkün olduğu kadar kullanıcı için şeffaf olacak şekilde tasarlandı ve yine de var olan uygulamalarla uyumlu olacak şekilde tasarlandı. Bir COMMINFO nesnesi tanımlanması ve KONU nesnesinin **MCAS** ve **COMMINFO** parametrelerinin ayarlanması, var olan IBM MQ uygulamalarının çoklu yayını kullanmak için önemli bir yeniden yazma gerektirmemesi anlamına gelir. Ancak, bazı sınırlamalar olabilir (ek bilgi için bkz. [“Çoklu Yayın ve MQI” sayfa 1078](#)) ve dikkat edilmesi gereken bazı güvenlik sorunları (ek bilgi için bkz. [Çoklu Yayın Güvenliği](#)).

## Çoklu Yayın ve MQI

Ana İleti Kuyruğu Arabirimi (MQI) kavramlarını ve bu kavramların IBM MQ Multicast ile nasıl ilişkili olduğunu anlamak için bu bilgileri kullanın.

Çok hedefli abonelikler kalıcı değildir; ilgili fiziksel kuyruklar olmadığından, kalıcı abonelikler tarafından oluşturulan çevrimdışı iletilerin saklanacak bir yeri yoktur.

Bir uygulama çok noktaya gönderim konusuna abone olduktan sonra, bir nesne tanıtıcısı gibi, kuyruğun kullanabileceği bir nesne tanıtıcısı ya da MQGET işlemi tarafından verilir. Bu, yalnızca yönetilen birden çok noktaya gönderim aboneliklerinin (MQSO\_YANED ile yaratılan abonelikler) desteklediği anlamına gelir; bu, abonelik yapmak ve iletileri bir kuyrukta 'noktası' olarak göstermek olanaklı değildir. Bunun anlamı, iletilerin abonelik çağrısında döndürüldüğü nesne tanıtıcısından tüketilmesi gerektiği anlamına gelir. İstemcide iletiler, istemci tarafından tüketilinceye kadar bir ileti arabelleğiyle saklanır; ek bilgi için bkz. İstemci yapılandırma dosyasının MessageBuffer kısmı . İstemci yayınlama hızına uymuyorsa, iletiler gerektiği gibi atılır ve en eski iletiler önce atılır.

Genellikle, bir uygulamanın KONU nesnesinin MCAST özneliği ayarlanarak, uygulamanın Multicast ya da not Multicast özelliğini kullanıp kullanmadığı bir yönetim karardır. Bir yayınlama uygulamasının çoklu yayın kullanımının kullanılmamasını sağlaması gerekiyorsa, bu uygulama MQ00\_NO\_MULTICAST seçeneğini kullanabilir. Similarly, a subscribing application can ensure that multicast is not used by subscribing with the MQSO\_NO\_MULTICAST option.

IBM MQ Multicast, ileti seçicilerinin kullanımını destekler. Bir uygulama, bir uygulama tarafından yalnızca, seçim dizgisinin temsil ettiği SQL92 sorgusuna uyan özelliklere sahip olan iletilere kaydolmak için kullanılır. İleti seçiciyle ilgili daha fazla bilgi için bkz. “Seçiciler” sayfa 29.

Aşağıdaki çizelge, tüm ana MQI kavramlarını ve bunların Multicast ile nasıl ilişkilendirileceğini göstermektedir:

MQI Kavramı	Çoklu yayın kullanılarak denendiğinde işlem	Neden kodu
Sıfır uzunluklu bir ileti koymak	Reddedildi	<u>2005 (07D5) (RC2005): MQRC_BUFFER_LENGTH_ERROR</u>
Gruplandırma	Reddedildi	<u>2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR</u>
Bölümleme	Reddedildi	<u>2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWEND</u>
Dağıtım listeleri	Reddedildi	<u>2154 (086A) (RC2154): MQRC_RECS_PRESENT_ERROR</u>
MQINQ	Konular için reddedilen konular: MQINQ ve MQSET konuları desteklenmiyor.	<u>2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_SORGULAMA</u>
MQINQ	Yönetilen tanıtıcı için kabul edildi. Yalnızca Yürürlükteki Derinlik sorgulanabilir.	<ul style="list-style-type: none"><li>Değer Yürürlükteki Derinlik ise, geçerli bir neden kodu yoktur.</li><li>Değer, Yürürlükteki Derinlik değerinden başka bir değerse, neden kodu <u>2067 (0813) (RC2067): MQRC_SELECTOR_ERROR</u>.</li></ul>
MQSET	Tüm tutamaçlar için reddedildi.	<u>2040 (07F8) (RC2040): MQRC_NOT_OPEN_FOR_SET</u>
İşlemler (XA ya da değil)	Reddedildi	<u>2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_AVAILABLE</u>
İleti göz atma	Reddedildi	<u>2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE</u>

Çizelge 161. MQI kavramları ve bunların çok hedefli olarak nasıl ilişkilendirildikleri (devamı var)		
MQI Kavramı	Çoklu yayın kullanılarak denendiğinde işlem	Neden kodu
İletileri kilitle	Reddedildi	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
İşaretle göz at	Reddedildi	2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE
Geçiş bağlamı	Reddedildi	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPUT1	Reddedildi. Bir Multicast only konusuna denemek için MQPUT1 ' yi geçersiz kılar.	2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
Sürekli Abonelik	Konu "Yalnızca çoklu yayın" olarak işaretlenmişse, tersi durumda, Multicast aboneliği olmayan bir abonelik yapılır.	2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWALIZE
TopicString > 255	Reddedildi. Konu dizesi 255 karakterden fazlaysa, istemcide reddedilir.	2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR
Yönetilmeyen abonelik yapıldı	Konu "Yalnızca çoklu yayın" olarak işaretlenmişse, tersi durumda, Multicast aboneliği olmayan bir abonelik yapılır.	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
MQPMO_NOT_OWN_SUBS	Reddedildi	2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Aşağıdaki öğeler, önceki çizelgeden bazı MQI kavramlarını genişletebilir ve çizelgede olmayan bazı MQI kavramlarına ilişkin bilgi sağlar:

#### İleti kalıcılığı

Kalıcı olmayan çok noktaya yayın aboneleri için, yayıncıdan gelen kalıcı iletiler kurtarılamaz bir şekilde teslim edilir.

#### İleti kesme

İletin kesilmesi destekleniyor; bu da, uygulamanın aşağıdaki gibi bir uygulama için mümkün olduğunu gösterir:

1. Bir MQGET komutu verin.



2. MQRCTRUNCATEDMSG\_BAŞARISIZ oldu.
3. Daha büyük bir arabellek ayırın.
4. İletiyi almak için MQGET ' yi yeniden yayınlayın.

### Abonelik süre bitimi

Abonelik süre bitimi desteklenmiyor. Süre bitimi ayarlama girişimi yoksayılr.

## Çok noktaya gönderim için yüksek kullanılabilirlik

IBM MQ Multicast sürekli eşler arası işlemi anlamak için bu bilgileri kullanın; ancak IBM MQ , bir IBM MQ kuyruk yöneticisine bağsa da, iletiler kuyruk yöneticiliklerinden geçmez.

Bir kuyruk yöneticisine yönelik bir bağlantının MQOPEN ya da MQSUB için çok hedefli konu nesnesi için yapılması gerekse de, iletiler kuyruk yöneticisinde kendilerini aklamamalıdır. Bu nedenle, çok hedefli konu nesnesinde MQOPED ya da MQSUB tamamlandıktan sonra, kuyruk yöneticisine yönelik bağlantı kaybolduysa bile çoklu yayın iletilerini iletme işlemine devam etmek mümkündür. İki işlem kipi vardır:

### Kuyruk yöneticisine olağan bir bağlantı yapılır

Kuyruk yöneticisiyle bağlantı varsa, çoklu yayın iletişimi mümkün. Bağlantı başarısız olursa, normal MQI kuralları uygulanır; örneğin, çok hedefli nesne tanıtıcısında bir MQPUT işlemi 2009 (07D9) (RC2009): MQRCONNECTION\_BROKENdeğerini döndürür.

### Kuyruk yöneticisinden bir istemci bağlantısı yeniden bağlanıyor

Yeniden bağlantı döngüsü sırasında bile çoklu yayın iletişimi mümkün. Başka bir deyişle, kuyruk yöneticisine yönelik bağlantı kesilse bile, çoklu yayın iletilerinin yerleştirilmesinin ve tüketilmesinin etkilenmemesi anlamına gelir. İstemci bir kuyruk yöneticisine yeniden bağlanmayı dener ve bu yeniden bağlantı başarısız olursa, bağlantı tanıtıcısı bozuk olur ve çoklu yayın olanlar da içinde olmak üzere tüm MQI çağrıları başarısız olur. Ek bilgi için şu konuya bakın: Automatic client reconnection

Herhangi bir uygulama bir MQDISC ' yi belirttik olarak yayınlarsa, tüm çoklu yayın abonelikleri ve nesne tanıtıcıları kapatılır.

## Çok hedefli sürekli eşdüzeyler arası işlem

İstemciler arasındaki eşdüzeyler arası iletişimin avantajlarından biri, iletilerin kuyruk yöneticisi aracılığıyla akmasına gerek kalmaması; dolayısıyla, kuyruk yöneticisi ile bağlantı kesmesi durumunda, ileti aktarımı devam eder. Bu kipi sürekli ileti gereksinimleri için aşağıdaki kısıtlamalar geçerlidir:

- Kesintisiz işlem için MQCNO\_RECONNECT\_ \* seçeneklerinden biri kullanılarak bağlantı yapılmalıdır. Bu işlem, iletişim oturumu bozuk olsa da, gerçek bağlantı tanıtıcısı bozulmamış olsa da, bunun yerine yeniden bağlanma durumunda olduğu anlamına gelir. Yeniden bağlanma başarısız olursa, bağlantı tanıtıcısı artık bozuk olur ve bu da tüm MQI çağrılarını önler.
- Bu kipte yalnızca MQPUT, MQGET, MQINQ ve Async Consume desteklenmektedir. Herhangi bir MQOP, MQCLOSE ya da MQDISC fillerinin tamamlanması için kuyruk yöneticisine yeniden bağlantı yapılması gerekir.
- Durum, kuyruk yöneticisi durağına akar; kuyruk yöneticisinde herhangi bir durum eski ya da eksik olan herhangi bir durum olabilir. Bu, istemcilerin iletileri gönderip alabileceği ve kuyruk yöneticisinde herhangi bir durumun bilinmediği anlamına gelir. Daha fazla bilgi için bakınız: Multicast application monitoring

## Çok hedefli ileti alışverişi için MQI ' de veri dönüştürme

Use this information to understand how data conversion works for IBM MQ Multicast messaging.

IBM MQ Multicast paylaşılan, bağlantısız bir iletişim kuralıdır ve bu nedenle her istemcinin veri dönüştürmesi için belirli istekler hazırlanması mümkün değildir. Aynı çoklu yayın akışına abone olan her istemci aynı ikili verileri alır; bu nedenle, IBM MQ veri dönüştürme gerekiyorsa, dönüştürme her istemcide yerel olarak gerçekleştirilir.

Data is converted on the client for IBM MQ Multicast traffic. **MQGMO\_CONVERT** seçeneği belirtilirse, veri dönüştürme işlemi istendiği gibi yapılır. Kullanıcı tanımlı biçimler, istemcide kurulu veri dönüştürme

çıkışa gereksinim duyarlar; istemci ve sunucu paketlerinde şimdi kitaplıkların hangi kitaplıkların olduğunu görmek için bkz. “Veri dönüştürme çıkışları yazılıyor” sayfa 1014 .

Veri dönüştürmenin yönetilmesine ilişkin bilgi için [Multicast ileti alışverişi için veri dönüştürmenin etkinleştirilmesibaşlıklı konuya](#) bakın.

Veri dönüştürme hakkında daha fazla bilgi için bkz. [Veri dönüştürme](#).

Veri dönüştürme çıkışları ve ClientExitPathile ilgili daha fazla bilgi için, istemci yapılandırma dosyasının [ClientExitYol](#) kısmına bakın.

## Çok noktaya yayın kural dışı durumu

Use this information to learn about IBM MQ Multicast event handlers and reporting IBM MQ Multicast exceptions.

IBM MQ Multicast, standart IBM MQ olay işleyici mekanizması kullanılarak raporlanan çoklu yayın olaylarını raporlamak için olay işleyicisini çağırarak sorun belirlemeye yardımcı olur.

Tek bir çok noktaya gönderim olayı, birden çok IBM MQ bağlantısının çağırılması durumunda, aynı çoklu yayın vericisi ya da alıcısı kullanılarak birden çok MQHCONN bağlantı noktası olabileceği için ortaya çıkan bir olaydır. Ancak, her çoklu yayın kural dışı durumu, IBM MQ bağlantısı başına tek bir olay işleyicinin çağırılmasına neden olur.

IBM MQ MQCBDO\_EVENT\_CALL sabiti, uygulamaların yalnızca IBM MQ olaylarını almak için bir geri bildirme kaydetmesini sağlar ve MQCBDO\_MC\_EVENT\_CALL , uygulamaların yalnızca çok noktaya gönderim olaylarını alacak bir geri bildirme kaydetmesini sağlar. Her iki değişmez de kullanılırsa, her iki olay tipi de alınır.

## Multicast olayları istenmesi

IBM MQ Multicast events use the MQCBDO\_MC\_EVENT\_CALL constant in the cbd.Options field. Aşağıdaki örnek, çok hedefli olayların nasıl isteneceğini göstermektedir:

```
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 olaylarını göndermektedir. To request that both types of events are sent to the event handler, the application must specify the MQCBDO\_EVENT\_CALL constant in the cbd.Options field as well as the MQCBDO\_MC\_EVENT\_CALL constant as shown in the following example:

```
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 değişmezlerden hiçbiri kullanılmazsa, olay işleyiciye yalnızca bağlantı düzeyi olayları gönderilir.

Options alanına ilişkin değerler hakkında daha fazla bilgi için bkz. [Seçenekler \(MQlong\)](#).

## Çok hedefli olay biçimi

IBM MQ Multicast kural dışı durumları, geri çağırma işlevinin **Buffer** değiştirgesinde döndürülen destekleyici bazı bilgileri içerir. **Buffer** işaretçisi bir işaretçiler dizisini işaret eder ve MQCBC.DataLength alanı, dizinin bayt cinsinden büyüklüğünü belirtir. Dizinin ilk ögesi her zaman olayın kısa bir metin açıklamasına işaret eder. Olayın tipine bağlı olarak daha fazla parametre sağlanabilir. Aşağıdaki çizelge kural dışı durumları listeler:

Çizelge 162. Çok hedefli olay kodu açıklamaları

Olay Kodu	Tanım	Ek veriler
MQMCEV_PACKET_LOSS	Kurtarılamayan paket kaybı	Kaybedilen paket sayısı
MQMCEV_HEARTBEAT_TIMEOUT	Sağlıklı işletim bildirimi paketinin uzun olmaması	N/A
MQMCEV_VERSION_CONFLIP	Daha yeni protokol sürümü paketleri alma	N/A
MQMCEV_GÜVENİLİRLİĞİ	Verici ve alıcı vericinin farklı güvenilirlik kipleri	N/A
MQMCEV_CLOSED_TRANS	Konu iletimi 1 kaynak tarafından kapatılmış	N/A
MQMCEV_STREAM_ERROR	Akıшта hata saptandı	N/A
MQMCEV_NEW_SOURCE	Konu üzerinde yeni bir kaynak iletmeye başlar	Kaynak yapı
MQMCEV_RECEIVE_QUEUE_TRIMLED	Zaman ya da alan süre bitimi nedeniyle PacketQ ' dan kaldırılan paketler	Kırılan paketlerin sayısı
MQMCEV_PACKET_LOSS_NACK_EXPLORY	NACK süre bitimi nedeniyle kurtarılamayan paket kaybı	Kaybedilen paket sayısı
MQMCEV_ACK_RETRIES_EXCEED	<b>max_ack_retries</b> aşıldıktan sonra geçmişten kaldırılan paketler	Kaldırılan paket sayısı
MQMCEV_STREAM_SUSPEND_NACK	Bu konu tarafından kabul edilen bir akışta NACK ' lar askıya alındı	Akış tanıtıcısını askıya al Akımın askıya alındığı süre (milisaniye olarak)
MQMCEV_STREAM_RESUME_NACK	NACK ' lar bir akışta askıya alındıktan sonra sürdürülüyor	Akış Tanıtıcısı
MQMCEV_STREAM_ATILDI	Bu konu tarafından kabul edilen bir akış, bir except isteği nedeniyle reddedildi	Akış Tanıtıcısı
MQMCEV_FIRST_ILETISI	Bir kaynaktan ilk ileti	İleti numarası
MQMCEV_LATE_JOIN_FAILURE	Geciken birleştirme oturumu başlatılamadı	N/A
MQMCEV_MESSAGE_LOSS	Kurtarılamaz ileti kaybı	Kaybedilen iletilerin sayısı
MQMCEV_SEND_PACKET_FAILURE	Çok noktaya yayın vericisi çok noktalı bir paket gönderemedi	N/A
MQMCEV_REPAIR_DELAY	Çok hedefli alıcı, bekleyen bir NAK için onarım paketi almadı	N/A
MQMCEV_MEMORY_ALERT_ON	Alıcı alma arabellekleri dolduruyor	Arabellek havuzu kullanım yüzdesi
MQMCEV_MEMORY_ALERT_OFF	Alıcı alma arabellekleri olağan durumda.	Arabellek havuzu kullanım yüzdesi

Çizelge 162. Çok hedefli olay kodu açıklamaları (devamı var)		
Olay Kodu	Tanım	Ek veriler
MQMCEV_NACK_ALERT_ON	Alıcı onarımı paket isteği hızı, yüksek su işaretine ulaştı	Saniye başına paketlerde geçerli onarım isteği hızı
MQMCEV_NACK_ALERT_OFF	Günlük nesnesi onarma paketi istek hızı aşağı doğru	Saniye başına paketlerde geçerli onarım isteği hızı
MQMCEV_REPAIR_ALERT_ON	Verici onarımı paket gönderme hızı, yüksek su işaretine ulaştı	N/A
MQMCEV_REPAIR_ALERT_OFF	Verici onarma paketi gönderme hızı normale indi	N/A
MQMCEV_SHM_DEST_EWISIFY	Bir verici konu hedefi tarafından kullanılan Paylaşılan Bellek bölgesinin kullanılamaz olduğu saptandı.	N/A
MQMCEV_SHM_PORT_KULLANILAMAZ	Bir günlük nesnesi eşgörünümü tarafından kullanılan Paylaşılan Bellek kapısının kullanılmadığı saptandı.	N/A
MQMCEV_CCT_GETTIME_FAILED	Eşgüdümlü küme zamanından alma süresi başarısız oldu	N/A
MQMCEV_DEST_INTERFACE_FAILURE	Bir verici konu hedefi tarafından kullanılan ağ arabirimi başarısız oldu ve bir yedek ağ arabirimi kullanılamıyor	
MQMCEV_DEST_INTERFACE_FAILOVER	Bir verici konu hedefi tarafından kullanılan ağ arabirimi başarısız oldu ve başka bir Arabirim için başarılı bir hata durumunda yedek sisteme geçiş işlemi tamamlandı.	
MQMCEV_PORT_INTERFACE-HATA	rmmPort nesnesi tarafından kullanılan ağ arabirimi başarısız oldu ve bir yedek ağ arabirimi kullanılamıyor (ya da başarısız da var)	<u>RMM yapılandırması</u>
MQMCEV_PORT_INTERFACE_FAILOVER	The network interface used by a receiver rmmPort has failed and a successful failover to another Interface has been completed	<u>RMM yapılandırması</u>

## C içinde kodlama

C içindeki IBM MQ programlarını kodlarken aşağıdaki bölümlerdeki bilgileri not edin.

- [“MQI çağrılarına ilişkin deęiřtirgeler” sayfa 1085](#)
- [“Tanımlanmamıř veri tipine sahip parametreler” sayfa 1085](#)
- [“Veri tipleri” sayfa 1085](#)
- [“İkili dizgileri iřleme” sayfa 1085](#)
- [“Karakter dizgilerini iřleme” sayfa 1086](#)

- “Yapılara ilişkin ilk değerler” sayfa 1086
- “Dinamik yapılar için ilk değerler” sayfa 1086
- “C++ dilinde kullan” sayfa 1087

## MQI çağrılarına ilişkin değıştirgeler

*yalnızca giriş* ve MQHCONN, MQHOBJ, MQHMSG ya da MQHOMEN tipinde olan değıştirgeler değeri tarafından geçirilir; diğeri tüm parametreler için, değıştirgenin *adres*i değeri değeri olarak geçirilir.

Bir işlev çağrıldığında, adresle geçirilen tüm parametrelerin belirtilmesi gerekir. Belirli bir parametrenin gerekli olmadığı durumlarda, parametre verilerinin adresi yerine, işlev çağırımında parametre olarak boş değeri bir gösterge belirtilebilir. Bunun olası olduğu parametreler, çağrı açıklamalarında tanımlanır.

İşleve değeri olarak bir parametre döndürülmez; C terminolojisinde, bu, tüm işlevlerin geçersiz döndürdüğü anlamına gelir.

İşleve ilişkin öznitelikler MQENTRY makro değışkeniyle tanımlanır; bu makro değışkeninin değeri ortama bağlıdır.

## Tanımlanmamış veri tipine sahip parametreler

MQGET, MQPUT ve MQPUT1 işlevlerinin her biri, tanımlı olmayan bir veri tipine sahip bir **Buffer** parametresine sahip. Bu parametre, uygulamanın ileti verilerini göndermek ve almak için kullanılır.

Bu sıralamayı içeren değıştirgeler, C örneklerinde MQBYTE dizisi olarak gösterilir. Parametreleri bu şekilde bildirebilirsiniz, ancak genellikle iletilerde verilerin yerleşim düzenini tanımlayan yapı olarak bildirilmesi daha kolay olur. İşlev parametresi bir işaretçi-void değeri olarak bildirilir ve bu nedenle, herhangi bir verinin adresi, işlev çağırımı üzerinde parametre olarak belirtilebilir.

## Veri tipleri

Tüm veri tipleri `typedef` deyimiyle tanımlanır.

Her veri tipi için, ilgili gösterge veri tipi de tanımlıdır. İşaretçi veri tipinin adı, bir işaretçiyi göstermek için P harfiyle öneki olan temel ya da yapı veri tipinin adıdır. İşaretçinin öznitelikleri MQPOINTER makro değışkeni tarafından tanımlanır; bu makro değışkeninin değeri ortama bağlıdır. Aşağıdaki kod işaretçi veri tiplerinin nasıl bildirileceğini göstermektedir:

```
#define MQPOINTER          /* depends on environment */
...
typedef MQLONG MQPOINTER PMQLONG; /* pointer to MQLONG */
typedef MQMD MQPOINTER PMQMD; /* pointer to MQMD */
```

## İkili dizgileri işleme

İkili veri dizgileri, MQBYTEn veri tiplerinden biri olarak bildirilir.

Whenever you copy, compare, or set fields of this type, use the C functions `memcpy`, `memcmp`, or `memset`:

```
#include <string.h>
#include "cmqc.h"

MQMD MyMsgDesc;

memcpy(MyMsgDesc.MsgId,          /* set "MsgId" field to nulls */
       MQMI_NONE,               /* ...using named constant */
       sizeof(MyMsgDesc.MsgId));

memset(MyMsgDesc.CorrelId,       /* set "CorrelId" field to nulls */
       0x00,                    /* ...using a different method */
       sizeof(MQBYTE24));
```

Bu, MQBYTE24 olarak bildirilen verilerle doğru bir şekilde çalışmadığından, strcpy, strcmp, strncpy ya da strncmp dizgisiyle ilgili dizilimi kullanmayın.

## Karakter dizgilerini işleme

Kuyruk yöneticisi uygulamaya karakter verisi döndürdüğünde, kuyruk yöneticisi karakter verilerini her zaman, alanın tanımlı uzunluğuna sahip olacak şekilde boşluklarla doldurur. Kuyruk yöneticisi boş karakterle biten dizgileri döndürmüyor, ancak bunları girişinizde kullanabilirsiniz. Bu nedenle, bu tür dizgileri kopyalarken, karşılaştırırken ya da bitiştirirken, strncpy, strcmp ya da strncat dizgi işlevlerini kullanın.

Dizginin boş değerli (strcpy, strcmp ve strcat) sona erdirilmesini gerektiren dizgi işlevlerini kullanmayın. Ayrıca, dizilimin uzunluğunu belirlemek için strlen işlevini kullanmayın; alanın uzunluğunu belirlemek için büyüklük işlevi yerine kullanılır.

## Yapılara ilişkin ilk değerler

İçerme dosyası <mqqc.h>, bu yapıların somut örneklerini bildirirken yapılara ilişkin başlangıç değerlerini sağlamak için kullanabileceğiniz çeşitli makro değişkenlerini tanımlar. Bu makro değişkenlerinin adları MQxxx\_default biçiminin adlarına sahiptir; burada MQxxx, yapının adını gösterir. Bunları aşağıdaki gibi kullanın:

```
MQMD MyMsgDesc = {MQMD_DEFAULT};
MQPMO MyPutOpts = {MQPMO_DEFAULT};
```

Bazı karakter alanları için, geçerli olan MQI (örneğin, *StrucId* alanları için ya da MQMD 'deki *Format* alanı için) belirli değerleri tanımlar. Geçerli değerlerin her biri için iki makro değişkeni sağlanır:

- Bir makro değişkeni, değeri, alanın tanımlı uzunluğuna tam olarak uyan örtük boş (null) boş değer dışında bir uzunluğa sahip bir dizgi olarak tanımlar. Örneğin, simge boş bir karakteri temsil eder:

```
#define MQMD_STRUC_ID "MD--"
#define MQFMT_STRING "MQSTR--"
```

Bu formu memcpy ve memcpy işlevleriyle kullanın.

- Diğer makro değişkeni değeri bir karakter dizisi olarak tanımlıyor; bu makro değişkeninin adı, \_ARRAY ile birlikte suffixed dizgi biçiminin adıdır. Örneğin:

```
#define MQMD_STRUC_ID_ARRAY 'M','D',' ',' '
#define MQFMT_STRING_ARRAY 'M','Q','S','T','R',' ',' ',' ' "
```

Bu formu, yapının bir eşgörünümü, MQMD\_XX\_ENCODE\_CASE\_ONE default makro değişkeni tarafından sağlanan değerlerden farklı değerlerle bildirildiğinde alanı kullanıma hazırlamak için kullanın.

## Dinamik yapılar için ilk değerler

Bir yapının eşgörünümlerinin değişken sayısı gerektiğinde, yönetim ortamları tipik olarak, calloc ya da malloc işlevleri kullanılarak dinamik olarak elde edilen ana saklama alanı içinde yaratılır.

Bu tür yapılardaki alanları kullanıma hazırlamak için aşağıdaki teknik önerilir:

1. Yapıyı kullanıma hazırlamak için uygun MQxxx\_XX\_ENCODE\_CASE\_ONE default makro değişkenini kullanarak yapının bir eşgörünümünü bildirin. Bu yönetim ortamı, diğer yönetim ortamları için *model* olur:

```
MQMD ModelMsgDesc = {MQMD_DEFAULT};
/* declare model instance */
```

Gerekli olduğu şekilde, model eşgörünümü statik ya da dinamik kullanım süresini vermek için bildirimde bulunan statik ya da otomatik anahtar sözcükleri kodlayın.

2. Yapıya ilişkin dinamik bir yönetim ortamı 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 eşgörünüme kopyalamak için `memcpy` işlevini kullanın:

```
memcpy(InstancePtr,&ModelMsgDesc,sizeof(MQMD));  
/* initialize dynamic instance */
```

## C++ dilinde kullan

C++ programlama dili için, üstbilgi kütükleri, yalnızca bir C++ derleyicisi kullanıldığında içerilen aşağıdaki ek deyimleri içerir:

```
#ifdef __cplusplus  
extern "C" {  
#endif  
  
/* rest of header file */  
  
#ifdef __cplusplus  
}  
#endif
```

## Windows Visual Basicinde kodlama

Microsoft Visual Basic' taki IBM MQ programlarını kodlarken dikkate alınacak bilgiler. Visual Basic yalnızca Windows üzerinde desteklenir.

**Not:** From IBM WebSphere MQ 7.0, outside the .NET environment, support for Visual Basic (VB) has been stabilized at the IBM WebSphere MQ 6.0 level. IBM WebSphere MQ 7.0 ya da sonraki yayın düzeylerine eklenen en yeni işlev VB uygulamaları tarafından kullanılamaz. VB.NET' de programlıyorsanız, .NET için IBM MQ sınıflarını kullanın. Daha fazla bilgi için bkz [Developing .NET applications](#).

IBM MQ 9.0' tan Microsoft Visual Basic 6.0 için IBM MQ desteği kullanımdan kaldırılmıştır. IBM MQ classes for .NET are the recommended replacement technology.

To avoid unintended translation of binary data passing between Visual Basic and IBM MQ, use an `MQBYTE` definition instead of `MQSTRING`. `CMQB.BAS`, C baytlık bir tanımlamayla eşdeğer birkaç yeni `MQBYTE` tipi tanımlar ve bunları IBM MQ yapıları içinde kullanır. Örneğin, `MQMD` (ileti tanımlayıcı) yapısı için, `MsgId` (ileti tanıtıcısı) `MQBYTE24` olarak tanımlanır.

Visual Basic, işaretçi veri tipine sahip değildir, bu nedenle diğer IBM MQ veri yapılarına yapılan başvurular, işaretçi yerine görel konuma göre olur. İki bileşen yapısından oluşan bir bileşik yapı bildirin ve çağrıya ilişkin bileşik yapıyı belirtin. Visual Basic için IBM MQ desteği, bu işlemi mümkün kılmak için bir `MQCONNAny` çağrısı sağlar ve istemci uygulamalarının istemci bağlantısında kanal özelliklerini belirtmesine olanak sağlar. Tipik `MQCNO` yapısının yerine tip atanmamış bir yapıyı (`MQCNOCD`) kabul eder.

`MQCNOCD` yapısı, `MQCNO` ve arkasından gelen bir `MQCD` ' den oluşan bir bileşik yapıdır. Bu yapı, `CMQXB` ' nin çıkış üstbilgi dosyası olarak bildirilmiş. `MQCNOCD_DEFAULTS` yordamını kullanarak bir `MQCNOCD` yapısını kullanıma hazırlayın. `MQCONNX` çağrıları yapan bir örnek (`amqscnxb.vbp`) sağlanıyor.

`MQCONNX`, `MQCONNX` ile aynı parametrelere sahiptir; ancak, **ConnectOpts** parametresinin `MQCNO` veri tipi yerine herhangi bir veri tipi olarak bildirilmiş olması dışında. Bu, işlevin `MQCNO` ya da `MQCNOCD` yapısını kabul etmesini sağlar. Bu işlev, ana üstbilgi dosyası `CMQB` ' de bildirilir.

### İlgili kavramlar

“Preparing Visual Basic programs in Windows” sayfa 1056

Windows üzerinde Microsoft Visual Basic programlarını kullanırken dikkate alınması gereken bilgiler.

## İlgili başvurular

“Visual Basic uygulamalarını IBM MQ MQI client koduyla bağlantılandırma” sayfa 952  
Microsoft Visual Basic uygulamalarını Windows üzerindeki IBM MQ MQI client kodlarıyla bağlantılayabilirsiniz.

## COBOL içinde kodlama

COBOL ' de IBM MQ programları kodlanırken aşağıdaki bölümdeki bilgileri not edin.

### Adlandırılmış Değişmezler

Değişmezlerin adları, adın bir parçası olarak altçizgi karakteri ( \_ ) içerir. COBOL ' de, tire karakterini ( - ) alt çizgi karakterini yerine kullanmanız gerekir. Karakter dizilimi değerlerine sahip sabit değerler, dizilim sınırlayıcısı olarak tek tırnak işareti ( ' ) karakterini kullanır. Derleyicinin bu karakteri kabul etmesini sağlamak için, APHOST derleyici seçeneğini kullanın.

CMQV kopya dosyası, adlandırılmış değişmezlerin bildirimlerini level-10 öge olarak içerir. Sabitleri kullanmak için, level-01 ögesini açık bir şekilde bildirerek, değişmezlerin bildirimlerinde kopyalamak için COPY deyimini kullanın:

```
WORKING-STORAGE SECTION.  
01 MQM-CONSTANTS.  
COPY CMQV.
```

Ancak bu yöntem, değişmezler, başvuruda bulunulmamış olsa da programdaki depolamayı işgal etmeye neden olur. Sabit değerler aynı çalıştırma birimindeki birçok ayrı programa dahil ediliyorsa, sabitlerin birden çok kopyası bulunur; bu, kullanılan ana saklama alanının önemli bir miktarına neden olabilir. You can avoid this by adding the GLOBAL clause to the level-01 declaration:

```
* Declare a global structure to hold the constants  
01 MQM-CONSTANTS GLOBAL.  
COPY CMQV.
```

Bu işlem, çalıştırma birimi içinde yalnızca *bir* sabit disk kümesi için depolama ayırır; ancak, sabitler, yalnızca level-01 bildirimini içeren program değil, çalıştırma birimindeki *herhangi bir* program tarafından başvurulabilir.

### Yapı hizalamasının sağlanması

MQ çağrısının başlatılacak şekilde geçirilen IBM MQ yapılarının sözcük sınırlarında hizalanması için özen gösterilebilir. 32 bit işlemler için 4 byte, 64 bit işlemleri için 8 bayt ve 128 bit işlemler için 16 bayt ( IBM i ) için 4 byte.

Mümkün olduğunda, tüm IBM MQ yapılarını bir araya getirin, böylece tüm sınırları hizalanmış olur.

## System/390 çevirici dilinde kodlama (İleti kuyruğu arabirimi)

Note the information in the following sections when coding IBM MQ for z/OS programs in assembler language.

- “Adlar” sayfa 1089
- “MQI çağrılarının kullanılması” sayfa 1089
- “Değişmezlerin” sayfa 1089
- “Yapı adının belirtilmesi” sayfa 1089
- “Yapı formunun belirtilmesi” sayfa 1090
- “Listelemeyi denetleme” sayfa 1090
- “Alanlar için ilk değerleri belirtme” sayfa 1090



- “Yeniden girişlenebilir programlar yazılıyor” sayfa 1090
- “CEDF ' nin kullanılması” sayfa 1091

## Adlar

Çağrılarının tanımlarındaki değıştirtgelerin adları ve yapılara ilişkin açıklamalardaki alanların adları karışık olarak gösterilir. In the assembler-language macros supplied with IBM MQ, all names are in uppercase.

## MQI çağrılarının kullanılması

MQI bir çağrı arabirimidir, bu nedenle çevirici dili programları, işletim sistemi bağlantı kuralını gözetmelidir.

Özellikle, bir MQI çağrısı yayınlamadan önce, çevirici-dil programları en az 18 tam sözcük içeren bir saklama alanında R13 kaydını göstermelidir. Bu saklama alanı, çağrılan program için saklama alanı sağlar. Arayan kayıtlarını, içerikleri yok edilmeden önce saklar ve geri dönüşte çağırmanın kayıt dosyalarının içeriğini geri yükler.

**Not:** Bu, dinamik depolamalarını ayarlamak için DFHEIENT makrosunu kullanan, ancak varsayılan DATAREG 'yi R13 ' den diğer kayıt defterlerine geçersiz kılacak CICS assemblör dil programları için önemlidir. When the CICS Resource Manager Interface receives control from the stub, it saves the current contents of the registers at the address to which R13 is pointing. Bu amaçla bir saklama alanı ayırlamaması öngörülemez sonuçlar verir ve CICS' ta bir olağandışı sonda neden olur.

## Değişmezlerin

Sabitlerin çoğu, CMQA makrosu olarak bildirilir.

Ancak, aşağıdaki sabitler eş değer olarak tanımlanamaz ve makroyu varsayılan seçenekleri kullanarak çağırduğunuzda bu değerler içerilmezler:

- MQACT\_NONE
- MQCI\_NONE
- MQFMT\_NONE
- MQFMT\_ADMIN
- MQFMT\_COMMAND\_1
- MQFMT\_COMMAND\_2
- MQFMT\_DEAD\_LETTER\_HEADER
- MQFMT\_OLAY
- MQFMT\_IMS
- MQFMT\_IMS\_VAR\_STRING
- MQFMT\_PCF
- MQFMT\_STRING
- MQFMT\_TETIKLEYICISI
- MQFMT\_XMIT\_Q\_HEADER
- MQMI\_NONE

Bunları eklemek için, makroyu çağırduğunuzda EQUONLY=NO anahtar sözcüğünü ekleyin.

CMQA birden çok bildirgeye karşı korunuyor, bu nedenle bunu birçok kez içerebilirsiniz. Ancak EQUONLY anahtar sözcüğü yalnızca makronun ilk kez eklendiği anda yürürlüğe girer.

## Yapı adının belirtilmesi

Bir yapının birden çok eşgörünümünün bildirilmesine olanak sağlamak için, yapı örneklerini oluşturan makro, her alanın adını, kullanıcı özellebilir bir dizgiyle ve bir alt çizgi karakteri ( ) ekler.

Makroyu çağırduğunuzda dizgiyi belirtin. Bir 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_"
```

Arama açıklamaları içindeki yapı bildirimleri varsayılan öneki gösterir.

## Yapı formunun belirtilmesi

Makrolar, DSECT parametresi tarafından denetlenen iki biçimden birinde yapı bildirimleri üretebilir.

### DSECT = EVET

Bir çevirici dili DSECT yönergesi, yeni bir veri bölümü başlatmak için kullanılır; yapı tanımlaması, DSECT deyimini hemen izler. Herhangi bir saklama alanı ayrılmadı, bu nedenle kullanıma hazırlama olanaklı değil. Makro çağırımı üzerindeki etiket, veri bölümünün adı olarak kullanılır; etiket belirtilmediyse, yapının adı kullanılır.

### DSECT = NO

Çevirici dili DC yönergeleri, yordamı yordamda yürürlükteki konumda tanımlamak için kullanılır. Alanlar, makro çağırılırken ilgili değiştirgeleri kodlayarak belirleyebileceğiniz değerlerle kullanıma hazırlandı. Makro çağırımı üzerinde hiçbir değer belirtilmediği alanlar varsayılan değerlerle başlatılır.

DSECT = NO belirlendiyse, DSECT parametresi belirlenmez.

## Listelemeyi denetleme

Derleyici dili listelemesinde yapı bildiriminin görünüşünü LIST değiştirgesiyle denetleyebilirsiniz:

### LISTE = EVET

Yapı bildirimini, çevirici dili listelemesinde görünür.

### LISTE = HAYIR

Yapı bildirimini, çevirici dili listelemesinde görünmez. LISTE parametresi belirlenmezse, bu durum varsayılır.

## Alanlar için ilk değerleri belirtme

Bir alandaki bir alanı ilk kullanıma hazırlamak için kullanılacak değeri, makro çağırımında bir değiştirge olarak (önek olmadan), gereken değer yanında kodlayarak bir alanı kullanıma hazırlamak için kullanılacak değeri belirtebilirsiniz.

Örneğin, *MsgType* alanıyla, MQMT\_REQUEST ile ilk kullanıma hazırlanmış bir ileti tanımlayıcısı yapısını bildirmek için ve *ReplyToQ* alanı MY\_REPLY\_TO\_QUEUE dizgisiyle kullanıma hazırlansa, aşağıdaki kodu kullanın:

```
MY_MQMD      CMQMDA      MSGTYPE=MQMT_REQUEST,      X
REPLYTOQ=MY_REPLY_TO_QUEUE
```

Makro çağırısında değer olarak adlandırılmış bir değişmez (ya da eşitle) belirtirseniz, adı belirtilen değişmezi tanımlamak için CMQA makrosunu kullanın. Karakter dizilimleri olan tek tırnak (") değerleri içine girmemelisiniz.

## Yeniden girişlenebilir programlar yazılıyor

IBM MQ , hem giriş hem de çıkış için yapılarını kullanır. Programınızın yeniden kullanılabilir durumda kalmasını istiyorsanız:

1. Yapılara ilişkin çalışma depolama sürümlerini DSECT olarak tanımlayın ya da önceden tanımlı bir DSECT içinde yerleşik yapıları tanımlayın. Daha sonra, DSECT ' yi kullanarak elde edilen depolamaya koyulayın:

- Toplu ve TSO programları için, STORAGE ya da GETMAIN z/OS çevirici makroları
- CICS için, 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 depolama alanına kopyalamak için MVCL çevirici yönergesini kullanın.

2. CALL makrounun LIST formunu ( MF=L) kullanarak depolamada yer ayırın. When you use the CALL macro to make an MQI call, use the EXECUTE form ( MF=E) of the macro, using the storage reserved earlier, as shown in the example under “CEDF ' nin kullanılması” sayfa 1091. For more examples of how to do this, see the assembler language sample programs as shipped with IBM MQ.

Programınızın yeniden kullanıma uygun olup olmadığını belirlemenize yardımcı olması için çevirici dil RENT seçeneğini kullanın.

Yeniden birleştirilebilir programlar yazmaya ilişkin bilgi için bkz. [z/OS MVS Application Development Guide: Assembler Language Programs](#).

## CEDF ' nin kullanılması

If you want to use the CICS-supplied transaction, CEDF ( CICS Execution Diagnostic Facility) to help you to debug your program, add the , VL keyword to each CALL statement, for example:

```
CALL MQCONN, (NAME, HCONN, COMPCODE, REASON), MF=(E, PARMAREA), VL
```

The previous example is reenterable assembler-language code where PARMAREA is an area in the working storage that you specified.

## MQI çağrılarının kullanılması

MQI bir çağrı arabirimidir, bu nedenle çevirici dili programları, işletim sistemi bağlantı kuralını gözetmelidir. Özellikle, bir MQI çağrısı yayınlamadan önce, çevirici-dil programları en az 18 tam sözcük içeren bir saklama alanında R13 kaydını göstermelidir. Bu saklama alanı, çağrılan program için saklama alanı sağlar. Arayan kayıtlarını, içerikleri yok edilmeden önce saklar ve geri dönüşte çağırmanın kayıt dosyalarının içeriğini geri yükler.

**Not:** Bu, dinamik depolamalarını ayarlamak için DFHEIENT makrosunu kullanan, ancak varsayılan DATAREG 'yi R13 ' den diğer kayıt defterlerine geçersiz kılacak CICS assemblör dil programları için önemlidir. When the CICS Resource Manager Interface receives control from the stub, it saves the current contents of the registers at the address to which R13 is pointing. Bu amaçla uygun bir saklama alanı ayırmaması öngörülemeyen sonuçlar verir ve CICS için bir olağandışı sonda neden olur.

IBM i

## Coding IBM MQ programs in RPG (IBM i only)

IBM MQ belgelerinde, çağrılarının parametreleri, veri tiplerinin adları, yapı alanları ve değişmezlerin adları, uzun adlarıyla açıklanmıştır. RPG ' de bu adlar altı ya da daha az büyük harf karakteriyle kısaltılır.

For example, the field *MsgType* becomes *MDMT* in RPG. Daha fazla bilgi için [IBM i Application Programming Reference \(ILE/RPG\)](#) belgesine bakın.

## PL/I içinde kodlama (yalnızca z/OS )

PL/I içinde IBM MQ için kodlama sırasında yararlı bilgiler.

### Yapılar

Yapılar, BASED öznitelikli olarak bildirilir ve program bir yapının bir ya da daha fazla örneğini bildirmediği sürece hiçbir depolama alanını işgal etmez.

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 öznitelikle bildirilir; bir yapının eşgörünümünü bildirmek için like özniteliği kullanıldığında, bu eşgörünüm o yapı için tanımlanmış ilk değerleri devralır. Yalnızca gerekli değer başlangıç değerinden farklı olduğu alanları ayarlamanız gerekir.

PL/I, büyük ve küçük harfe duyarlı değildir; bu nedenle, çağrıların, yapı alanlarının ve sabitlerin adları küçük harfle, büyük harfle ya da büyük harfle kodlanabilir.

## Adlandırılmış Değişmezler

Adlandırılan değişmezler makro değişkenleri olarak bildirilir; sonuç olarak, program tarafından gönderme yapılan sabit değerler derlenen yordamda herhangi bir saklama alanı kaplamaz.

Ancak, kaynak programın derlendiğinde makro önışlemci tarafından işlenmesine neden olan derleyici seçeneği belirlenmelidir.

Tüm makro değişkenleri, sayısal değerleri temsil eden karakter değişkenleridir. Bu durum sezgisel olarak algılanabilir gibi görünse de, makro değişkenlerinin yerine makro işlemcisi değiştirildikten sonra veri tipi çakışmasına neden olmaz. Örneğin:

```
%dcl MQMD_STRUC_ID char;
%MQMD_STRUC_ID = ' 'MD ' ';

%dcl MQMD_VERSION_1 char;
%MQMD_VERSION_1 = '1';
```

## IBM MQ örnek yordamsal programlarının kullanılması


Bu örnek programlar yordamsal dillere yazılır ve İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımları gösterir. Farklı platformlardaki IBM MQ programları.

### Bu görev hakkında

İki örnek kümesi vardır:

- Dağıtılmış sistemlere ve IBM i' e ilişkin örnek programlar.
- z/OS için örnek programlar.

### Yordam

- Örnek programlar hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:
  - [“Örnek Programların çoklu Platformlar Üzerinde Kullanılması” sayfa 1093](#)
  -  [“z/OS için örnek programların kullanılması” sayfa 1193](#)

### İlgili kavramlar

[“Uygulama geliştirme kavramları” sayfa 7](#)

IBM MQ uygulamalarını yazmak için yordamsal ya da nesne yönelimli dil seçenekleri kullanabilirsiniz. IBM MQ uygulamalarınızı tasarlamaya ve yazmaya başlamadan önce, temel IBM MQ kavramlarına aşinalık edin.

[“IBM MQ için uygulama geliştirilmesi” sayfa 5](#)

İletileri göndermek ve almak, kuyruk yöneticilerinizi ve ilgili kaynaklarınızı yönetmek için uygulamalar geliştirebilirsiniz. IBM MQ , birçok farklı dil ve çerçeve içinde yazılmış uygulamaları destekler.

[“IBM MQ uygulamaları için dikkat edilmesi gereken noktalar” sayfa 46](#)

Uygulamalarınızın kullanımınıza sunulan platformlardan ve ortamlardan nasıl yararlanabileceğinize karar verdiğinizde, IBM MQ tarafından sunulan özelliklerin nasıl kullanılacağına karar vermeniz gerekir.

[“Kuyruğa alma için bir yordamsal uygulama yazma” sayfa 759](#)

Kuyruk yöneticisi, yayınlama/abone olma, nesnelere açma ve kapama nesnelere bağlanma ve bağlantıyı kesme, kuyruğa alma ve bağlantı kesme hakkında bilgi edinmek için bu bilgileri kullanın.

[“İstemci yordamsal uygulamaları yazılıyor” sayfa 945](#)

Bir yordamsal dil kullanarak IBM MQ üzerinde istemci uygulamaları yazmak için bilmeniz gerekenler.

[“Yayınlama/abone olma uygulamaları yazılıyor” sayfa 845](#)

Yayınlama/abone olma IBM MQ uygulamalarını yazmaya başlayın.

[“Yordamsal uygulama oluşturulması” sayfa 1030](#)

Bir IBM MQ uygulamasını birden çok yordamsal dilden birine yazabilir ve uygulamayı farklı platformlarda çalıştırabilirsiniz.

[“Yordamsal program hatalarının işlenmesi” sayfa 1073](#)

Bu bilgiler, bir çağrı yaparken ya da iletisi son hedefine teslim edildiğinde, uygulama MQI çağrılarınızla ilişkili hataları açıklar.

## **Multi Örnek Programların çoklu Platformlar Üzerinde Kullanılması**

Bu örnek yordamsal programlar ürünle birlikte teslim edilir. Örnekler, C ve COBOL 'de yazılır ve Message Queue Interface (MQI)' ın tipik kullanımları gösterir.

### **Bu görev hakkında**

Örnekler, genel programlama tekniklerini göstermek üzere tasarlanmadığından, bir üretim programına dahil etmek isteyebileceğiniz bazı hata denetimi atlanır.

Tüm örneklere ilişkin kaynak kodu ürünle birlikte sağlanır; bu kaynak, programlarda gösterilen ileti kuyruklama tekniklerini açıklayan yorumları içerir.

**IBM i** RPG programlaması için bkz. [IBM i Application Programming Reference \(ILE/RPG\)](#).

Örneklerin adları amqönekiyle başlar. Dördüncü karakter programlama dilini ve derleyiciyi gerekli durumlarda gösterir:

- s: C dili
- O: Hem IBM hem de Micro Focus derleyicilerindeki COBOL dili
- i: Yalnızca IBM derleyicilerindeki COBOL dili
- m: yalnızca Micro Focus derleyicilerindeki COBOL dili

Yürütülebilir dosyanın sekizinci karakteri, örneğin yerel bağlama kipinde mi, yoksa istemci kipinde mi çalıştırılacağını gösterir. Sekizinci karakter yoksa, örnek yerel bağ tanımları kipinde çalıştırılır. Sekizinci karakter ' c' ise, örnek istemci kipinde çalıştırılır.

Örnek uygulamaları çalıştırabilmeniz için önce bir kuyruk yöneticisi yaratmalı ve yapılandırmalısınız. Kuyruk yöneticisini istemci bağlantılarını kabul edecek şekilde ayarlamak için bkz. [“Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102](#).

### **Yordam**

- Örnek programlar hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:
  - [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#)
  - [“Yayınlama/Abone Olma örnek programları” sayfa 1131](#)
  - [“Put Sample programlar” sayfa 1136](#)
  - [“Dağıtım Listesi örnek programı” sayfa 1122](#)
  - [“Göz At örnek programları” sayfa 1111](#)

- [“Tarayıcı örnek programı” sayfa 1112](#)
- [“Get Sample programlar” sayfa 1124](#)
- [“Reference Message Sample programlar” sayfa 1138](#)
- [“İstek örnek programları” sayfa 1145](#)
- [“Sorma örnek programları” sayfa 1130](#)
- [“Message Handle örnek programının Sorgusu Özellikleri” sayfa 1131](#)
- [“Örnek programları ayarla” sayfa 1150](#)
- [“Echo örnek programları” sayfa 1123](#)
- [“Data-Conversion örnek programı” sayfa 1115](#)
- [“Tetikleme örnek programları” sayfa 1154](#)
- [“Zamanuyumsuz Koy örnek programı” sayfa 1110](#)
- [“Veritabanı koordinasyonu örnekleri” sayfa 1115](#)
- [“CICS hareket örneği” sayfa 1113](#)
- [“UNIX ve Windowsüzerinde SMOKIN örneklerini kullanma” sayfa 1156](#)
- [“Ölü-harfli kuyruk işleyicisi örneği” sayfa 1122](#)
- [“Connect örnek programı” sayfa 1113](#)
- [“API çıkış örnek programı” sayfa 1108](#)
- [“Windowsüzerindeki SSPI güvenlik çıkışısının kullanılması” sayfa 1167](#)
- [“Uzak kuyruklar kullanarak örnekleri çalıştırma” sayfa 1168](#)
- [“Küme Kuyruğu İzleme örnek programı \(AMQSCLM\)” sayfa 1168](#)
- [“Bağlantı Uç Noktası Araması örnek programı \(CEPL\)” sayfa 1178](#)

### İlgili kavramlar

[“C++ örnek programları” sayfa 503](#)

İletileri almayı ve yerleştirmeyi göstermek için dört örnek program sağlanır.

### İlgili görevler

[“z/OSiçin örnek programların kullanılması” sayfa 1193](#)

IBM MQ for z/OS ile teslim edilen örnek yordamsal uygulamalar, İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımları göstermektedir.

### **Multi** **Örnek Programlar On Multiplatforms içinde gösterilen özellikler**

IBM MQ örnek programları tarafından gösterilen teknikleri gösteren tablolardan oluşan bir derlem.

MQOPER ve MQCLOSE çağrılarını kullanan tüm örnekler açık ve kapalı kuyruklar, bu nedenle bu teknikler tablolarda ayrı olarak listelenmez. İlgilendiğiniz platformu içeren başlığa bakın.

**z/OS** z/OS platformu için bkz. [“z/OSiçin örnek programların kullanılması” sayfa 1193.](#)

**Linux** **UNIX** *UNIX and Linux sistemleri için örnekler*

UNIX and Linux sistemlerinde IBM MQ için örnek programlar tarafından gösterilen teknikler.

See [“Örnek programların UNIX and Linuxüzerinde hazırlanması ve çalıştırılması” sayfa 1105](#) to find out where the sample programs for IBM MQ on UNIX and Linux systems are stored.

Çizelge 163 [sayfa 1095](#) Tablo, hangi C ve COBOL kaynak dosyalarının sağlanıp sağlanmadığını ve bir sunucunun ya da istemci yürütülebilir dosyasının dahil edilip edilip eklenmeyeceğini listeler.

Çizelge 163. Sample programs that demonstrate the use of the MQI (C and COBOL) on UNIX and Linux.

Dört sütuna sahip bir tablo. İlk kolonlar, örneklerin gösterdiği teknikleri listeler. İkinci sütunda C örnekleri ve üçüncü sütunda, birinci sütunda listelenen tekniklerin her birini gösteren COBOL örnekleri listelenir. Dördüncü sütun, bir sunucu C yürütülebilir dosyasının içerilip içerilmeyeceğini ve beşinci sütunun bir istemci C yürütülebilir dosyasının dahil edilip edilmediğini gösterir.

<b>Teknik</b>	<b>C (kaynak) (“1” sayfa 1097)</b>	<b>COBOL (kaynak) (“2” sayfa 1097)</b>	<b>Sunucu (C yürütülür dosyası)</b>	<b>İstemci (C yürütülür dosyası)</b>
Yayınlama/abone olma arabirimini kullanma	amqspuba amqssuba amqssbxa	örnek yok	amqspub amqssub amqssbx	örnek yok
MQPUT çağrısını kullanarak ileti koyma	amqsput0	amq0put0	amqsput	amqsputc
MQPUT1 çağrısını kullanarak tek bir ileti koyma	amqsinqa amqsecha	amqminqx amqmechx amqiinqx amqiechx	amqsinq amqsech	amqsechc
Dağıtım listesine ileti koyma (“3” sayfa 1097)	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc
İstek iletisi yanıtlama	amqsinqa	amqminqx amqiinqx	amqsinq	örnek yok
Göz atma kullanılarak ileti alınması (bekleme yok)	amqsgbr0	amq0gbr0	amqsgbr	örnek yok
İletileri alma (bir süre sınırlaması ile bekleme)	amqsget0	amq0get0	amqsget	amqsgetc
İletileri alma (sınırsız bekleme)	amqstrg0	örnek yok	amqstrg	amqstrgc
İletileri alma (veri dönüştürme ile)	amqsecha	örnek yok	amqsech	örnek yok
Bir Kuyruğa Başvuru İletileri Koyma (“3” sayfa 1097)	amqsprma	örnek yok	amqsprm	amqsprmc
Kuyruktan Başvuru İletileri Alınması (“3” sayfa 1097)	amqsgrma	örnek yok	amqsgrm	amqsgrmc
Başvuru İletisi kanalı çıkışı (“3” sayfa 1097)	amqsqrma amqsxrma	örnek yok	amqsxrm	örnek yok
İletin ilk 20 karakterine göz atma	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Tüm iletilere göz atma	amqsbcg0	örnek yok	amqsbcg	amqsbcgc
Paylaşılan giriş kuyruğunun kullanılması	amqsinqa	amqminqx amqiinqx	amqsinq	amqsinqc
Dışlayıcı bir giriş kuyruğunu kullanma	amqstrg0	amq0req0	amqstrg	amqstrgc
MQINQ çağrısının kullanılması	amqsinqa	amqminqx amqiinqx	amqsinq	örnek yok
MQSET çağrısının kullanılması	amqsseta	amqmsetx amqisetx	amqsset	amqssetc
Yanıtlama Kuyruğu Kullanılması	amqsreq0	amq0req0	amqsreq	amqsreqc
İleti kural dışı durumları isteniyor	amqsreq0	amq0req0	amqsreq	örnek yok
Kesilmiş bir iletinin kabul edilmesi	amqsgbr0	amq0gbr0	amqsgbr	örnek yok
Çözülmüş bir kuyruk adının kullanılması	amqsgbr0	amq0gbr0	amqsgbr	örnek yok

Çizelge 163. Sample programs that demonstrate the use of the MQI (C and COBOL) on UNIX and Linux.

Dört sütuna sahip bir tablo. İlk kolonlar, örneklerin gösterdiği teknikleri listeler. İkinci sütunda C örnekleri ve üçüncü sütunda, birinci sütunda listelenen tekniklerin her birini gösteren COBOL örnekleri listelenir. Dördüncü sütun, bir sunucu C yürütülebilir dosyasının içerilip içerilmeyeceğini ve beşinci sütunun bir istemci C yürütülebilir dosyasının dahil edilip edilmediğini gösterir.

(devamı var)

<b>Teknik</b>	<b>C (kaynak) (“1” sayfa 1097)</b>	<b>COBOL (kaynak) (“2” sayfa 1097)</b>	<b>Sunucu (C yürütülür dosyası)</b>	<b>İstemci (C yürütülür dosyası)</b>
Süreci tetikleme	amqstrg0	örnek yok	amqstrg	amqstrgc
Veri dönüştürmenin kullanılması	(“4” sayfa 1097)	örnek yok	örnek yok	örnek yok
IBM MQ (XA uyumlu veritabanı yöneticilerini koordine etmek), SQL kullanarak tek bir veritabanına erişilmesi	amqsxas0.sqc Db2 amqsxas0.ec Informix	amq0xas0.sq b	örnek yok	örnek yok
IBM MQ (XA uyumlu veritabanı yöneticilerinin eşgüdümü) SQL kullanarak iki veritabanına erişilmesi	amqsxag0.c amqsxab0.sq c amqsxaf0.sqc	amq0xag0.cbl amq0xab0.sq b amq0xaf0.sqb	örnek yok	örnek yok
CICS işlemi (“5” sayfa 1097)	amqscic0.ccs	örnek yok	amqscic0	örnek yok
Encina işlemi (“3” sayfa 1097)	amqsxae0	örnek yok	amqsxae0	örnek yok
İletileri yerleştirmek için SMOKIN işlemi (“6” sayfa 1097)	amqstxpx	örnek yok	örnek yok	örnek yok
İletileri almak için SMOKIN işlemi (“6” sayfa 1097)	amqstxgx	örnek yok	örnek yok	örnek yok
SMOKIN sunucusu (“6” sayfa 1097)	amqstxsx	örnek yok	örnek yok	örnek yok
Kuyruk-harf kuyruğu işleyicisi	Dizin ./ tools/c/ Samples/dl q (“7” sayfa 1097)	örnek yok	amqsdldq	örnek yok
Bir MQI istemcisinden bir ileti yerleştirerek	örnek yok	örnek yok	örnek yok	amqsputc
Bir MQI istemcisinden ileti alma	örnek yok	örnek yok	örnek yok	amqsgetc
MQCONNX kullanılarak kuyruk yöneticisiyle bağlantı kuruluyor	amqscnxc	örnek yok	örnek yok	amqscnxc
API çıkışlarının kullanılması	amqsaxe0	örnek yok	amqsaxe	örnek yok
Küme iş yükü dengeleme çıkışı	amqswlm0	örnek yok	amqswlm	örnek yok
MQSTAT çağrısını kullanarak iletileri zamanuyumsuz olarak alma ve durum alma	amqsapt0	örnek yok	amqsapt	amqsaptc
Yeniden bağlanabilir istemciler	amqsphac amqsghac amqsmhac	örnek yok	geçerli değil	amqsphac amqsghac amqsmhac



Çizelge 163. Sample programs that demonstrate the use of the MQI (C and COBOL) on UNIX and Linux.

Dört sütuna sahip bir tablo. İlk kolonlar, örneklerin gösterdiği teknikleri listeler. İkinci sütunda C örnekleri ve üçüncü sütunda, birinci sütunda listelenen tekniklerin her birini gösteren COBOL örnekleri listelenir. Dördüncü sütun, bir sunucu C yürütülebilir dosyasının içerilip içerilmeyeceğini ve beşinci sütunun bir istemci C yürütülebilir dosyasının dahil edilip edilmediğini gösterir.

(devamı var)

Teknik	C (kaynak) ("1" sayfa 1097)	COBOL (kaynak) ("2" sayfa 1097)	Sunucu (C yürütülür dosyası)	İstemci (C yürütülür dosyası)
Birden çok kuyruktan iletileri zamanuyumsuz olarak tüketmek için ileti tüketicilerinin kullanılması	amqscbf0	örnek yok	amqscbf	amqscbfc
MQCONNX üzerinde 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 derleyicisi ile 'amqm' başlayan programları, IBM COBOL derleyicisiyle başlayan 'amqi' ve 'amq0' başlangıçlarıyla başlayan programları derleyin.
3. **UNIX** Yalnızca IBM MQ for AIX ve IBM MQ for Solaris üzerinde desteklenir.
4. **UNIX** On IBM MQ for AIX, ve IBM MQ for Solaris this program is called amqsvfc0.c
5. **AIX** CICS , yalnızca IBM MQ for AIX tarafından desteklenir.
6. **Linux** SMOKIN, System püzerinde Linux için IBM MQ tarafından desteklenmiyor.
7. Ölü-harfli kuyruk işleyicisi için kaynak birkaç dosyadan oluşur ve ayrı bir dizinde sağlanır.

UNIX and Linux sistemlerine ilişkin destek hakkında ayrıntılı bilgi için bkz. [IBM MQ](#).

#### **Windows** IBM MQ for Windows için örnekler

IBM MQ for Windows için örnek programlar tarafından gösterilen teknikler.

Çizelge 164 sayfa 1097 , hangi C ve COBOL kaynak dosyalarının sağlanıp sağlanmadığını ve bir sunucunun ya da istemci yürütülebilir dosyasının dahil edilip etmeyeceğini listeler.

Çizelge 164. 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
MQPUT çağrısını kullanarak ileti koyma	amqsput0	amq0put0	amqsput	amqsputc
MQPUT1 çağrısını kullanarak tek bir ileti koyma	amqsinqa amqsecha	amqminq2 amqmech2 amqiinq2 amqiech2	amqsinq amqsech	amqsinqc amqsechc
İletileri Dağıtım Listesine Koyma	amqsptl0	amq0ptl0.cbl	amqsptl	amqsptlc

Çizelge 164. 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 iletisi yanıtlama	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
İleti alınıyor (bekleme yok)	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
İletileri alma (bir süre sınırlaması ile bekleme)	amqsget0	amq0get0	amqsget	amqsgetc
İletileri alma (sınırsız bekleme)	amqstrg0	örnek yok	amqstrg	amqstrgc
İletileri alma (veri dönüştürme ile)	amqsecha	örnek yok	amqsech	amqsechc
Başvuru İletilerini Kuyruğa Koyma	amqsprma	örnek yok	amqsprm	amqsprmc
Kuyruktan Başvuru İletileri Alınması	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
Tüm iletilere göz atma	amqsbcg0	örnek yok	amqsbcg	amqsbcgc
Paylaşılan giriş kuyruğunun kullanılması	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
Dışlayıcı bir giriş kuyruğunu kullanma	amqstrg0	amq0req0	amqstrg	amqstrgc
MQINQ çağrısının kullanılması	amqsinqa	amqminq2 amqiinq2	amqsinq	amqsinqc
MQSET çağrısının kullanılması	amqsseta	amqmset2 amqiset2	amqsset	amqssetc
MQINQMP çağrısının kullanılması	amqsiqma	örnek yok	örnek yok	örnek yok
Yanıtlama Kuyruğu Kullanılması	amqsreq0	amq0req0	amqsreq	amqsreqc
İleti kural dışı durumları isteniyor	amqsreq0	amq0req0	amqsreq	amqsreqc
Kesilmiş bir iletinin kabul edilmesi	amqsgbr0	amq0gbr0	amqsgbr	amqsgbrc
Çözülmüş bir 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 yöneticilerini koordine etmek), SQL kullanarak tek bir veritabanına erişilmesi	amqsxas0.sqc Db2 amqsxas0.ec Informix	amq0xas0.sq b	örnek yok	örnek yok
IBM MQ (XA uyumlu veritabanı yöneticilerinin eşgüdümü) SQL kullanarak iki veritabanına erişilmesi	amqsxag0.c amqsxab0.sq c Db2 amqsxaf0.sqc Db2	amq0xag0.cbl amq0xab0.sq b amq0xaf0.sqb	örnek yok	örnek yok

Çizelge 164. 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 SMOKIN işlemi	amqstpxx	örnek yok	örnek yok	örnek yok
İletileri almak için SMOKIN işlemi	amqstxgx	örnek yok	örnek yok	örnek yok
SMOKIN sunucusu	amqstxsx	örnek yok	örnek yok	örnek yok
Kuyruk-harf kuyruğu işleyicisi	Dizin ./ tools/c/ Samples/dl q ("1" sayfa 1099)	örnek yok	amqsdldq	örnek yok
From an IBM MQ MQI client, putting a message	örnek yok	örnek yok	örnek yok	amqsputc
IBM MQ MQI client' dan bir ileti alma	örnek yok	örnek yok	örnek yok	amqsgetc
MQCONN kullanılarak kuyruk yöneticisiyle bağlantı kuruluyor	amqscnxc	örnek yok	örnek yok	amqscnxc
API çıkışlarının kullanılması	amqsaxe0	örnek yok	amqsaxe	örnek yok
Küme iş yükü dengeleme	amqswlm0	örnek yok	amqswlm	örnek yok
SSPI güvenlik yordamları	amqsspin	örnek yok	amqrspin.dll	amqrspin.dll
MQSTAT çağrısını kullanarak iletileri zamanuysuz olarak alma ve durum alma	amqsapt0	örnek yok	amqsapt	amqsaptc
Yeniden bağlanabilir istemciler	amqsphac amqsghac amqsmhac	örnek yok	Burada geçerli değil	amqsphac amqsghac amqsmhac
Birden çok kuyruktan iletileri zamanuysuz olarak tüketmek için ileti tüketicilerinin kullanılması	amqscbf0	örnek yok	amqscbf	amqscbfc
MQCONN üzerinde TLS bağlantı bilgilerinin belirtilmesi	amqssslc	örnek yok	geçerli değil	amqssslc

#### Notlar:

1. Ölü-harfli kuyruk işleyicisi için kaynak birkaç dosyadan oluşur ve ayrı bir dizinde sağlanır.

#### **Windows** Visual Basic samples for IBM MQ for Windows

Windows sistemlerinde IBM MQ için örnek programlar tarafından gösterilen teknikler.

Çizelge 165 sayfa 1100 , IBM MQ for Windows örnek programları tarafından gösterilen teknikleri gösterir.

Bir proje birkaç dosya içerebilir. Visual Basic içinde bir proje açmanken, diğer ktklerin otomatik olarak yklenir. Yürütülebilir bir program programı sağlanmaz.

mqrtrvc.vbpdışında tüm örnek projeler IBM MQ Server ile çalışacak şekilde ayarlanır. To find out how to change the sample projects to work with the IBM MQ clients see ["Preparing Visual Basic programs in Windows"](#) sayfa 1056.

Çizelge 165. MQI ' nin (Visual Basic) kullanımını gösteren IBM MQ for Windows örnek programları	
Teknik	Proje dosyası adı
MQPUT çağrısını kullanarak ileti koyma	amqspub.vbp
MQGET çağrısını kullanarak ileti alınması	amqsgetb.vbp
MQGET çağrısını kullanarak kuyruğa göz atma	amqsbcgb.vbp
Basit MQGET ve MQPUT örneği (istemci)	mqrtrvc.vbp
Basit MQGET ve MQPUT örneği (sunucu)	mqrtrivs.vbp
Dizgileri ve kullanıcı tanımlı yapıları MQPUT ve MQGET kullanarak koyma ve alma	strings.vbp
Bir kanalı başlatmak ve durdurmak için PCF yapılarının kullanılması	pcfsamp.vbp
MQAI kullanılarak kuyruk yaratılması	amqsaicq.vbp
MQAI kullanarak kuyruk yöneticisinin kuyrukları listeleniyor	amqsailq.vbp
MQAI kullanarak olayları izleme	amqsaiem.vbp

### IBM i IBM için örnekler

IBM i sistemlerinde IBM MQ için örnek programlar tarafından gösterilen teknikler.

Çizelge 166 sayfa 1100 , IBM MQ for IBM i örnek programları tarafından gösterilen teknikleri gösterir. Bazı teknikler birden çok örnek programda ortaya çıkar, ancak çizelgede yalnızca bir program listelenir.

Çizelge 166. Sample programs demonstrating use of the MQI (C and COBOL) on IBM i				
Teknik	C (kaynak) ( "1" sayfa 1101 )	COBOL (kaynak) ( "2" sayfa 1101 )	RPG (kaynak) ( "3" sayfa 1102 )	İstemci (C yürütülür dosyası) (4)
MQPUT çağrısını kullanarak ileti koyma	AMQSPUT0	AMQ0PUT4	AMQ3PUT4	AMQSPUTC
MQPUT çağrısını kullanarak bir veri dosyasından ileti konması	AMQSPUT4	örnek yok	örnek yok	örnek yok
MQPUT1 çağrısını kullanarak tek bir ileti koyma	AMQSINQ4, AMQSECH4	AMQ0INQ4, AMQ0ECH4	AMQ3INQ4, AMQ3ECH4	AMQSINQC, AMQSECHC
İletileri Dağıtım Listesine Koyma	AMQSPTL4	örnek yok	örnek yok	AMQSPTLC
İstek iletisi yanıtlama	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
İleti alınıyor (bekleme yok)	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
İletileri alma (bir süre sınırlaması ile bekleme)	AMQSGET4	AMQ0GET4	AMQ3GET4	AMQSGETC
İletileri alma (sınırsız bekleme)	AMQSTRG4	örnek yok	AMQ3TRG4	AMQSTRGC
İletileri alma (veri dönüştürme ile)	AMQSECH4	AMQ0ECH4	AMQ3ECH4	AMQSECHC
Başvuru İletilerini Kuyruğa Koyma	AMQSPRM4	örnek yok	örnek yok	AMQSPRMC
Kuyruktan Başvuru İletileri Alınması	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
İletinin ilk 49 karakterine göz atma	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC

Çizelge 166. Sample programs demonstrating use of the MQI (C and COBOL) on IBM i (devamı var)

Teknik	C (kaynak) ( "1" sayfa 1101 )	COBOL (kaynak) ( "2" sayfa 1101 )	RPQ (kaynak) ( "3" sayfa 1102 )	İstemci (C yürütülür dosyası) (4)
Tüm iletilere göz atma	AMQSBCG4	örnek yok	örnek yok	AMQSBCGC
Paylaşılan giriş kuyruğunun kullanılması	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
Dışlayıcı bir giriş kuyruğunu kullanma	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQC
MQINQ çağrısının kullanılması	AMQSINQ4	AMQ0INQ4	AMQ3INQ4	AMQSINQC
MQSET çağrısının kullanılması	AMQSSET4	AMQ0SET4	AMQ3SET4	AMQSSETC
Yanıtlama Kuyruğu Kullanılması	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQC
İleti kural dışı durumları isteniyor	AMQSREQ4	AMQ0REQ4	AMQ3REQ4	AMQSREQC
Kesilmiş bir iletinin kabul edilmesi	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Çözülmüş bir kuyruk adının kullanılması	AMQSGBR4	AMQ0GBR4	AMQ3GBR4	AMQSGBRC
Süreci tetikleme	AMQSTRG4	örnek yok	AMQ3TRG4	AMQSTRGC
Tetikleyici sunucusu	AMQSERV4	örnek yok	AMQ3SRV4	örnek yok
Tetikleme sunucusunun kullanılması ( CICS işlemleri de içinde olmak üzere)	AMQSERV4	örnek yok	AMQ3SRV4	örnek yok
Veri dönüştürmenin kullanılması	AMQSVFC4	örnek yok	örnek yok	örnek yok
API çıkışlarının kullanılması	AMQSAXE0	örnek yok	örnek yok	örnek yok
Küme iş yükü dengeleme	AMQSWLM0	örnek yok	örnek yok	örnek yok
MQSTAT çağrısını kullanarak iletileri zamanuyumsuz olarak alma ve durum alma	AMQSAPT0	örnek yok	örnek yok	AMQSAPTC
Yayınlama/abone olma arabirimini kullanma	AMQSPUBA, AMQSSUBA, AMQSSBXA	örnek yok	örnek yok	AMQSPUBC, AMQSSUBC, AMQSSBXC
Yeniden bağlantılanabilir istemciler (5)	AMQSPHAC, AMQSGHAC, AMQSMHAC	örnek yok	örnek yok	örnek yok
Birden çok kuyruktan iletileri zamanuyumsuz olarak tüketmek için ileti tüketicilerinin kullanılması (5)	AMQSCBFO	örnek yok	örnek yok	örnek yok
MQCONNX üzerinde TLS bağlantı bilgilerinin belirtilmesi	AMQSSSLC	örnek yok	örnek yok	AMQSSSLC
MQCONNX kullanılarak kuyruk yöneticisiyle bağlantı kuruluyor	AMQSCNXC	örnek yok	örnek yok	AMQSCNXC

**Notlar:**

1. C örneklerine ilişkin kaynak QMQMSAMP/QCSRC dosyasında yer alıyor. Dosyaları, QMQM/H dosyasına üye olarak varlar.
2. COBOL örnekleri kaynağı QMQMSAMP/QCBLLSRC dosyalarında yer alır. Üyeler, AMQ0 xxx 4 adını taşır; burada xxx , örnek işlevi belirtir.

3. RPG örnekleri için kaynak QMQMSAMP/QRPGLESRC ' de yer alıyor. Üyeler, AMQ3 xxx 4 adını taşır; burada xxx , örnek işlevi belirtir. Kopyalama üyeleri QMQM/QRPGLESRC içinde var olabilir. Her üye adı, G sonetine sahiptir.
4. IBM MQ MQI client örneklerinin yürütülebilir sürümü, bir sunucu ortamında çalışan örneklerle aynı kaynağı paylaşır. İstemci ortamındaki örneklerin kaynağı, sunucuyla aynı. IBM MQ MQI client örnekleri, LIBMQIC istemci kitaplığı ve IBM MQ sunucu örnekleri ile bağlantılı olduğundan, LIBMQM sunucu kitaplığı ile bağlantılandırılır.
5. Reconnectable istemcisinin örnek uygulaması için istemci yürütülebilir dosyası ve zamanuyumsuz olarak tüketici uygulamasının çalıştırılması gerekiyorsa, bu, LIBMQIC\_R adlı iş parçacıklı kitaplıkla derlenmesi ve bağlanması gerekir. Bu nedenle, iş parçacıklı ortamda çalıştırılmalıdır. QIBM\_MULTI\_YIVCID ortam değişkenini 'Y' olarak ayarlayın ve uygulamayı qsh ' den çalıştırın.

Ek bilgi için [IBM MQ olanağının Java ve JMS ile kurulması](#) başlıklı konuya bakın.

Bunlara ek olarak, IBM MQ for IBM i örnek seçeneği, örnek programlar, AMQSDATA ve yönetim görevlerini gösteren örnek CL programları için giriş olarak kullandığınız bir örnek veri dosyası içerir. CL örnekleri, [IBM uygulamasını yönetme](#) içinde açıklanmaktadır. Bu konuda açıklanan örnek programlarla kullanmak üzere kuyruklar yaratmak için amqsamp4 örnek CL programını kullanabilirsiniz.

## IBM i Windows UNIX Örnek programların hazırlanması ve çalıştırılması

İlk hazırlık işlemini tamamladıktan sonra, örnek programları çalıştırabilirsiniz.

### Bu görev hakkında

Örnek programları çalıştırmadan önce, önce bir kuyruk yöneticisi yaratmalı ve gerek duyduğunuz kuyrukları da yaratmalısınız. COBOL örneklerini çalıştırmak istiyorsanız, örneğin ek hazırlık yapmanız da gerekebilir. Gerekli hazırlık işlemini tamamladıktan sonra, örnek programları çalıştırabilirsiniz.

### Yordam

Örnek programların hazırlanmasına ve çalıştırılmasına ilişkin bilgi edinmek için aşağıdaki konulara bakın:

- “[Örnek programların IBM üzerinde hazırlanması ve çalıştırılması](#)” sayfa 1104
- “[Örnek programların UNIX and Linux üzerinde hazırlanması ve çalıştırılması](#)” sayfa 1105
- “[Örnek programların Windows üzerinde hazırlanması ve çalıştırılması](#)” sayfa 1107

## IBM i Windows UNIX Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi 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 etmek için yapılandırabilirsiniz.

### Başlamadan önce

Kuyruk yöneticisinin zaten var olduğundan ve başlatıldığından emin olun. Kanal doğrulama kayıtlarının, MQSC komutu vererek önceden etkinleştirilmiş olup olmadığını saptayın:

```
DISPLAY QMGR CHLAUTH
```

**Önemli:** Bu görev, kanal doğrulama kayıtlarının etkinleştirilmesini bekler. Bu, diğer kullanıcılar ve uygulamalar tarafından kullanılan bir kuyruk yöneticisiyse, bu ayarın değiştirilmesi diğer tüm kullanıcıları ve uygulamaları etkiler. If your queue manager does not make use of channel authentication records then step 4 can be replaced with an alternate authentication method (for example, a security exit) which sets the MCAUSER to the *ayrıcaksız-kullanıcı kimliği* you will obtain in step “1” sayfa 1103.

Uygulamanızın hangi kanal adını kullanmayı beklediğini bilmeniz gerekir. Böylece, uygulamanın kanalı kullanmasına izin verilmelidir. Ayrıca, hangi nesnelere, örneğin kuyruklara ya da konulara için uygulamanızın

kullanılmasını beklediğinden, uygulamanızın bunları kullanmasına izin verilebilmesi için izin verileceğini de bilmeniz gerekir.

## Bu görev hakkında

Bu görev, kuyruk yöneticisine bağlanan bir istemci uygulaması için kullanılmak üzere ayrıcalıklı olmayan bir kullanıcı kimliği yaratır. İstemci uygulaması için erişim izni verilir; yalnızca, gereken kanalı ve bu kullanıcı kimliğini kullanarak gereksinim duyduğu kuyruğu kullanabilmelidir.

## Yordam

1. Kuyruk yöneticinizin çalışmakta olduğu sistemde bir kullanıcı kimliği edinin. Bu görev için bu kullanıcı kimliği ayrıcalıklı bir yönetimle görevli kullanıcı olmamalıdır. Bu kullanıcı kimliği, istemci bağlantısının kuyruk yöneticisi üzerinde çalışacağı yetki olacaktır.
2. Aşağıdaki komutlarla bir dinleyici programı başlatın:

*qmgr-name* , kuyruk yöneticinizin adıdır  
*nnnn* , seçtiğiniz kapı numarasıdır

a) 

UNIX ve Windows sistemleri için:

```
runmqclsr -t tcp -m qmgr-name -p nnnn
```

b) 

IBM için:

```
STRMQMLSR MQMNAME(qmgr-name) PORT(nnnn)
```

3. Uygulamanızın SYSTEM.DEF.SVRCONN daha sonra bu kanal önceden tanımlıdır. Uygulamanız başka bir kanal kullanıyorsa, MQSC komutunu kullanarak bu kanalı yaratın:

```
DEFINE CHANNEL(' channel-name ') CHLTYPE(SVRCONN) TRPTYPE(TCP) +  
DESCR('Channel for use by sample programs')
```

*kanal-adi* , kanalınızın adıdır.

4. İstemci sisteminizin yalnızca IP adresinin, bu kanalı kullanarak, MQSC komutunu vererek kanal kullanmasını sağlayacak bir kanal doğrulama kuralı yaratın:

```
SET CHLAUTH(' channel-name ') TYPE(ADDRESSMAP) ADDRESS(' client-machine-IP-address ') +  
MCAUSER(' non-privileged-user-id ')
```

*kanal-adi* , kanalınızın adıdır.

*client-machine-IP-address* , istemci sisteminizin IP adresidir.

Örnek istemci uygulamanız kuyruk yöneticisiyle aynı makinede çalıştırılıyorsa, uygulamanız 'localhost' ile bağlantı kuracaksa '127.0.0.1' IP adresini kullanın. Birden çok farklı istemci makinesi bağlanacaksa, tek bir IP adresi yerine bir kalıp ya da aralık kullanabilirsiniz. Ayrıntılı bilgi için [Soysal IP adresleri](#) başlıklı konuya bakın.

*ayrıcalıklı olmayan-kullanıcı-kimliği* , “1” sayfa 1103adımında edindiğiniz kullanıcı kimliğidir.


5. Uygulamanızın SYSTEM.DEFAULT.LOCAL.QUEUE (Kuyruk), bu kuyruk zaten tanımlı. Uygulamanızın başka bir kuyruk kullanıyorsa, MQSC komutunu vererek yaratın:

```
DEFINE QLOCAL(' queue-name ') DESCR('Queue for use by sample programs')
```

*kuyruk-adi* , kuyruğunuzun adıdır.

6. Kuyruk yöneticisine bağlanmak ve sorgulamak için erişim izni verin:

a)   

 IBM i, UNIX ve Windows sistemleri için, MQSC komutlarını yayınlayın:

```
SET AUTHREC OBJTYPE(QMGR) PRINCIPAL(' non-privileged-user-id ') +  
AUTHADD(CONNECT, INQ)
```

*ayrıcalklı olmayan-kullanıcı-kimliği* , “1” sayfa 1103adımında edindiğiniz kullanıcı kimliğidir.

7. Uygulamanız noktadan noktaya iletişim uygulamasıysa, bu, kuyrukların kullanılmasını sağlar ve MQSC komutları vererek, kullanılacak kullanıcı kimliği tarafından kuyruğunuzu kullanarak kuyruğa alma ve iletileri alma ve iletileri alma izni verir.

a)   

 IBM i, UNIX ve Windows sistemleri için, MQSC komutlarını yayınlayın:

```
SET AUTHREC PROFILE(' queue-name ') OBJTYPE(Queue) +  
PRINCIPAL(' non-privileged-user-id ') AUTHADD(PUT, GET, INQ, BROWSE)
```

*kuyruk-adi* , kuyruğunuzun adıdır.

*ayrıcalklı olmayan-kullanıcı-kimliği* , “1” sayfa 1103adımında edindiğiniz kullanıcı kimliğidir.

8. Uygulamanız bir yayınlama/abone olma uygulamasıysa, bu, konuların kullanılmasını sağlar, MQSC komutları vererek, kullanılacak kullanıcı kimliği ile konularınızı kullanarak yayınlamaya ve abone olmaya izin verir.

a)   

 IBM i, UNIX ve Windows sistemleri için, MQSC komutlarını yayınlayın:

```
SET AUTHREC PROFILE(' SYSTEM.BASE.TOPIC ') OBJTYPE(TOPIC) +  
PRINCIPAL(' non-privileged-user-id ') AUTHADD(PUB, SUB)
```

*ayrıcalklı olmayan-kullanıcı-kimliği* , “1” sayfa 1103adımında edindiğiniz kullanıcı kimliğidir.

Bu işlem, konu ağacındaki herhangi bir konuya *ayrıcalksız-kullanıcı-kimliği* erişimi verir; diğer bir seçenek olarak, **DEFINE TOPIC** kullanarak bir konu nesnesi tanımlayabilir ve yalnızca o konu nesnesinin gönderme yaptığı konu ağacının bir bölümüne erişimler verir. Ayrıntılı bilgi için [Konulara kullanıcı erişiminin denetlenmesi](#) başlıklı konuya bakın.

## Sonraki adım

Artık istemci uygulamanız kuyruk yöneticisine bağlanabilir ve kuyruğu kullanarak ileti alabilir ya da alabilir.

### İlgili kavramlar

 [IBM üzerindeki IBM MQ yetkiler](#)

### İlgili görevler

[UNIX ya da Linux sistemlerinde bir IBM MQ nesnesine erişim verilmesi ve Windows](#)


### İlgili başvurular

[CHLAUTH KÜMESİ](#)

[KANAL TANIMLA](#)

[QLOCAL ' I TANIMLA](#)

[AUTHREC](#)

 [Örnek programların IBM üzerinde hazırlanması ve çalıştırılması](#)

Örnek programları IBM i' ta çalıştırmadan önce, önce bir kuyruk yöneticisi yaratmalı ve gerek duyduğunuz kuyrukları da yaratmalısınız. COBOL örneklerini çalıştırmak istiyorsanız, biraz ek hazırlık yapmanız gerekebilir.



## Bu görev hakkında

IBM MQ for IBM i örnek programlarının kaynağı QMQMSAMP kitaplığında QCSRC, QCLSRC, QCBLLSRC ve QRPGLSRC üyeleri olarak sağlanır.

Örnekleri çalıştırdığınızda kendi kuyruklarınızı kullanabilir ya da bazı örnek kuyruklar oluşturmak için AMQSAMP4 örnek programını çalıştırabilirsiniz. The source for the AMQSAMP4 program is included in file QCLSRC in library QMQMSAMP. Bunu, CRTCLPGM komutunu kullanarak derleyebilirsiniz.

Örnekleri çalıştırmak için, QMQM kitaplığında sağlanan C yürütülebilir sürümlerini kullanın ya da bunları başka bir IBM MQ uygulamasına benzer bir şekilde derleyin.

## Yordam

1. Bir kuyruk yöneticisi yaratın ve varsayılan tanımlamaları ayarlayın.

Örnek programlardan herhangi birini çalıştırabilmek için bunu yapmanız gerekir. Kuyruk yöneticisi oluşturma hakkında daha fazla bilgi için bkz. [IBM MQ Yönetimi](#). İstemci kipinde çalışan uygulamalardan gelen bağlantı isteklerini güvenli bir şekilde kabul etmek üzere bir kuyruk yöneticisinin yapılandırılmasına ilişkin bilgi edinmek için bkz. [“Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102](#).

2. QMQMSAMP kitaplığındaki AMQSDATA adlı üyeden 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 kütük sistemini kullanması için, CRTCMOD üzerinde SYSIFCOPT (\*IFSIO) seçeneğini belirleyin, sonra değiştirge olarak geçirilen kütük adının aşağıdaki biçimde belirtilmesi gerekir:

```
home/me/myfile
```

3. Sorgula, Set, and Echo örneklerinin COBOL sürümlerini kullanmak istiyorsanız, bu örnekleri çalıştırmadan önce süreç tanımlamalarını değiştirin.

Sorgula, Set, and Echo örnekleri için, örnek tanımlamaları bu örneklerin C sürümlerini tetikler. COBOL sürümlerinin olmasını istiyorsanız, süreç tanımlamalarını değiştirmeniz gerekir:

- SYSTEM.SAMPLE.INQPROCESS
- SYSTEM.SAMPLE.SETPROCESS
- SYSTEM.SAMPLE.ECHOPROCESS

IBM i' ta, **CHGMQMPRC** komutunu (ayrıntılar için [Change MQ Process \(CHGMQMPRC\)](#) başlıklı konuya bakın) ya da diğer tanımlamayla **AMQSAMP4** komutunu düzenleyebilir ve çalıştırabilirsiniz.

4. Örnek programları çalıştırın.

Örneklerin her birinin beklediği parametrelerle ilgili daha fazla bilgi için, tek tek örneklerin açıklamalarına bakın.

**Not:** COBOL örnek programları için, kuyruk adlarını parametre olarak geçirdiğinizde, 48 karakter sağlamanız gerekir, gerekirse boş karakterlere sahip olarak doldurma yapmanız gerekir. 48 karakterden başka bir değer, programın 2085. neden koduyla başarısız olmasına neden olur.

## İlgili başvurular

“IBM için örnekler” sayfa 1100

IBM i sistemlerinde IBM MQ için örnek programlar tarafından gösterilen teknikler.

**UNIX** Örnek programların UNIX and Linux üzerinde hazırlanması ve çalıştırılması

Örnek programları UNIX' ta çalıştırmadan önce, önce bir kuyruk yöneticisi yaratmalı ve gerek duyduğunuz kuyrukları da yaratmalısınız. COBOL örneklerini çalıştırmak istiyorsanız, biraz ek hazırlık yapmanız gerekebilir.

## Bu görev hakkında

The IBM MQ on UNIX and Linux systems sample files are in the directories listed in [Çizelge 167 sayfa 1106](#) if the defaults were used at installation time.

Çizelge 167. UNIX and Linux sistemlerinde IBM MQ için örneklerin nerede bulunması gerekir	
İçerik	Dizin
Kaynak dosyalar	<code>MQ_INSTALLATION_PATH/samp</code>
Ölü-harflı kuyruk işleyicisi kaynak dosyaları	<code>MQ_INSTALLATION_PATH/samp/dlq</code>
yürütülür dosyalar	<code>MQ_INSTALLATION_PATH/samp/bin</code>

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

Örneklerin çalışması için bir kuyruk kümesi gerekir. Kendi kuyruklarınızı kullanabilir ya da bir küme yaratmak için örnek MQSC kütüğünü `amqscos0.tst` çalıştırabilirsiniz. Örnekleri çalıştırmak için, sağlanan yürütülebilir sürümleri kullanın ya da bir ANSI derleyicisini kullanarak, kaynak sürümlerini başka uygulamalar gibi derleyin.

## Yordam

1. Bir kuyruk yöneticisi yaratın ve varsayılan tanımlamaları ayarlayın.

Örnek programlardan herhangi birini çalıştırmak için bunu yapmanız gerekir. Kuyruk yöneticisi oluşturma hakkında daha fazla bilgi için bkz. IBM MQ Yönetimi. İstemci kipinde çalışan uygulamalardan gelen bağlantı isteklerini güvenli bir şekilde kabul etmek üzere bir kuyruk yöneticisinin yapılandırılmasına ilişkin bilgi edinmek için bkz. [“Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102.](#)

2. Kendi kuyruklarınızı kullanmayacaksa, bir kuyruk kümesi yaratmak için örnek MQSC kütüğünü `amqscos0.tst` komutunu çalıştırın.

Bunu UNIX and Linux sistemlerinde yapmak için şunu girin:

```
runmqsc QManagerName <amqscos0.tst > /tmp/sampobj.out
```

Hata olmadığından emin olmak için `sampobj.out` dosyasını denetleyin.

3. Sorgula, Set, and Echo örneklerinin COBOL sürümlerini kullanmak istiyorsanız, bu örnekleri çalıştırmadan önce süreç tanımlamalarını değiştirin.

Sorgula, Set, and Echo örnekleri için, örnek tanımlamaları bu örneklerin C sürümlerini tetikler. COBOL sürümlerinin olmasını istiyorsanız, süreç tanımlamalarını değiştirmeniz gerekir:

- SYSTEM.SAMPLE.INQPROCESS
- SYSTEM.SAMPLE.SETPROCESS
- SYSTEM.SAMPLE.ECHOPROCESS

Windows üzerinde, bu örnekleri çalıştırmak için `runmqsc` komutunu kullanmadan önce `amqscos0.tst` dosyasını düzenleyerek ve C yürütülebilir dosya adlarını COBOL yürütülür dosyası adlarına değiştirerek bunu yapın.

4. Örnek programları çalıştırın.

Bir örneği çalıştırmak için, adını herhangi bir değiştirgelerden sonra girin; örneğin:

```
amqsput myqueue qmanagername
```

Burada `myqueue` , iletilerin yerleştirilecek kuyruğun adıdır; `qmanagername` ise, `myqueue` ' un sahibi olan kuyruk yöneticidir.

Örneklerin her birinin beklediği parametrelerle ilgili daha fazla bilgi için, tek tek örneklerin açıklamalarına bakın.

**Not:** COBOL örnek programları için, kuyruk adlarını parametre olarak geçirdiğinizde, 48 karakter sağlamanız gerekir, gerekirse boş karakterlere sahip olarak doldurma yapmanız gerekir. 48 karakterden başka bir değer, programın 2085. neden koduyla başarısız olmasına neden olur.

## İlgili başvurular

“UNIX and Linux sistemleri için örnekler” sayfa 1094

UNIX and Linux sistemlerinde IBM MQ için örnek programlar tarafından gösterilen teknikler.

**Windows** Örnek programların Windows üzerinde hazırlanması ve çalıştırılması

Örnek programları Windows' ta çalıştırmadan önce, önce bir kuyruk yöneticisi yaratmalı ve gerek duyduğunuz kuyrukları da yaratmalısınız. COBOL örneklerini çalıştırmak istiyorsanız, biraz ek hazırlık yapmanız gerekebilir.

## Bu görev hakkında

The IBM MQ for Windows sample files are in the directories listed in [Çizelge 168 sayfa 1107](#), if the defaults were used at installation time. Kuruluş sürücüsü varsayılan olarak < c: > değerine ayarlanır.

Çizelge 168. IBM MQ for Windows için örneklerin nerede bulunması gerekir	
İçerik	Dizin
C kaynak kodu	<code>MQ_INSTALLATION_PATH\Tools\C\Örnekler</code>
Ölü-mektup işleyici örneği için kaynak kod	<code>MQ_INSTALLATION_PATH\Tools\C\Samples\DLQ</code>
COBOL kaynak kodu	<code>MQ_INSTALLATION_PATH\Tools\Cobol \ Örnekler</code>
C yürütülür dosyaları <sup>1</sup>	<code>MQ_INSTALLATION_PATH\ Tools\C\Samples \ Bin (32 bit sürümler)</code> <code>MQ_INSTALLATION_PATH\ Tools\C\Samples\Bin64 (64-bit sürümler)</code>
Örnek MQSC dosyaları	<code>MQ_INSTALLATION_PATH\Tools\MQSC\Samples</code>
Visual Basic kaynak kodu	<code>MQ_INSTALLATION_PATH\Tools\VB\SampVB6</code>
.NET örnekleri	<code>MQ_INSTALLATION_PATH\Tools\dotnet \ Örnekler</code>

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

**Not:** 64 bit sürümler bazı C yürütülebilir dosya örneklerinden kullanılabilir.

Örneklerin çalışması için bir kuyruk kümesi gerekir. Kendi kuyruklarınızı kullanabilir ya da bir kuyruk kümesi yaratmak için örnek MQSC kütüğünü `amqscos0.tst` çalıştırabilirsiniz. Örnekleri çalıştırmak için, sağlanan yürütülebilir sürümleri kullanın ya da kaynak sürümlerini diğer IBM MQ for Windows uygulamalarıyla birlikte derleyin.

## Yordam

1. Bir kuyruk yöneticisi yaratın ve varsayılan tanımlamaları ayarlayın.

Örnek programlardan herhangi birini çalıştırabilmek için bunu yapmanız gerekir. Kuyruk yöneticisi oluşturma hakkında daha fazla bilgi için bkz. [IBM MQ Yönetimi](#). İstemci kipinde çalışan uygulamalardan gelen bağlantı isteklerini güvenli bir şekilde kabul etmek üzere bir kuyruk yöneticisinin yapılandırılmasına ilişkin bilgi edinmek için bkz. [“Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102](#).

2. Kendi kuyruklarınızı kullanmayacaksa, bir kuyruk kümesi yaratmak için örnek MQSC kütüğünü `amqscos0.tst` komutunu çalıştırın.

Bunu Windows sistemlerinde yapmak için şunu girin:

```
runmqsc QManagerName < amqscos0.tst > sampobj.out
```

Hata olmadığından emin olmak için `sampobj.out` dosyasını denetleyin. Bu dosya geçerli dizininizde.

**Not:**

3. Sorgula, Set, and Echo örneklerinin COBOL sürümlerini kullanmak istiyorsanız, bu örnekleri çalıştırmadan önce süreç tanımlamalarını değiştirin.

Sorgula, Set, and Echo örnekleri için, örnek tanımlamaları bu örneklerin C sürümlerini tetikler. COBOL sürümlerinin olmasını istiyorsanız, süreç tanımlamalarını değiştirmeniz gerekir:

- SYSTEM.SAMPLE.INQPROCESS
- SYSTEM.SAMPLE.SETPROCESS
- SYSTEM.SAMPLE.ECHOPROCESS

Windows üzerinde, bu örnekleri çalıştırmak için `runmqsc` komutunu kullanmadan önce `amqscos0.tst` dosyasını düzenleyerek ve C yürütülebilir dosya adlarını COBOL yürütülür dosyası adlarına değiştirerek bunu yapın.

4. Örnek programları çalıştırın.

Bir örneği çalıştırmak için, adını herhangi bir değiştirgelerden sonra girin; örneğin:

```
amqsput myqueue qmanagername
```

Burada `myqueue`, iletilerin yerleştirilecek kuyruğun adıdır; `qmanagername` ise, `myqueue`' un sahibi olan kuyruk yöneticidir.

Örneklerin her birinin beklediği parametrelerle ilgili daha fazla bilgi için, tek tek örneklerin açıklamalarına bakın.

**Not:** COBOL örnek programları için, kuyruk adlarını parametre olarak geçirdiğinizde, 48 karakter sağlamanız gerekir, gerekirse boş karakterlere sahip olarak doldurma yapmanız gerekir. 48 karakterden başka bir değer, programın 2085. neden koduyla başarısız olmasına neden olur.

### İlgili başvurular

[“IBM MQ for Windows için örnekler” sayfa 1097](#)

IBM MQ for Windows için örnek programlar tarafından gösterilen teknikler.

[“Visual Basic samples for IBM MQ for Windows” sayfa 1099](#)

Windows sistemlerinde IBM MQ için örnek programlar tarafından gösterilen teknikler.

### API çıkış örnek programı

Örnek API çıkışı, kullanıcı tarafından belirtilen bir dosyaya `MQAPI_TRACE_LOGFILE` ortam değişkeninde tanımlı bir örnek içeren bir MQI izleme oluşturur.

API çıkışlarına ilişkin daha fazla bilgi için bkz. [“API çıkışlarının yazılması ve derlenmesi” sayfa 984.](#)

### Kaynak

`amqsaxe0.c`

### İkili

`amqsaxe`

### Örnek çıkışa ilişkin yapılandırma

1. Aşağıdaki bilgileri `qm.ini` dosyasına 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 temsil eder.

## 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 temsil eder.

### 2. Ortam değişkenini ayarla

```
MQAPI_TRACE_LOGFILE=/tmp/MqiTrace
```

### 3. Uygulamanızı çalıştırın.

Çıkış dosyaları, namesgibi adlarla /tmp dizininde yaratılır: `MqiTrace.pid.tid.log`

## Zamanuyumsuz tüketim örnek programı

amqscbf örnek programı, iletileri zamanuyumsuz olarak birden çok kuyruktan tüketebilmek için MQCB ve MQCTL ' nin kullanılmasını gösterir.

amqscbf, C kaynak kodu olarak sağlanır ve Windows, UNIX and Linux platformlarında ikili istemci ve sunucu yürütülebilir dosyası 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 belirtin (örnek tarafından en çok on kuyruk desteklenir.)

**Not: Reconnect type** yalnızca istemci programları için geçerlidir.

## Örnek

The example shows amqscbf run as a server program reading one message from QL1 and then being stopped.

QL1' ta bir sına iletisi yerleştirmek 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 'in gösterdiği

Bu örnek, geliş sırasına göre birden çok kuyruktan iletilerin nasıl okunacağını gösterir. Bu, zamanuyumlu MQGET kullanarak çok daha fazla kod gerektirecektir. Zamanuyumsuz tüketim durumunda, herhangi bir yoklama yapılması gerekmez, iş parçacığı ve depolama yönetimi IBM MQ tarafından gerçekleştirilir. "Gerçek bir dünya" örneğinin hatalarla başa çıkması gerekir; örnek hatalarda konsola yazılıyor.

Örnek kodda aşağıdaki adımlar bulunur:

1. Tek ileti tüketimi geri bildirme işlevini tanımlayın,

```
void MessageConsumer(MQHCONN hConn,  
                    MQMD * pMsgDesc,  
                    MQGMO * pGetMsgOpts,  
                    MQBYTE * Buffer,  
                    MQCBC * pContext)  
{ ... }
```

2. Kuyruk yöneticisine bağlan,

```
MQCONN(QMName, &cno, &Hcon, &CompCode, &CReason);
```

3. Giriş kuyruklarını açın ve her birini MessageConsumer geri bildirme işleviyle ilişkilendirin,

```
MQOPEN(Hcon, &od, 0_options, &Hobj, &OpenCode, &Reason);  
cbd.CallbackFunction = MessageConsumer;  
MQCB(Hcon, MQOP_REGISTER, &cbd, &Hobj, &md, &gmo, &CompCode, &Reason);
```

cbd.CallbackFunction, her kuyruk için ayarlanması gerekmez; bu yalnızca giriş alanıdır. Ancak, farklı bir geri bildirme işlevini her kuyrukla ilişkilendirebilirsiniz.

4. İletilerin tüketimine başla,

```
MQCTL(Hcon, MQOP_START, &ctlo, &CompCode, &Reason);
```

5. Kullanıcının Enter tuşuna bastıktan sonra ileti tüketimini durduruncaya kadar bekleyin.

```
MQCTL(Hcon, MQOP_STOP, &ctlo, &CompCode, &Reason);
```

6. Son olarak kuyruk yöneticisinden kopuyor,

```
MQDISC(&Hcon, &CompCode, &Reason);
```

### **Zamanuyumsuz Koy örnek programı**

Amqsapt örneği ve Asynchronous Sput örnek programının tasarımını çalıştırma hakkında bilgi edinin.

Zamanuyumsuz koyma örnek programı, zamanuyumsuz MQPUT çağrısını kullanarak bir kuyruğa ileti koyar ve MQSTAT çağrısını kullanarak durum bilgilerini alır. Farklı platformlarda bu programın adı için bkz. [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#) .

### **amqsapt örneğinin çalıştırılması**

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)

Kuyruk yöneticisi belirtilmediyse, amqspt varsayılan kuyruk yöneticisine bağlanır.

## Zamanuyumsuz Put örnek programının tasarımı

Program, iletileri koymak için hedef kuyruğu açmak için, sağlanan çıkış seçenekleri ile ya da M<sub>QOO</sub>\_OUTPUT ve M<sub>QOO</sub>\_FAIL\_IF\_QUIESCING seçenekleriyle M<sub>QOPER</sub> çağrısını kullanır.

Kuyruk açılmazsa, program, M<sub>QOPED</sub> çağrısının döndürdüğü neden kodunu içeren bir hata iletisi görüntüler. Programı basit tutmak için, bu konuda ve sonraki M<sub>QI</sub> çağrılarında, program seçeneklerin çoğu için varsayılan değerleri kullanır.

Her giriş satırı için, program metni bir arabelleğe okur ve M<sub>QPMO</sub>\_ASYNYC\_response ile M<sub>QPUT</sub> çağrısını kullanır ve bu satırın metnini içeren bir veri paketi iletisi yaratılır ve zamanuyumsuz olarak hedef kuyruğa konmasını sağlar. Program, girişin sonuna ulaşıncaya kadar ya da M<sub>QPUT</sub> çağrısının başarısız olması için devam eder. Program girişin sonuna ulaşırsa, M<sub>QCLOSE</sub> çağrısını kullanarak kuyruğu kapatır.

Daha sonra program M<sub>QSTAT</sub> çağrısını yayınlar, bir M<sub>QSTS</sub> yapısını döndürür ve ileti sayısını başarıyla içeren iletileri, uyarı içeren ileti sayısını ve hata sayısını görüntüler.

## Göz At örnek programları

Göz At örnek programları, M<sub>QGET</sub> çağrısını kullanarak kuyruklardaki iletilere göz atar.

Bu programların adları için bkz. [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#).

## Göz At örnek programının tasarımı

Program, M<sub>QOO</sub>\_BROWSE seçeneğiyle M<sub>QOPEN</sub> çağrısını kullanarak hedef kuyruğu açar. Kuyruk açılmazsa, program, M<sub>QOPED</sub> çağrısının döndürdüğü neden kodunu içeren bir hata iletisi görüntüler.

Kuyruktaki yer alan her ileti için, program iletiyi kuyruktan kopyalamak için M<sub>QGET</sub> çağrısını kullanır, ardından iletide yer alan verileri görüntüler. M<sub>QGET</sub> çağrısı aşağıdaki seçenekleri kullanır:

### M<sub>QGMO</sub>\_BROWSE\_NEXT

M<sub>QOPEN</sub> çağrısının ardından, göz atma imleci kuyruktaki ilk iletiden önce mantıksal olarak konumlandırılır, bu nedenle bu seçenek, arama ilk kez yapıldığında **ilk** iletisinin döndürülmesine neden olur.

### M<sub>QGMO</sub>\_NO\_BEKLEME

Kuyruğun üzerinde ileti yoksa program beklemez.

### M<sub>QGMO</sub>\_ACCEPT\_TRUNCATED\_MSG

M<sub>QGET</sub> çağrısı, sabit büyüklüklerin arabelleğinden birini belirtir. Bu arabellekten daha uzun bir ileti varsa, program kısaltılmış iletiyi görüntüler; bu ileti, iletinin kesildiğini bildiren bir uyarıyla birlikte görüntülenir.

Bu program, bu alanları, aldığı iletide yer alan değerlere ayarlaması nedeniyle, her M<sub>QGET</sub> çağrısından sonra M<sub>QMD</sub> yapısının *MsgId* ve *CorrelId* alanlarını nasıl temizlemeniz gerektiğini gösterir. Bu alanların temizlenmesi, art arda gelen M<sub>QGET</sub> çağrılarının iletilerin kuyruktaki tutulmakta olduğu sırayla alma çağrıları anlamına gelir.

Program kuyruğun sonuna kadar devam eder; M<sub>QGET</sub> çağrısı, M<sub>QRC</sub>\_NO\_MSG\_AVAILABLE neden kodunu döndürür ve program bir uyarı iletisi görüntüler. M<sub>QGET</sub> çağrısının başarısız olması durumunda, program neden kodunu içeren bir hata iletisi görüntüler.

Daha sonra, program M<sub>QCLOSE</sub> çağrısını kullanarak kuyruğu kapatır.

### UNIX, Linux, and Windows için Göz At örnek programları

UNIX, Linux, and Windows' ta Göz At örnek programları hakkında bilgi edinirken bu konuyu kullanmayı düşünün.

Programın C sürümü 2 parametre alır

1. Kaynak kuyruğun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)

Kuyruk yöneticisi belirtilmediyse, varsayılan değer olarak varsayılan değer olarak bağlanır. Örneğin, aşağıdakilerden birini girin:

- amqsgbr myqueue qmanagername
- amqsgbrC myqueue qmanagername
- amq0gbr0 myqueue

Burada myqueue , iletilerin görüntüleneceği kuyruğun adıdır; qmanagername ise, myqueue' un sahibi olan kuyruk yöneticidir.

If you omit the qmanagername, when running the C sample, it assumes that the default queue manager owns the queue.

COBOL sürümünün herhangi bir parametresi yok. Bu, varsayılan kuyruk yöneticisine bağlanır ve çalıştırdığınız zaman sizden sorulur:

```
Please enter the name of the target queue
```

Bu durum söz sahibi olduğunda, her iletinin yalnızca ilk 50 karakteri görüntülenir ve - - - truncated tarafından izlenilir.

#### *IBM üzerindeki Göz At örnek programları*

Her program, programı çağırdığınızda belirlediğiniz kuyruklardaki tüm iletilerin kopyalarını alır; iletiler kuyrukta kalır.

You can use the supplied queue SYSTEM.SAMPLE.LOCAL; run the Put sample program first to put some messages on the queue. Aynı yerel kuyruk için bir diğer ad olan SYSTEM.SAMPLE.ALIASKuyruğunu kullanabilirsiniz. Program, kuyruğun sonuna ulaşıncaya ya da bir MQI çağrısı başarısız oluncaya kadar devam eder.

C örnekleri, genellikle ikinci parametre olarak, Windows sistem örneklerine benzer bir şekilde ikinci parametre olarak kuyruk yöneticisi adını belirtmenize olanak sağlar. Örneğin:

```
CALL PGM(QMQM/AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER' 'QM01')
```

Kuyruk yöneticisi belirtilmediyse, varsayılan değer olarak varsayılan değer olarak bağlanır. Bu, RPG örnekleriyle de ilişkilidir. Ancak, RPG örnekleriyle, varsayılan olarak izin vermek yerine bir kuyruk yöneticisi adı belirtmelisiniz.

#### **ULW** *Tarayıcı örnek programı*

Tarayıcı örnek programı, bir kuyruktaki tüm iletilerin ileti tanımlayıcısını ve ileti içerik alanlarını okur ve yazar.

Örnek program, sadece bir teknik göstermek için değil, bir yardımcı program olarak yazılmıştır. Bu programların adları için bkz. [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#).

Bu program, bu konumlu parametreleri alır:

1. Kaynak kuyruğun adı (gerekli)
2. Kuyruk yöneticisinin adı (gerekli)
3. Özellikler için isteğe bağlı değiştirge (isteğe bağlı)

Bu programlar, bağlantı kimlik doğrulaması için kullanılacak kullanıcı kimliğine ayarlanması gereken **MQSAMP\_USER\_ID** adlı bir ortam değişkeni de kullanır. Bu ayarlandığında, program ilgili kullanıcı kimliğine eşlik etmek için bir parola girilir.

Bu programları çalıştırmak için aşağıdaki komutlardan birini girin:

- amqsbcg myqueue qmanagername
- amqsbcgc myqueue qmanagername



Burada *myqueue* , iletilerin göz atılacağı kuyruğun adıdır; *qmanagername* ise, *myqueue*' un sahibi olan kuyruk yöneticidir.

Kuyruktan her bir iletiyi okur ve stdout 'a şunları yazar:

- Biçimlendirilmiş ileti tanımlayıcı alanları
- İleti verileri (onaltılı biçimde dökümü ve olas olas, karakter biçimi)

Değer	Davranış
0	Varsayılan davranış, IBM WebSphere MQ 6 için olduğu gibi. Uygulamaya teslim edilen özellikler, iletinin alındığı <b>PropertyControl</b> kuyruk özniteliğine bağlıdır.
1	Bir ileti tanıtıcısı yaratılır ve MQGET ile kullanılır. İleti tanımlayıcısında (ya da uzantıda) yer alan durumlar dışında, iletinin özellikleri ileti tanımlayıcısına benzer bir şekilde görüntülenir. Örneğin: <pre>****Message properties**** property name: property value</pre> Ya da kullanılabilir özellik yoksa: <pre>****Message properties**** None</pre> Sayısal değerler printf kullanılarak görüntülenir, dizgi değerleri tek tırnak işaretleriyle çevrilir ve bayt dizgileri, ileti tanımlayıcısına göre X ve tek tırnak işaretleriyle çevrelenir.
2	MQGMO_NO_XX_ENCODE_CASE_ONE properties belirtildi, bu nedenle yalnızca ileti tanımlayıcısı özellikleri döndürülecek.
3	İleti verilerinde tüm özelliklerin döndürülmesi için MQGMO_PROPERTIES_FORCE_MQRFH2 belirtildi.
4	MQGMO_PROPERTIES_COMPATIBILITY belirtildi; böylece, bir IBM WebSphere MQ 6 özelliğinin dahil edilip edilmediğine bağlı olarak tüm özellikler döndürülebilir, tersi durumda özellikler atılır.

Program, iletinin ilk 65535 karakterinin yazdırılması ile kısıtlanmıştır ve daha uzun bir ileti okunduysa, `truncated msg` neden ile başarısız olur.

Bu yardımcı programdaki çıkışa bir örnek için bakınız: [Browsing queue](#).

### **CICS hareket örneği**

executablekaynak kodu için `amqscic0.ccs` adlı bir örnek CICS hareket programı ve yürütülebilir sürüm için `amqscic0` adı verilir. Standart CICS olanaklarını kullanarak işlem yapabilirsiniz.

Altyapınız için gerekli olan komutlara ilişkin ayrıntılar için "[Yordamsal uygulama oluşturulması](#)" sayfa 1030 ' e bakın.

Hareket, `SYSTEM.SAMPLE.CICS`iletim kuyruğundan ileti okur.Varsayılan kuyruk yöneticisinde `WORKQUEUE` ve bunları, iletinin iletim üstbilgisinde yer alan adı, yerel kuyruğa yerleştirir. `SYSTEM.SAMPLE.CICS`kuyruğuna herhangi bir hata gönderilsin.DQ.

**Not:** Bu kuyrukları ve örnek giriş kuyruklarını yaratmak için örnek bir MQSC komut dosyası `amqscic0.tst` kullanabilirsiniz.

### **Connect örnek programı**

Connect örnek programı, bir istemcideki MQCONNX çağrısını ve seçeneklerini keşfetmenizi sağlar. Bu örnek, MQCONNX çağrısını kullanarak kuyruk yöneticisine bağlanır, MQINQ çağrısını kullanarak kuyruk yöneticisinin adını sorgular ve görüntüler. Ayrıca, `amqscnxc` örneğinin çalıştırılmasıyla ilgili bilgi edinin.

**Not:** Connect örnek programı bir istemci örneğidir. Bir sunucuda derleyebilir ve çalıştırabilirsiniz, ancak işlev yalnızca bir istemcide anlamlıdır ve yalnızca istemci yürütülebilir dosyaları sağlar.

## amqscnxc örneğinin çalıştırılması

Connect örnek programının komut satırı sözdizimi şöyledir:

```
amqscnxc [-x ConnName [-c SvrconnChannelName]] [-u User] [QMgrName]
```

Parametreler isteğe bağlıdır ve sırası QMgrNamedışında önemli değildir; bu değer, belirtilirse, son olarak gelmelidir. Değiştireliler şunlardır:

### ConnName

Sunucu kuyruk yöneticisinin TCP/IP bağlantı adı

TCP/IP bağlantı adını belirtmezseniz, MQCONN, *ClientConnPtr* değeri NULL (boş değer) olarak ayarlanmış bir şekilde verilir.

### SvrconnChannelAd

Sunucu bağlantı kanalının adı

TCP/IP bağlantı adını belirtirseniz, ancak sunucu bağlantı kanalına (ters çevirme işlemine izin verilmez) belirtirseniz, örnek SYSTEM.DEF.SVRCONN.

### Kullanıcı

Bağlantı kimlik doğrulaması için kullanılacak kullanıcı adı

Bu seçeneği belirlerseniz, program, o kullanıcı kimliğine eşlik edecek bir parola için bilgi isteminde olur.

### QMgrName

Hedef kuyruk yöneticisinin adı

Hedef kuyruk yöneticisini belirtmezseniz, örnek olarak belirtilen TCP/IP bağlantı adına hangi kuyruk yöneticisine bağlanıyorsa, bu örnek bağlantı kurar.

**Not:** Tek parametre olarak bir soru işareti girerseniz ya da yanlış parametreler girdiğinizde, programın nasıl kullanılacağını açıklayan bir ileti elde edin.

Örneği komut satırı seçenekleri olmadan çalıştırırsanız, bağlantı bilgilerini belirlemek için MQSERVER ortam değişkeninin içeriği kullanılır. (Bu örnekte MQSERVER, SYSTEM.DEF.SVRCONN/TCP/machine.site.company.comolarak ayarlıdır.) Şu şekilde çıktıyı görüyorsunuz:

```
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 çalıştırıp bir TCP/IP bağlantı adı ve bir sunucu bağlantısı kanal adı sağlıyorsa, ancak hedef kuyruk yöneticisi adı girmiyorsa, aşağıdaki gibi:

```
amqscnxc -x machine.site.company.com -c SYSTEM.ADMIN.SVRCONN
```

Varsayılan kuyruk yöneticisi adı kullanılır ve şu şekilde çıktıyı 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 çalıştırırsanız ve bir TCP/IP bağlantı adı ve hedef kuyruk yöneticisi adı sağlıyorsa, aşağıdaki gibi:

```
amqscnxc -x machine.site.company.com MACHINE
```

Çıktıyı şu şekilde görüyorsunuz:

```
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, veri dönüştürme çıkış yordamlarından oluşan bir iskelettir. Veri dönüştürme örneğinin tasarımıyla ilgili bilgi edinin.

Bu programların adları için bkz. [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#).

### **Veri dönüştürme örneğinin tasarımı**

Her veri dönüştürme çıkış yordamı, adlandırılan tek bir ileti biçimini dönüştürür. Bu iskelet, veri dönüştürme çıkış oluşturma yardımcı programı tarafından oluşturulan kod parçalarına ilişkin bir sarıcı olarak tasarlanmıştır.

Bu yardımcı program her veri yapısı için bir kod parçası üretir; bu tür yapılar bir biçim oluşturur; bu nedenle, tüm biçimdeki veri dönüştürme işlemini yapmak üzere bir yordam üretmek üzere bu iskeletten çok sayıda kod parçası eklenir.

Daha sonra program, dönüştürmenin başarılı mı, yoksa başarısız mı olduğunu denetler ve çağırana için gereken değerleri döndürür.

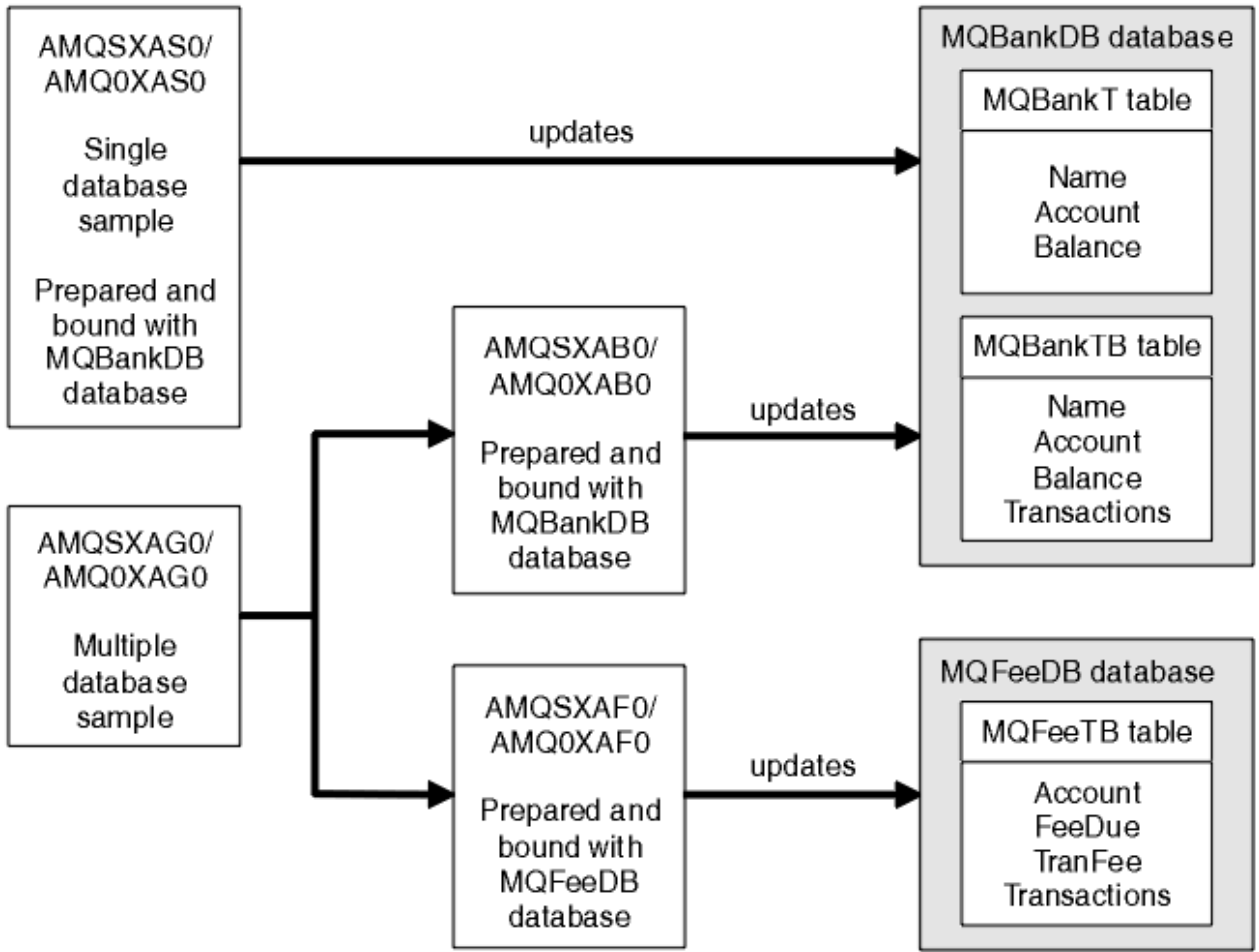
### **Veritabanı koordinasyonu örnekleri**

IBM MQ ' in aynı iş birimi içinde hem IBM MQ güncelleştirmelerini hem de veritabanı güncellemelerini nasıl koordine edebildiğini gösteren iki örnek verilmiştir.

Bu örnekler şunlardır:

1. AMQXSAS0 (in C) or AMQ0XAS0 (in COBOL), which updates a single database within an IBM MQ unit of work.
2. AMQSXAG0 (C içinde) ya da AMQ0XAG0 (COBOL içinde), AMQSXAB0 (C içinde) ya da AMQ0XAB0 (COBOL ' de) ve AMQXAF0 (C içinde) ya da AMQ0XAF0 (COBOL içinde), bir IBM MQ iş birimi içindeki iki veritabanını güncelleyen, birden çok veritabanının nasıl erişilebileceğini gösterir. Bu örnekler, MQBEGIN çağrısının, karma SQL ve IBM MQ çağrılarının ve bir veritabanına bağlanmanın nerede ve ne zaman kullanılacağını göstermek için sağlanmıştır.

[Şekil 133 sayfa 1116](#), sağlanan örneklerin veritabanılarını güncellemek için nasıl kullanıldığını gösterir:



Şekil 133. Veritabanı koordinasyonu örnekleri

Programlar bir kuyruktan (syncpoint altında) bir ileti okur, daha sonra, iletiyle ilgili bilgileri kullanarak, ilgili bilgileri veritabanından alıp güncellemesini sağlar. Daha sonra, veritabanının yeni durumu yazdırılır.

Program mantığı aşağıdaki gibidir:

1. Program bağımsız değişkeninden giriş kuyruğunun adını kullan
2. MQCONN kullanarak varsayılan kuyruk yöneticisine (ya da isteğe bağlı olarak C ' de sağlanan ada) bağlan
3. Hata olmamakla birlikte, giriş için bir kuyruk açın (MQOPEN komutunu kullanın)
4. MQSTART komutunu kullanarak bir iş birimi başlatma
5. Syncpoint altında kuyruktan sonraki iletiyi (MQGET ile) al
6. Veritabanlarından bilgi al
7. Veritabanlarındaki bilgileri güncelle
8. Değişiklikleri MQCMIT kullanarak kesinleştir
9. Güncellenen bilgileri yazdır (hata olarak kullanılacak ileti yok ve döngü sona eriyor)
10. MQCLOSE komutunu kullanarak kuyruğu kapat
11. MQDISC komutunu kullanarak kuyruktan bağlantıyı kes

Örneklerde SQL imleçleri kullanılır; bu nedenle, veri tabanlarından (yani birden çok yönetim ortamı) okuyan bir ileti, bir ileti işlenirken kilitlenir ve bu programların birden çok örneğinin aynı anda çalışmasına olanak sağlar. Geçici çizelgeler açık bir şekilde açılmıştır, ancak MQCMIT çağrısıyla örtük olarak kapatılır.

The single database sample (AMQSXAS0 or AMQ0XAS0) has no SQL CONNECT statements and the connection to the database is implicitly made by IBM MQ with the MQBEGIN call. Çoklu veritabanı örneği

(AMQXSAG0 ya da AMQ0XAG0, AMQXSAB0 ya da AMQ0XAB0ya da AMQ0XAF0), bazı veritabanı ürünleri yalnızca bir etkin bağlantıya izin verdiği için, SQL CONNECT deyimlerine sahiptir. Bu durum veritabanı ürününüz için geçerli değilse ya da birden çok veritabanı ürününde tek bir veritabanına erişiyorsanız, SQL CONNECT deyimleri kaldırılabilir.

Örnekler IBM Db2 veritabanı ürünüyle birlikte hazırlanır; bu nedenle, diğer veritabanı ürünleriyle çalışmak için bunları değiştirmeniz gerekebilir.

The SQL error checking uses routines in UTIL.C and CHECKERR.CBL supplied by Db2. Derleme ve bağlantı oluşturulmadan önce bunlar derlenmeli ya da değiştirilmelidir.

**Not:** Micro Focus COBOL kaynağı CHECKERR.MFC , program tanıtıcısını büyük harfe çevirmeniz gerekir, bu CHECKERR, AMQ0XAS0 için doğru bağlantı olmalıdır.

*Veritabanlarının ve çizelgelerin yaratılması*

Örnekleri derlemeden önce veritabanlarını ve tabloları oluşturun.

Veritabanlarını yaratmak için, veritabanı ürününüz için olağan yöntemi kullanın; örneğin:

```
DB2 CREATE DB MQBankDB
DB2 CREATE DB MQFeeDB
```

SQL deyimlerini kullanarak çizelgeleri yaratın:

C içinde:

```
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 'da:

```
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.
```

SQL deyimlerini aşağıdaki gibi 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 ' ı ekleyin.

### Örneklerin derlenmesi, derlenmesi ve bağlanması

C ve COBOL içindeki örnekleri derleme öncesi, derleme ve bağlama hakkında bilgi edinin.

.SQC dosyalarını (C içinde) ve .SQB dosyalarını (COBOL ' da) ön derleyin ve .C ya da .CBL dosyalarını üretmek için uygun veritabanına karşı bunları bağlayın. Bunu yapmak için, veritabanı ürününüz için tipik bir yöntemi kullanın.

## C içinde önderleme

```

db2 connect to MQBankDB
db2 prep AMQXSAS0.SQC
db2 connect reset

db2 connect to MQBankDB
db2 prep AMQXAB0.SQC
db2 connect reset

db2 connect to MQFeeDB
db2 prep AMQXAF0.SQC
db2 connect reset

```

## COBOL ' de ön

```

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

```

## Derleme ve bağlantı oluşturma

Aşağıdaki örnek komutlar *DB2TOP* ve *MQ\_INSTALLATION\_PATH* simgelerini kullanır. *DB2TOP*, Db2 ürününe ilişkin kuruluş dizinini gösterir. *MQ\_INSTALLATION\_PATH*, IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

-  AIX üzerinde dizin yolu şudur:

```
/usr/lpp/db2_05_00
```

- **Solaris** Solaris' ta dizin yolu şudur:

```
/opt/IBMDB2/V5.0
```

- **Windows** Windows sistemlerinde, dizin yolu, ürünü kurarken seçilen yola bağlıdır. Varsayılan ayarları seçerseniz, yol şöyle olur:

```
c:\sqllib
```

**Not:** Before issuing the link command on Windows systems, ensure that the LIB environment variable contains paths to the Db2 and IBM MQ libraries.

Aşağıdaki dosyaları geçici bir dizine kopyalayın:

- IBM MQ kurulumunuzdaki amqsxag0.c dosyası

**Not:** Bu dosya aşağıdaki dizinlerde bulunabilir:

- **Linux** **UNIX** UNIX 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 dosyalarını önceden derleyerek elde etmiş olduğunuz .c dosyaları.
- The files util.c and util.h from your Db2 installation.

**Not:** Bu dosyalar şu dizinde bulunabilir:

```
DB2TOP/samples/c
```

Kullanmakta olduğunuz platforma ilişkin aşağıdaki derleyici komutunu kullanarak her bir .c dosyası için nesne dosyalarını oluşturun:

- **AIX** AIX

```
xlc_r -I MQ_INSTALLATION_PATH/inc -I DB2TOP/include -c -o  
FILENAME.o FILENAME.c
```

- **Solaris** Solaris

```
cc -Aa -KPIC -mt -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
```

Kullanmakta olduğunuz platform için aşağıdaki bağlantı komutunu kullanarak amqsxag0 yürütülebilir 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
```

• **Solaris** Solaris

```
cc -mt -L DB2TOP/lib -ldb2 -L MQ_INSTALLATION_PATH/lib  
-lmqm -lthread -lsocket -lc -lnsl -ldl util.o  
amqsxaf0.o amqsxab0.o amqsxag0.o -o amqsxag0
```

• **Windows** Windows sistemleri

```
link util.obj amqsxaf0.obj amqsxab0.obj amqsxag0.obj mqm.lib db2api.lib  
/out:amqsxag0.exe
```

Kullanmakta olduğunuz platforma ilişkin aşağıdaki derleme ve bağlantı oluşturma komutlarını kullanarak amqsxas0 yürütülebilir dosyasını oluşturun:

• **AIX** AIX

```
xlc_r -H512 -T512 -L DB2TOP/lib -ldb2  
-L MQ_INSTALLATION_PATH/lib -lmqm util.o amqsxas0.o -o amqsxas0
```

• **Solaris** Solaris

```
cc -mt -L DB2TOP/lib -ldb2 -L MQ_INSTALLATION_PATH/lib  
-lqm -lthread -lsocket -lc -lnsl -ldl util.o  
amqsxas0.o -o amqsxas0
```

• **Windows** Windows sistemleri

```
link util.obj amqsxas0.obj mqm.lib db2api.lib /out:amqsxas0.exe
```

## Ek bilgi

**AIX** AIX üzerinde çalışıyorsanız ve Oracle' a erişmek istiyorsanız, xlc\_r derleyicisini kullanın ve libmqm\_r.a bağlantısını tıkladın.

### Örnekleri çalıştırma

C ve COBOL üzerinde veritabanı eşgüdümü örnekleri çalıştırılmadan önce kuyruk yöneticisinin nasıl yapılandırılacağı hakkında bilgi edinmek için bu bilgileri kullanın.

Örnekleri çalıştırmadan önce, kuyruk yöneticisini kullanmakta olduğunuz veritabanı ürünüyle yapılandırın. Bunun nasıl gerçekleştirileceği hakkında bilgi için bkz. [Senaryo 1: Kuyruk yöneticisi koordinasyonu gerçekleştirir.](#)

Aşağıdaki başlıklar, C ve COBOL ' de örnekleri çalıştırma hakkında bilgi sağlar:

- [“C örnekleri” sayfa 1120](#)
- [“COBOL örnekleri” sayfa 1121](#)

## C Örnekleri

İletiler kuyruktan okunmak üzere aşağıdaki biçimde olmalıdır:

```
UPDATE Balance change=nnn WHERE Account=nnn
```



İletileri kuyruğa koymak için MQSPUT kullanılabilir.

Veritabanı koordinasyonu örnekleri iki parametre alır:

1. Kuyruk adı (gerekli)
2. Kuyruk yöneticisi adı (isteğe bağlı)

Assuming that you have created and configured a queue manager for the single database sample called singDBQM, with a queue called singDBQ, you increment Mr Fred Bloggs's account by 50 as follows:

```
AMQSPUT singDBQ singDBQM
```

Daha sonra, aşağıdaki iletiyle ilgili anahtar:

```
UPDATE Balance change=50 WHERE Account=1
```

Kuyruğa birden çok ileti yerleştirebilirsiniz.

```
AMQSXAS0 singDBQ singDBQM
```

Daha sonra, Bay Fred Bloggs 'in hesabında güncellenen durum yazdırılır.

Assuming that you have created and configured a queue manager for the multiple-database sample called multDBQM, with a queue called multDBQ, you decrement Ms Mary Brown's account by 75 as follows:

```
AMQSPUT multDBQ multDBQM
```

Daha sonra, aşağıdaki iletiyle ilgili anahtar:

```
UPDATE Balance change=-75 WHERE Account=3
```

Kuyruğa birden çok ileti yerleştirebilirsiniz.

```
AMQSXAG0 multDBQ multDBQM
```

Daha sonra, Bayan Mary Brown 'ın hesabında güncellenen durum yazdırılır.

## COBOL örnekleri

İletiler kuyruktan okunmak üzere aşağıdaki biçimde olmalıdır:

```
UPDATE Balance change=snnnnnnnn WHERE Account=nnnnnnnn
```

Basitlik için, Balance change imzalı sekiz karakterli bir sayı olmalı ve Account sekiz karakterden oluşan bir sayı olmalıdır.

İletileri kuyruğa koymak için örnek AMQSPUT örneği kullanılabilir.

Örnekler parametre alır ve varsayılan kuyruk yöneticisini kullanır. Bu, herhangi bir zamanda örneklerden yalnızca birini çalıştırabilecek şekilde yapılandırılabilir. Assuming that you have configured the default queue manager for the single database sample, with a queue called singDBQ, you increment Mr Fred Bloggs's account by 50 as follows:

```
AMQSPUT singDBQ
```

Daha sonra, aşağıdaki iletiyle ilgili anahtar:

```
UPDATE Balance change=+00000050 WHERE Account=00000001
```

Kuyruğa birden çok ileti koyabilirsiniz:

```
AMQ0XAS0
```

Kuyruğun adını yazın:

```
singDBQ
```

Daha sonra, Bay Fred Bloggs 'in hesabında güncellenen durum yazdırılır.

Assuming that you have configured the default queue manager for the multiple database sample, with a queue called multDBQ, you decrement Ms Mary Brown's account by 75 as follows:

```
AMQSPUT multDBQ
```

Daha sonra, aşağıdaki iletiyle ilgili anahtar:

```
UPDATE Balance change=-00000075 WHERE Account=00000003
```

Kuyruğa birden çok ileti koyabilirsiniz:

```
AMQ0XAG0
```

Kuyruğun adını yazın:

```
multDBQ
```

Daha sonra, Bayan Mary Brown 'ın hesabında güncellenen durum yazdırılır.

### ***Ölü-harfli kuyruk işleyicisi örneği***

Örnek bir dead-letter queue işleyicisi sağlanıyor, yürütülebilir sürümün adı amqsdlq. RUNMQDLQ ' dan farklı bir ileti kuyruğu işleyicisi istiyorsanız, tabanınız olarak kullanmak üzere, örneğin kaynağı kullanılabilir.

Örnek, ürün içinde sağlanan ölü harf işleyiciye benzer, ancak izleme ve hata raporlamasının farklı olduğunu da sağlar. Kullanabileceğiniz iki ortam değişkeni vardır:

#### **ODQ\_TRACE**

İzlemeyi değiştirmek için EVET ya da Evet olarak ayarla

#### **ODQ\_MSG**

Hata ve bilgi iletilerini içeren dosyanın adını girin. Sağlanan dosyaya amqsdlq.msgadı verilir.

Platformunuza bağlı olarak, **export** ya da **set** komutlarını kullanarak ortamınız tarafından bilinen bu değişkenleri **unset** komutunu kullanarak kapatmanız gerekir.

You can modify the error message file, amqsdlq.msg, to suit your own requirements. The sample puts messages to stdout, **değil** to the IBM MQ error log file.

Altyapınıza ilişkin [Administering](#) ya da *System Management Guide* adlı kılavuz, ölü harf işleyicinin nasıl çalıştığını ve nasıl çalıştığınızı açıklar.

### ***Dağıtım Listesi örnek programı***

Dağıtım Listesi örneği amqsptl0 , ileti kuyruklarına bir ileti yerleştirmenin bir örneğini verir. MQPUT örneğine, amqsput0' a dayalıdır.

## Dağıtım Listesi örneği çalıştırılıyor, amqsptl0

Dağıtım Listesi örneği, Koyma örneklerine benzer bir şekilde çalışır.

Bu değiştirge aşağıdaki değiştireleri alır:

- Kuyrukların adları
- Kuyruk yöneticilerinin adları

Bu değerler çift olarak girilir. Örneğin:

```
amqsptl0 queue1 qmanagername1 queue2 qmanagername2
```

Kuyruklar, MQPUT kullanılarak kuyruklara açılır ve MQPUT ile kuyruklara konar. Kuyruk ya da kuyruk yöneticisi adlarından herhangi biri tanınmazsa neden kodları döndürülür.

İletiler arasında akış yapabilmeleri için kuyruk yöneticileri arasında kanallar tanımlamayı unutmayın. Örnek program bunu sizin için yapmaz.

## Dağıtım Listesi örneğinin tasarımı

İleti Kayıtları 'nı (MQPMR ' lar) her hedef için ileti özniteliklerini belirtin. Örnek, *MsgId* ve *CorrelId* için değerler sağlar ve bu değerler MQMD yapısında belirtilen değerleri geçersiz kılar.

MQPMO yapısındaki *PutMsgRecFields* alanı, MQPMRS ' de hangi alanların bulunduğunu gösterir:

```
MLONG PutMsgRecFields=MQPMRF_MSG_ID + MQPMRF_CORREL_ID;
```

Daha sonra, örnek yanıt kayıtlarını ve nesne kayıtlarını ayırır. Nesne kayıtları (MQORs) en az bir çift ad ve çift sayıda ad gerektirir; bu da, *ObjectName* ve *ObjectQMgrName*.

Sonraki aşama, MQCONN kullanan kuyruk yöneticilerine bağlanmayı içerir. Bu örnek, MQOR içindeki ilk kuyrukla ilişkilendirilmiş kuyruk yöneticisine bağlanmayı dener; bu işlem başarısız olursa, nesne kayıtlarında sırayla devam eder. Herhangi bir kuyruk yöneticisine ve program çıkışlarına bağlanmak olanaklı değilse, size bilgi verilir.

Hedef kuyruklar, MQPUT kullanılarak açılır ve MQPUT kullanılarak bu kuyruklara ileti konması gerekir. Yanıt kayıtlarında (MQRR ' lar) herhangi bir sorun ve hata bildirilir.

Son olarak, hedef kuyruklar MQCLOSE kullanılarak kapatılır ve program MQDISC kullanılarak kuyruk yöneticisinden bağlantıyı keser. *CompCode* ve *Reason*' yi belirten her çağrı için aynı yanıt kayıtları kullanılır.


## Echo örnek programları

Echo örnek programları, ileti kuyruğundan yanıt kuyruğuna bir ileti echo eder.

Bu programların adları için bkz. [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#).

Programlar, tetiklenen programlar olarak çalıştırılmak üzere tasarlanmıştır.

IBM i, UNIX, Linux, and Windows sistemlerinde, tek giriş girişi, hedef kuyruk ve kuyruk yöneticisi adını içeren bir MQTMC2 (tetikleme iletisi) yapısıdır. COBOL sürümü, varsayılan kuyruk yöneticisini kullanır.

 IBM i üzerinde, tetikleme işleminin çalışması için, kullanmak istediğiniz Echo örnek programının, SYSTEM.SAMPLE.ECHO. To do this, specify the name of the Echo sample program that you want to use in the *AppId* field of the process definition SYSTEM.SAMPLE.ECHOPROCESS. (Bunun için, CHGMQMPRC komutunu kullanabilirsiniz; ayrıntılı bilgi için bkz. [Change MQ Process \(CHGMQMPRC\)](#).) Örnek kuyruğun bir FIRST tetikleyicisi tipi vardır, bu nedenle, İstek örneğini çalıştırmadan önce kuyruğunda önceden iletiler varsa, gönderdiğiniz iletiler Echo örneği tetiklenmez.

When you have set the definition correctly, first start AMQSERV4 in one job, then start AMQSREQ4 in another. AMQSERV4 yerine AMQSTRG4 'yi kullanabilirsiniz, ancak olası iş gönderimi gecikmeleri, gerçekleşenleri takip etmeyi daha az kolaylaştırabilirdi.

Use the Request sample programs to send messages to queue SYSTEM.SAMPLE.ECHO. Echo örnek programları, istek iletilisinde, istek iletilisinde belirtilen yanıt kuyruğuna veri içeren bir yanıt iletilisi gönderir.

## Echo örnek programlarının tasarımı

Program, tetikleme iletilisi yapısında adı belirtilen kuyruğu açarken, bu işlem başlatıldığında iletileceği bir kuyrukta yer alan kuyruğu açar. (Bu netlik için, istek kuyruğu olarak anılır.) Program, bu kuyruğu paylaşılan giriş için açmak için MQOPEN çağrısını kullanır.

Program, bu kuyruktan iletileri kaldırmak için MQGET çağrısını kullanır. Bu çağrı, 5 saniye bekleme süresi ile MQGMO\_ACCEPT\_TRUNCATED\_MSG, MQGMO\_CONVERT ve MQGMO\_WAWT seçeneklerini kullanır. Program, bir istek iletilisi olup olmadığını görmek için her iletilinin tanımlayıcısını sınar; değilse, program iletiyi atar ve bir uyarı iletilisi görüntüler.

Her giriş satırı için program, metni bir arabelleğe okur ve MQPUT1 çağrısını kullanarak, o satırın metnini içeren bir istek iletilisini yanıtlama kuyruğunda kullanır.

MQGET çağrısının başarısız olması durumunda, program yanıt kuyruğuna bir rapor iletilisi koyar ve ileti tanımlayıcısının *Feedback* alanını MQGET tarafından döndürülen neden koduna göre ayarlar.

İstek kuyruğunda bir ileti kalmadığında, program o kuyruğu kapatır ve kuyruk yöneticisinden bağlantıyı keser.

**IBM i** On IBM i, the program can also respond to messages sent to the queue from platforms other than IBM MQ for IBM i, although no sample is supplied for this situation. Echo programının çalışmasını sağlamak için:

- Metin isteği iletileri göndermek için, **Format**, **Encoding** ve **CCSID** parametrelerini doğru biçimde belirten bir program yazın.  
ECHO programı, gerekiyorsa, kuyruk yöneticisinin ileti verisi dönüştürme işlemini gerçekleştirmesini ister.
- Specify CONVERT(\*YES) on the IBM MQ for IBM i sending channel, if the program that you have written does not provide similar conversion for the reply.

## Get Sample programlar

Alma örnek programları, MQGET çağrısını kullanarak kuyruktan ileti alır.

Bu programların adları için bkz. [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#).

## Get Sample programının tasarımı

Program, MQOPEN çağrısını MQOO\_INPUT\_AS\_Q\_DEF seçeneğiyle kullanarak hedef kuyruğu açar. Kuyruğu açamazsa, program, MQOPED çağrısının döndürdüğü neden kodunu içeren bir hata iletilisi görüntüler.

Kuyrukta yer alan her ileti için, program iletiyi kuyruktan kaldırmak için MQGET çağrısını kullanır, daha sonra iletide bulunan verileri görüntüler. MQGET çağrısı, kuyruğun üzerinde bir ileti yoksa programın bu dönemi bekleyeceği şekilde, 15 saniyelik bir *WaitInterval* belirten MQGMO\_WATM seçeneğini kullanır. Bu aralığın süresi dolmadan bir ileti gelmezse, çağrı başarısız olur ve MQRC\_NO\_MSG\_AVAILABLE neden kodunu döndürür.

Bu program, bu alanları, aldığı iletide yer alan değerlere ayarlaması nedeniyle, her MQGET çağrısından sonra MQMD yapısının *MsgId* ve *CorrelId* alanlarını nasıl temizlemeniz gerektiğini gösterir. Bu alanların temizlenmesi, art arda gelen MQGET çağrılarının iletilerin kuyrukta tutulmakta olduğu sırayla alma çağrıları anlamına gelir.

MQGET çağrısı, sabit büyüklerin arabelleğinden birini belirtir. Bu arabellekten daha uzun bir ileti varsa, arama başarısız olur ve program durur.

Bu program, MQGET çağrısına ilişkin MQRC\_NO\_MSG\_AVAILABLE neden kodunu döndürünceye ya da MQGET çağrısının başarısız olduğu zamana kadar devam eder. Arama başarısız olursa, program neden kodunu içeren bir hata ileti görüntüler.

Daha sonra, program MQCLOSE çağrısını 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, bir ortam değişkeni ya da istemci kanal tanımlama dosyası tarafından tanımlanmış 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).

Bu programlar, bağlantı kimlik doğrulaması için kullanılacak kullanıcı kimliğine ayarlanması gereken **MQSAMP\_USER\_ID** adlı bir ortam değişkeni de kullanır. Bu seçenek belirlendiğinde, program bu kullanıcı kimliğine eşlik edecek bir parola ister.

Bu programları çalıştırmak için aşağıdakilerden birini girin:

- amqsget myqueue qmanage:name
- amqsgetc myqueue qmanage:name

Burada myqueue , programın iletileri alacağı kuyruğun adı ve qmanage:name , myqueue' 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 ulaşılabileceğine ilişkin uygulama ayrıntılarını sağlamanız gerekir.

Olağan durumda bu, MQSERVER ortam değişkeni kullanılarak ya da **amqsgetc** ' e ortam değişkenleri kullanılarak da sağlanabilen bir istemci kanal tanımlama çizelgesi kullanılarak bağlantı ayrıntıları tanımlanarak gerçekleştirilir; örneğin, MQCCDTURL.

1414 kapısında çalışan ve varsayılan sunucu bağlantı kanalını kullanan bir dinleyicisi olan bir kuyruk yöneticisine yerel olarak bağlanarak MQSERVER ' in kullanılmasına ilişkin bir örnek:

```
export MQSERVER="SYSTEM.DEF.SVRCONN/TCP/ localhost(1414) "
```

## **Yüksek kullanılabilirlikli örnek programlar**

**amqsgbac**, **amqspbac** ve **amqsmbac** yüksek düzeyde kullanılabilirlik örnek programları, bir kuyruk yöneticisinin arızalanması sonrasında kurtarmayı göstermek için otomatik istemci yeniden bağlantısını kullanır. **amqsfbac** , ağ üzerinden depolama kullanan bir kuyruk yöneticisinin bir hatanın ardından veri bütünlüğünü korumasını denetler.

**amqsgbac**, **amqspbac** ve **amqsmbac** programları komut satırından başlatılır ve çok eşgörümlü bir kuyruk yöneticisinin bir eşgörümlü başarısız olduktan sonra yeniden bağlantıyı göstermek için birlikte kullanılabilir.

Diğer bir yöntem olarak, genellikle bir kuyruk yöneticisi grubunda yapılandırılmış olan tek eşgörümlü kuyruğu yöneticilerine istemci yeniden bağlantısını göstermek için **amqsgbac**, **amqspbac** ve **amqsmbac** 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 1128.

Dosya sistemi bütünlüğünü denetlemek için **amqmfsc** ile paralel olarak **amqsfbac** komutunu kullanın. Daha fazla bilgi için bkz. [amqmfsc](#) (dosya sistemi denetimi) ve [Paylaşılan dosya sistemi davranışının doğrulanması](#) .

### **amqspbac queueName [qMgrAd]**

- **amqspbac** 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.

### **amqsgbac queueName [qMgrAd]**

- **amqsgbac** 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.

### **amqsmbac -s sourceQueueAdı -t targetQueueAdı [ -m qMgrAdı ] [ -w waitInterval ]**

- **amqsmbac** 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.

### **amqsfbac QueueManagerAdı QueueName SideQueueName InTransactionCount RepeatCount (0 | 1 | 2)**

- **amqsfbac** bir IBM MQ MQI client uygulamasıdır. NAS ya da küme kütük sistemi gibi ağa bağlı depolama kullanan bir IBM MQ çok eşgörümlü kuyruk yöneticisinin veri bütünlüğünü korumasını denetler. **amqsfbac** komutunu [Paylaşılan dosya sistemi davranışının doğrulanması](#) içinde 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. **amqsfbac** her seferinde kuyruk yöneticisine yeniden bağlanır ve devam eder. Sinama, hiçbir iletinin kaybolmadığından emin olmaktır.
- *InTransactionSayı* iletileri her işlem içine konmuştur. İşlem *RepeatCount* kez yinelenir. Bir hareket içinde bir hata oluşursa, **amqsfbac** kuyruk yöneticisine yeniden bağlandığında **amqsfbac** 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

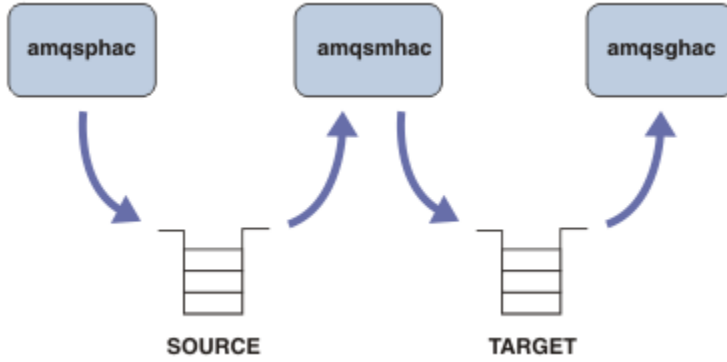
Örnekleri çalıştırmak için bir istemci ve sunucu bağlantı kanalı yapılandırmanız gerekir. İstemci doğrulama yordamı, bir istemci test ortamının nasıl ayarlanacağını açıklar.

Diğer bir seçenek olarak, aşağıdaki örnekte sağlanan yapılanışı kullanın.

### amqsgnac, amqsfhac ve amqsmnac Yöntemlerinin Kullanımı-Örnek

Bu örnek, tek bir yönetim ortamı kuyruk yöneticisini kullanarak yeniden bağlanabilir istemcileri gösterir.

İletiler, SOURCE tarafından **amqsfhac** kuyruğuna yerleştirilir, **amqsmnac** tarafından TARGET 'e aktarılır ve **amqsgnac** tarafından TARGET 'den alınır; bkz. Şekil 134 sayfa 1127.



Şekil 134. 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, Windows üzerinde, pencerelerden birinde her programı başlatan önceki komut istemine **start** üç kez yazın. “Kuyruk yöneticisinin ayarlanması ve denetlenmesi” sayfa 1128 içindeki “5” sayfa 1129 adımına bakın.)
5. Kuyruk yöneticisini durdurmak için `endmqm -r -p QM1` komutunu yazın ve istemcilerin yeniden bağlanmasına izin verin.
6. Kuyruk yöneticisini yeniden başlatmak için `strmqm QM1` komutunu yazın.

**amqsgnac**, **amqspnac** ve **amqsmnac** 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 1128. 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 amqspnac SOURCE QM1
C:\> start amqsmnac -s SOURCE -t TARGET -m QM1
C:\> start amqsgnac 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.
```

### amqspnac

```
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
```

### amqsmnac

```
Sample AMQSMHAC 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 AMQSMHAC end
C:\>
```

### amqsgnac

```
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

[amqmfsc](#) (dosya sistemi denetimi)

## Sorma örnek programları

Sorgula örnek programları, MQINQ çağrısını kullanan bir kuyruğun bazı özniteliklerine ilişkin bilgi içerir.

Bu programların adları için bkz. “Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094 .

Bu programların tetiklenen programlar olarak çalıştırılması amaçlanır; bu nedenle, tek girişi, IBM i, Windows, UNIX and Linux sistemleri için bir MQTMC2 (tetikleme iletisi) yapısıdır. Bu yapı, sorgulanacak özniteliklere sahip bir hedef kuyruğun adını içerir. C sürümü kuyruk yöneticisi adını da kullanır. COBOL sürümü, varsayılan kuyruk yöneticisini kullanır.

For the triggering process to work, ensure that the Inquire sample program that you want to use is triggered by messages arriving on queue SYSTEM.SAMPLE.INQ. To do this, specify the name of the Inquire sample program that you want to use in the *ApplicId* field of the process

definition SYSTEM.SAMPLE.INQPROCESS. **IBM i** IBM için, bu işlem için CHGMQMPCRC komutunu kullanabilirsiniz; ayrıntılar için Change MQ Process (CHGMQMPCRC)' e bakın. Örnek kuyruğun bir FIRST tetikleyicisi var; istek örneğini çalıştırmadan önce kuyruğunda önceden iletiler varsa, gönderdiğiniz iletiler sorgulamak için tetikleme örneği tetiklenmez.

Tanımlamayı doğru bir şekilde ayarladığınızda:

- **ULW** UNIX, Linux, and Windows için, bir oturumda **runmqtrm** programını başlatın ve ardından başka bir oturumda amqsreq programını başlatın.
- **IBM i** IBM için, bir oturumda AMQSERV4 programını başlatın ve ardından başka bir oturum içinde AMQSREQ4 programını başlatın. AMQSERV4 yerine AMQSTRG4 ' yi kullanabilirsiniz, ancak olası iş gönderimi gecikmeleri, gerçekleşenleri takip etmeyi daha az kolaylaştırabilirdi.

İstek örnek programlarını kullanarak, her biri yalnızca bir kuyruk adı içeren istek iletilerini SYSTEM.SAMPLE.INQ. Her istek iletisi için, Sorgula örnek programları, istek iletisinde belirtilen kuyruğa ilişkin bilgileri içeren bir yanıt iletisi gönderir. Yanıtlar, istek iletisinde belirtilen yanıtlama kuyruğuna gönderilir.

**IBM i** IBM üzerinde, örnek giriş kütüğü üyesi QMQMSAMP.AMQSDATA(INQ) kullanılıyorsa, adı belirtilen son kuyruk yok; bu nedenle, örnek başarısızlık için bir neden kodu içeren bir rapor iletisi döndürür.

## Sorgulamak için örnek program tasarımı

Program, tetikleme iletisi yapısında adı belirtilen kuyruğu açarken, bu işlem başlatıldığında iletileceği bir kuyruğa yer alan kuyruğu açar. (For clarity, we will call this the *istek kuyruğu*.) Program, bu kuyruğu paylaşılan giriş için açmak için MQOPEN çağrısını kullanır.

Program, bu kuyruktan iletileri kaldırmak için MQGET çağrısını kullanır. Bu çağrı, 5 saniye bekleme süresi ile MQGMO\_ACCEPT\_TRUNCATED\_MSG ve MQGMO\_WADET seçeneklerini kullanır. Program, bir istek iletisi olup olmadığını görmek için her iletinin tanımlayıcısını sınar; değilse, program iletisi atar ve bir uyarı iletisi görüntüler.

İstek kuyruğundan kaldırılan her istek iletisi için, program kuyruğun adını okur ( *hedef kuyruk* adını arayacağız). veri içeriyordu ve MQOO\_INQ seçeneğiyle MQOP çağrısını kullanarak kuyruğu açar. Daha sonra, program, hedef kuyruğun *InhibitGet*, **CurrentQDepth** ve **OpenInputCount** özniteliklerinin değerlerini sorgulamak için MQINQ çağrısını kullanır.

MQINQ çağrısı başarılı olursa, program yanıt kuyruğuna yanıt iletisi koymak için MQPUT1 çağrısını kullanır. Bu ileti, üç özniteliğin değerlerini içerir.

MQAUT ya da MQINQ çağrısı başarısız olursa, program yanıt kuyruğuna bir rapor iletisi koymak için MQPUT1 çağrısını kullanır. Bu rapor iletisinin ileti tanımlayıcısının *Feedback* alanında, başarısız olan buna bağlı olarak, MQOPED ya da MQINQ çağrısının döndürdüğü neden kodudur.

MQINQ çağrısından sonra program, MQCLOSE çağrısını kullanarak hedef kuyruğu kapar.

İstek kuyruğunda bir ileti kalmadığında, program o kuyruğu kapatır ve kuyruk yöneticisinden bağlantıyı keser.

### **Message Handle örnek programının Sorgusu Özellikleri**

AMQSIQMA, bir ileti kuyruğunun özelliklerini bir ileti kuyruğundan sorgulamak için kullanılan bir örnek C programıdır ve MQINQMP API çağrısının kullanılmasına bir örnektir.

Bu örnek, bir ileti tanıtıcısı yaratır ve bunu, MQGMO yapısının MsgHandle alanına yerleştirir. Daha sonra, örnek bir ileti alır ve ileti tanıtıcısının doldurulduğu tüm özellikleri sorgular 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**

The publish/subscribe sample programs demonstrate the use of the publish and subscribe features in IBM MQ.

IBM MQ yayınlama/abone olma arabirimine nasıl programlaşacağını gösteren üç adet C dili örnek programı vardır. Daha eski arabirimleri kullanan bazı C örnekleri vardır ve bunlar Java örnekleri vardır. Java örnekleri, com.ibm.mq.jariçindeki IBM MQ yayınlama/abone olma arabirimini ve com.ibm.mqjmsiçindeki JMS yayınlama/abone olma arabirimlerini kullanır. JMS örnekleri bu konuda kapsam dahilinde değildir.

## **C**

Find the publisher sample amqspub in the C samples folder. Bunu, ilk parametre olarak beğendiğiniz herhangi bir konu adıyla çalıştırın ve ardından isteğe bağlı bir kuyruk yöneticisi adını girin. Örneğin, amqspub mytopic QM3 . Ayrıca, amqspubcadlı bir istemci sürümü de vardır. İstemci sürümünü çalıştırmayı seçerseniz, ayrıntılar için önce [“Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102](#) ' i (bkz..) bakın.

Yayıncı, varsayılan kuyruk yöneticisine bağlanır ve çıkışa yanıt verir: target topic is mytopic . Bu pencereye şu andan itibaren girdiğiniz her satır, mytopic olarak yayınlanır.

Aynı dizinde başka bir komut penceresi açın ve aynı konu adını ve isteğe bağlı kuyruk yöneticisi adını belirterek abone programını amqssubçalıştırın. Örneğin, amqssub mytopic QM3 .

Abone çıkışa yanıt veriyor, Calling MQGET : 30 seconds wait time . Şu andan itibaren, yayıncıya yazdığınız satırlar abonenin çıkışında görünür.

Başka bir komut penceresinde başka bir abone başlatın ve her iki aboneyi de yayın olarak izleyin.

Ayar seçenekleri de içinde olmak üzere parametrelere ilişkin eksiksiz belgeler için örnek kaynak koduna bakın. Abone seçenekleri alanına ilişkin değerler şu konuda açıklanmıştır: [Options \(MQUZE\)](#).

Komut satırı anahtarları olarak ek abonelik seçenekleri sunan başka birsubscriberabone örneği amqssbxvardır.

Abone sona erdirildikten sonra alıkonan dayanıklı abonelikler kullanarak aboneyi çağırmak için amqssbx -d mysub -t mytopic -k yazın.

Yayıncıyı kullanarak başka bir öge yayınlayarak aboneliği test edin. Abonenin sona ermesi için 30 saniye bekleyin. Aynı konu altında birkaç öge daha yayınlayın. Aboneyi yeniden başlatın. Abone çalışmazken yayınlanan son öge, abone tarafından görüntülenerek hemen yeniden başlatılır.

## **C mirası**

Kuyruğa alınan komutları gösteren ek bir C örneği kümesi vardır. Bu örneklerden bazıları başlangıçta MQOC Supportpac 'in bir parçası olarak gönderilmişti. Uyumluluk nedenleri için, örneklerin tam olarak desteklendiği yeteneklerdir.

Kuyruklanan komut arabirimini kullanmaktan vazgeçmenizi öneriyoruz. Yayınlama/abone olma API 'sinden çok daha karmaşıktır ve karmaşık kuyruğa alınmış komutları programlamak için zorlayıcı bir işlev nedeni yoktur. Ancak, kuyruğa yollanan yaklaşımı daha uygun bulabilir, belki de arayüzü kullanmakta olduğunuz için ya da programlama ortamınız karmaşık bir ileti oluşturmanızı ve MQSUB ' ye farklı çağrılar oluşturmak yerine sosyal bir MQPUT adını koymanızı kolaylaştırır.

Ek örnekler, samples klasöründeki pubsub alt dizininde bulunur.

Çizelge 170 sayfa 1132' ta listelenen altı tür örnek vardır.

<i>Çizelge 170. Eski yayınlama/abone olma örnek C programlarının kategorileri</i>		
<b>Kategori</b>	<b>Programlar</b>	<b>Açıklamalar</b>
RFH1	amqssr1a.c amqspr1a.c	Simple publish/subscribe example built using RFH1 format messages.
RFH2	amqssr2a.c amqspr2a.c	Simple publish/subscribe example built using RFH2 format messages.
MQAI örnekleri	amqsppca.c amqsspca.c	PCF komutları ve MQAI komut arabirimi kullanılarak oluşturulan basit yayınlama/abone olma örneği.
MAOC Results service using RFH1	amqsgama.c amqsresa.c	Results service built using RFH1 headers 1. Requires the queues defined in amqsgama.tst and amqsresa.tst 2. amqsresa must be started before amqsgama
MAOC RFH2kull anılarak sonuç hizmeti	amqsgr2a.c amqsrr2a.c	Results service built using RFH2 headers 1. Requires the queues defined in amqsgama.tst and amqsresa.tst 2. amqsresa must be started before amqsgama
Yönlendirme çıkışı yayınlama/abone olma örneği	amqspssa.c	Bir yönlendirme çıkışta yayınlama/abone olma ileti için 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 örnekleri klasöründe bulunur.

If you choose to run in client mode, first see [“Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102](#) for details.

Yapılandırılmış bir Java ortamınız varsa, örneği Java komutunu kullanarak komut satırından çalıştırın. Ayrıca, örneği IBM MQ Explorer Eclipse çalışma alanında, Java programlama çalışma ortamı önceden ayarlanmış bir örnek de çalıştırabilirsiniz.

Örnek programın özelliklerini değiştirmek için bazı özelliklerin bazılarını değiştirmeniz gerekebilir. Bunu, JVM ' ye parametreler sağlayarak ya da kaynak olarak düzenleme yaparak yapabilirsiniz.

[“MQPubSubApiSample Java örneğinin çalıştırılması” sayfa 1133](#) içindeki yönergeler, örneğin Eclipse çalışma alanından nasıl çalıştırılacağını gösterir.

*MQPubSubApiSample Java örneğinin çalıştırılması*

Eclipse platformundan Java Geliştirme Araçları 'nı kullanarak MQPubSubApiSample ' nin nasıl çalıştırılacağı.

## Başlamadan önce

Eclipse çalışma ortamını açın. Yeni bir çalışma alanı dizini yaratın ve seçin. Hoş geldiniz penceresini kapatın.

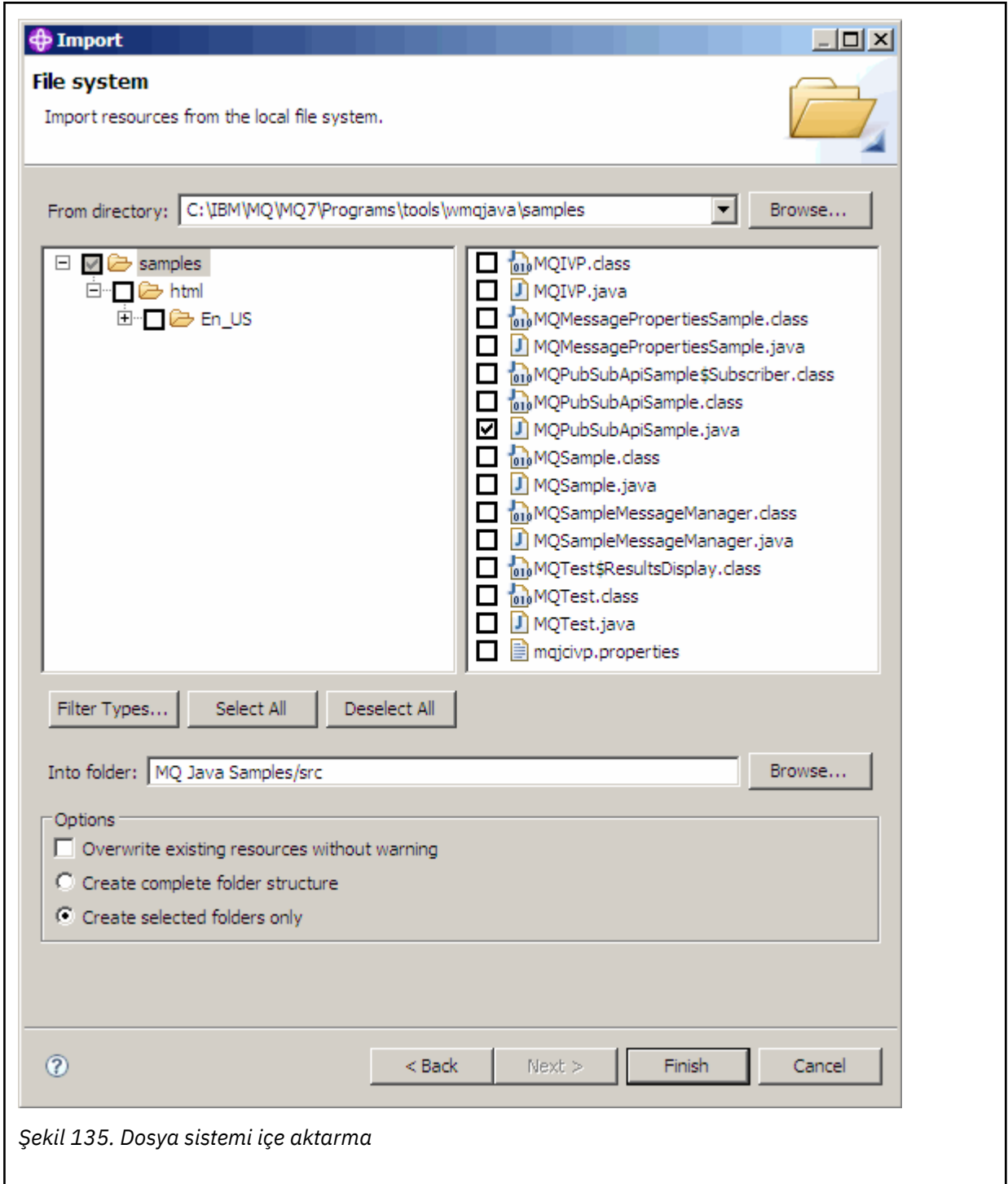
İstemci olarak çalışmadan önce “Çoklu Platformlar üzerinde istemci bağlantılarını kabul etmek için kuyruk yöneticisi yapılandırılması” sayfa 1102 içindeki adımları izleyin.

## Bu görev hakkında

Java publish/subscreen örnek programı bir IBM MQ MQI client Java programıdır. Bu örnek, 1414 numaralı bağlantı noktasında bulunan varsayılan kuyruk yöneticisi kullanılarak değişiklik yapılmaksızın çalışır. Bu görev, bu basit vakayı açıklar ve genel olarak, parametrelerin nasıl sağlanacağını ve örneklerin farklı IBM MQ yapılandırmalarına uygun olarak nasıl değiştirileceğini belirtir. Bu örnek, Windows üzerinde gösterilen şekilde gösterilmektedir. Dosya yolları diğer platformlarda farklılık gösterir.

## Yordam

1. Java örnek programlarını içe aktar
  - a) Çalışma ortamında, **Pencere > Perspektifi aç > Diğer > Java** öğelerini tıklattıktan sonra **Tamam** düğmesini tıklatın.
  - b) **Paket Gezgini** görünümüne geçin.
  - c) **Package Explorer** (Paket Gezgini) görünümünde yer alan beyaz alan için sağ tıklatın. **Yeni > Java projesi** seçeneğini tıklatın.
  - d) **Project name** alan tipinde MQ Java Samples. **İleridüğmesini** tıklatın.
  - e) **Java Settings** panelinde, **Libraries** (Kitaplıklar) sekmesine geçin.
  - f) **Dış JAR Ekle** seçeneğini tıklatın.
  - g) Browse to `MQ_INSTALLATION_PATH\java\lib` where `MQ_INSTALLATION_PATH` is the IBM MQ installation folder and select `com.ibm.mq.jar` and `com.ibm.mq.jmqi.jar`
  - h) **Aç > Bitir** öğesini tıklatın
  - i) **Paket Gezgini** görünümünde `src` öğesini farenin sağ düğmesiyle tıklatın.
  - j) Seç **İçe Aktar ... > Genel > Dosya Sistemi > Sonraki > Göz At...** and browse to the path `MQ_INSTALLATION_PATH\tools\wmqjava\samples` where `MQ_INSTALLATION_PATH` is the IBM MQ installation directory.
  - k) **İçe Aktar** panosunda, Şekil 135 sayfa 1134, `samples` seçeneğini tıklatın (onay kutusunu seçmeyin).
  - l) `MQPubSubApiSample.jav` seçeneğini belirleyin. **Into folder** alanı MQ Java Samples/ `src` içermelidir. **Bitir**'i tıklatın.



Şekil 135. Dosya sistemi içe aktarma

2. Yayınlama/abone olma örnek programını çalıştırın.

Varsayılan parametreleri değiştirmenize gerek olup olmadığına bağlı olarak, programı çalıştırmanın iki yolu vardır.

- İlk seçenek, herhangi bir değişiklik yapmadan programı çalıştırır:
  - Çalışma alanı ana menüsünde, src klasörünü genişletin. **MQPubSubApiSample.java Run-as > 1 seçeneğini sağ tıklayın. Java Application**
- İkinci seçenek, programı değiştirgelerle ya da ortamınıza ilişkin değiştirilmiş kaynak kodla çalıştırır:
  - MQPubSubApiSample.java 'ı açın ve MQPubSubApiSample oluşturucusunu çalıştırın.
  - Programın özniteliklerini değiştirin.

Bu öznitelikler, -D JVM anahtarı kullanılarak değiştirilebilir ya da kaynak kodu düzenleyerek System property için varsayılan bir değer sağlar.

- topicObject
- queueManagerAdı
- subscriberCount

Bu öznitelikler yalnızca oluşturucudaki kaynak kod düzenlenerek değiştirilebilir.

- hostname
- port
- channel

System protalarını ayarlamak için, erişimcide varsayılan bir değer kodunu kodlayın; örneğin:

```
queueManagerName = System.getProperty("com.ibm.mq.pubSubSample.queueManagerName",  
"QM3");
```

Or provide the parameter to the JVM using the -D option, as shown in the following steps:



- Ayarlamak istediğiniz System.Property 'nin tam adını kopyalayın, örneğin:  
`com.ibm.mq.pubSubSample.queueManagerName`.
- Çalışma alanında, **Çalıştır > Çalıştır İletişim Kutularını Aç** 'ı sağ tıklayın. **Uygulamaları Oluştur, Yönet ve Çalıştır** 'da Java Uygulaması 'nı çift tıklayın ve **(x) = Bağımsız Değişkenler** sekmesini tıklayın.
- VM bağımsız değişkenleri:** bölmesinde, -D yazın ve System.property adını, `com.ibm.mq.pubSubSample.queueManagerName`ve ardından =QM3adını yapıştırın. **Uygula > Çalıştır**seçeneğini tıklayın.
- Virgülle ayrılmış bir liste olarak ya da bölmede ek satırlar olarak, virgül ayırıcılar olmadan ek bağımsız değişkenler ekleyin.  
Örneğin: `-Dcom.ibm.mq.pubSubSample.queueManagerName=QM3,  
-Dcom.ibm.mq.pubSubSample.subscriberCount=6`.

### **Publish Exit örnek programı**

AMQSPSE0 , bir aboneye teslim edilmeden önce yayını kesmek için çıkılan bir çıkışa ait örnek bir C programıdır. Bundan sonra çıkış, örneğin ileti üstbilgilerini, bilgi yükünü ya da hedefi değiştirebilir ya da iletinin bir aboneye yayınlanmasını engelleyebilir.


Örneği çalıştırmak için aşağıdaki görevleri gerçekleştirin:

1. Kuyruk yöneticisini yapılandır:

-   UNIX and Linux sistemlerinde, `qm.ini` dosyasına benzer bir stanza ekleyin:

```
PublishSubscribe:  
PublishExitPath=Module  
PublishExitFunction=EntryPoint
```

Burada modül `MQ_INSTALLATION_PATH/samp/bin/amqspse.MQ_INSTALLATION_PATH` , IBM MQ 'in kurulu olduğu üst düzey dizini temsil eder.

-  Windows üzerinde eşdeğer öznitelikleri kayıt defterine ayarlayın.
2. Modülün IBM MQtarafından erişilebilir olduğundan emin olun.
  3. Yapılandırmayı almak için kuyruk yöneticisini yeniden başlatın.
  4. İzlenecek uygulama sürecinde, izleme dosyalarının nereye yazılacağı açıklanmalıdır. Örneğin:

- **Linux** **UNIX** UNIX and Linux sistemlerinde, /var/mqm/trace dizininin var olduğundan ve aşağıdaki ortam değişkenini dışa aktardığından emin olun:

```
export MQPSE_TRACE_LOGFILE=/var/mqm/trace/PubTrace
```

- **Windows** Windows' ta, C:\temp dizininin var olduğundan ve aşağıdaki ortam değişkenini ayarlandığından emin olun:

```
set MQPSE_TRACE_LOGFILE=C:\temp\PubTrace
```

### **Put Sample programlar**

Put örnek programları, MQPUT çağrısını kullanan bir kuyruğa ileti yerleştirdi.

Bu programların adları için bkz. [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#).

### **Put Sample programının tasarımı**

Program, iletileri koymak için hedef kuyruğu açmak için MQOO\_Output seçeneğiyle MQOPER çağrısını kullanır.

Kuyruk açılmazsa, program, MQOPED çağrısının döndürdüğü neden kodunu içeren bir hata iletilisi görüntüler. Programı basit tutmak için, bu konuda ve sonraki MQI çağrılarında, program seçeneklerin çoğu için varsayılan değerleri kullanır.

Her giriş satırı için, program metni bir arabelleğe okur ve MQPUT çağrısını kullanarak, o satırın metnini içeren bir veri paketi iletilisi yaratır. Program, girişin sonuna ulaşıncaya kadar ya da MQPUT çağrısının başarısız olduğu zamana kadar devam eder. Program girişin sonuna ulaşırsa, MQCLOSE çağrısını kullanarak kuyruğu kapatır.

*Put örnek programlarını çalıştırma*

### **amqspust ve amqspustc örneklerinin çalıştırılması**



Amqspust örneği, iletileri yerel bağ tanımlarını kullanarak yerleştirmek için kullanılan bir programdır ve amqspustc örneği, istemci bağ tanımlarını kullanmak için kullanılan bir programdır. Bu programların her biri aşağıdaki konumsal parametreleri alır:

1. Hedef kuyruğun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)

Bir kuyruk yöneticisi belirtilmezse, amqspust varsayılan kuyruk yöneticisine bağlanır ve amqspustc, [MQSERVER](#) ortam değişkeniyle ya da istemci kanal tanımlama dosyasıyla tanımlanan kuyruk yöneticisine bağlanır.

3. Açma seçenekleri (isteğe bağlı)

Açma seçenekleri belirtilmezse, örnek şu iki seçeneğin birleşiminden oluşan 8208 değerini kullanır:

- MQOO\_ÇIKIŞI
- 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.

Bu programlar, bağlantı kimlik doğrulaması için kullanılacak kullanıcı kimliğine ayarlanması gereken **MQSAMP\_USER\_ID** adlı bir ortam değişkeni de kullanır. Bu seçenek belirlendiğinde, program bu kullanıcı kimliğine eşlik edecek bir parola ister.

Bu programları çalıştırmak için aşağıdakilerden birini girin:

- amqspud myqueue qmanagername
- amqspudc myqueue qmanagername

Burada myqueue , iletilerin konacağı kuyruğun adı ve qmanagername , myqueue' un sahibi olan kuyruk yöneticisidir.

## amq0put örneğinin çalıştırılması



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 )



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
```

QMMSAMP AMQSDATA dosyası PUT kitaplığında, koyma örneklerine ilişkin bir giriş örneği sağlandı.

**Not:** Kuyruk adlarının büyük ve küçük harfe duyarlı olduğunu unutmayın. AMQSPUT4 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('QMMSAMP/AMQSDATA(PUT)')
```

## AMQ0PUT4 COBOL örneğini çalıştırma ( 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 iletili yaratır. Klavyede boş bir satır girdiğinizde program durur.

## Reference Message Sample programlar

Başvuru İletisi örnekleri, büyük bir nesnenin bir düğümden diğerine (genellikle farklı sistemlerde) aktarılmasını sağlar (genellikle, kaynak ya da hedef düğümlerde bulunan nesne IBM MQ kuyruklarında saklanmasına gerek kalmaksızın).

Başvuru İletilerinin bir kuyruğa nasıl konabileceğini, ileti çıkışları tarafından alınan ve bir kuyruktan nasıl alınabileceğini göstermek için bir dizi örnek program sağlanır. Örnek programlar, dosyaları taşımak için Başvuru İletilerini kullanır. Veritabanları gibi diğer nesnelere taşımak ya da güvenlik denetimlerini gerçekleştirmek istiyorsanız, örneğe dayalı olarak kendi çıkışınızı tanımlayın.

Kullanılacak Reference Message exit örnek programının sürümü, kanalın üzerinde çalışmakta olduğu altyapıya bağlıdır:

- Tüm altyapılarda, gönderme bitişindeki amqsxrma kullanın.
- Alıcı, IBM dışında herhangi bir platform altında çalışıyorsa, alma uçta amqsxrma 'yı kullanın.
- **IBM i** Günlük nesnesi IBM i altında çalışıyorsa, amqsxrm4öğesini kullanın.

### **IBM i** IBM i kullanıcıları için notlar

Örnek ileti çıkışı kullanarak bir Başvuru İletisi almak için, bir akış dosyasının yaratılabilmesi için, IFS 'nin kök dosya sisteminde ya da herhangi bir alt dizinde bir dosya belirtin.

IBM i üzerindeki örnek ileti çıkışı, dosyayı yaratır, verileri EBCDIC 'e dönüştürür ve kod sayfasını sistem kod sayfası olarak ayarlar. Daha sonra, CPYFRMSTMF komutunu kullanarak bu dosyayı QSYS.LIB dosya sistemine kopyalayabilirsiniz. Örneğin:

```
CPYFRMSTMF FROMSTMF('JANEP/TEST.TXT')
TOMBR('qsys.lib.janep.lib/test.fie/test.mbr') MBRPT(*REPLACE)
CVTDTA(*NONE)
```

CPYFRMSTMF komutu kütüğü yaratmaz. Bu komutu çalıştırmadan önce bunu yaratmanız gerekir.

QSYS.LIB, örneklerde herhangi bir değişiklik yapılması gerekmez. Diğer herhangi bir kütük sistemi için, MQRMH yapısındaki CodedCharSetId alanında belirlenen CCSID 'nin göndermekte olduğunuz toplu verilerle eşleştiğinden emin olun.

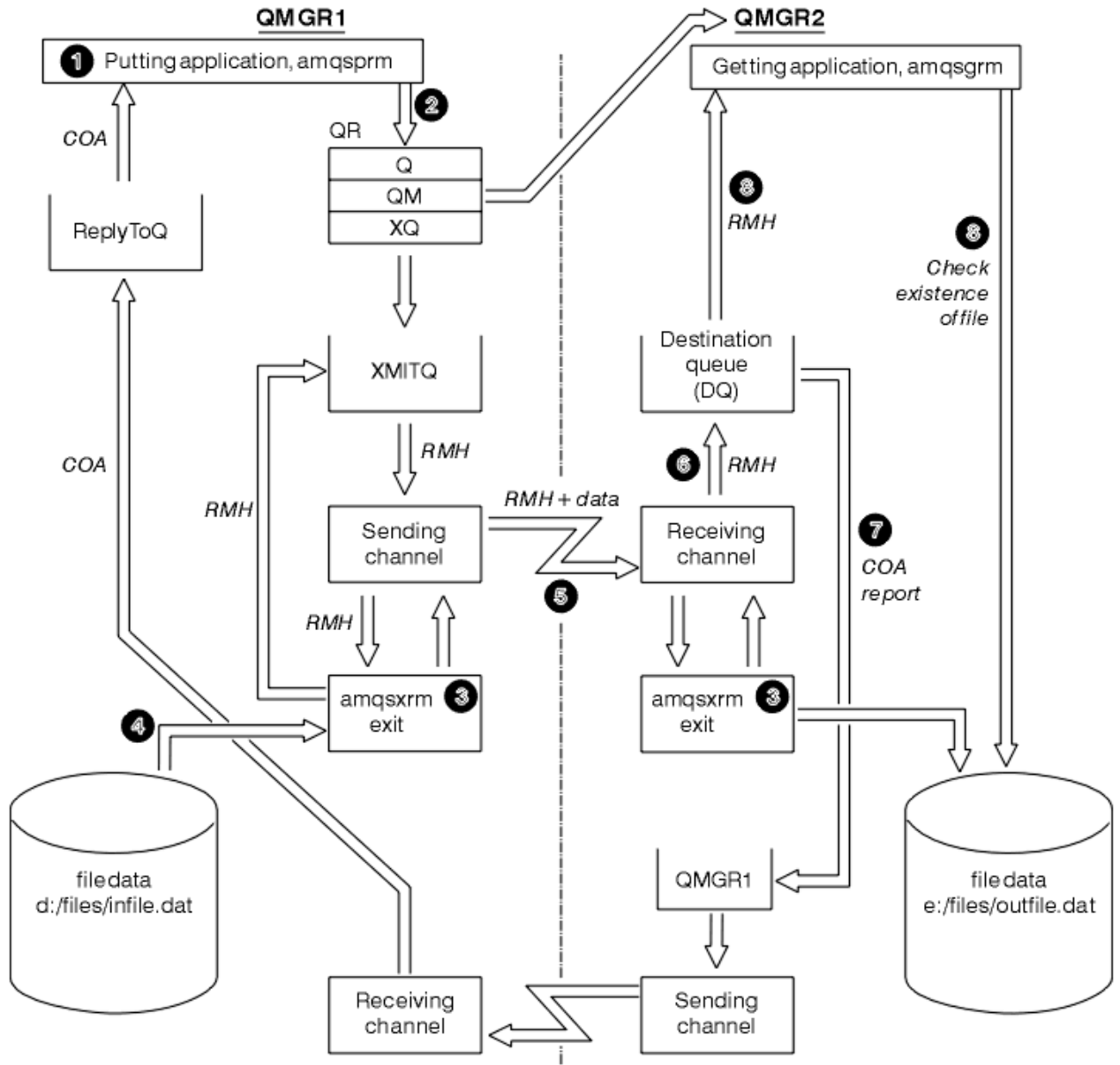
Tümleşik dosya sistemini kullanırken, SYSIFCOPT (\*IFSIO) seçenek kümesiyle program modülleri yaratın. If you want to move database or fixed-length record files, define your own exit based on the supplied sample AMQSRM4.

Bir veri tabanı dosyasının aktarılmasına ilişkin önerilen yöntem, CPYTOSTMF komutunu kullanarak, bu dosyayı IFS yapısına dönüştürerek IFS dosyasını bağlarken Reference Message (Başvuru İletisi) ögesini göndermektedir. Bir veri tabanı dosyasını IFS içinden gönderme yaparak aktarmayı seçerseniz, ancak bunu IFS yapısına dönüştürmeyin, üye adını belirtmeniz gerekir. Bu yöntemi seçerseniz, veri bütünlüğü garanti edilmez.

### **Multi** Başvuru İletisi örneklerinin çalıştırılması

Use this example to find out how to run Reference Message sample application AMQSPRM on UNIX, Linux, and Windows, or AMQSPRMA on IBM i. Bu örnekte, Başvuru İletilerin bir kuyruğa nasıl konabileceği, ileti çıkışlar tarafından alınan ve bir kuyruktan nasıl alınabileceği gösterilmektedir.

Reference Message Samples komutu aşağıdaki gibi çalışır:



Şekil 136. Başvuru İletisi örneklerinin çalıştırılması

1. Dinleyicileri, kanalları ve tetikleme izleyicilerini başlatmak ve kanallarınızı ve kuyruklarınızı tanımlamak için ortamı ayarlayın.

For the purposes of describing how to set up the Reference Message, this example refers to the sending machine as MACHINE1 with a queue manager called QMGR1 and the receiving machine as MACHINE2 with a queue manager called QMGR2.

**Not:** The following definitions allow a Reference Message to be built to send a file with an object type of FLATFILE from queue manager QMGR1 to QMGR2 and to re-create the file as defined in the call to AMQSPRM (ya da IBM üzerinde AMQSPRMA). Başvuru İletisi (dosya verileri de içinde olmak üzere), kanal CHL1 ve iletim kuyruğu XMITQ kullanılarak gönderilir ve kuyruk DQ 'ya yerleştirilir. Exception and COA reports are sent back to QMGR1 using the channel REPORT and transmission queue QMGR1.

Başvuru İletisi 'ni alan uygulama (AMQSGRM ya da AMQSGRMA on IBM i ), INITQ başlatma kuyruğu ve PROC işlemi kullanılarak tetiklenir. CONNAME alanlarının doğru olarak ayarlandığından ve MSGEXIT alanının, makine tipine ve IBM MQ ürününün kurulu olduğu yere bağlı olarak dizin yapılarınızı yansıttığınızdan emin olun.

**IBM i**

MQSC tanımlamaları çıkışların tanımlanması için bir AIX biçemi kullandıysa, bu nedenle IBM üzerinde MQSC kullanıyorsanız, bunları uygun şekilde değiştirmeniz gerekir. FLATFILE ileti veri FLATFILE 'in büyük ve küçük harfe duyarlı olduğunu ve büyük harfle çalışmadığı sürece çalışmayacağını unutmayın.

Makine üzerinde MACHINE1, 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:** 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/AMQSXR4) 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)
```

Makine üzerinde MACHINE2, 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/amqsgm')

define ql(dq) initq(initq) process(proc) trigger trigtype(first)
```

**IBM i****IBM i komut sözdizimi**

**Not:** **IBM i** On IBM i, if you do not specify a queue manager name the system uses the default queue manager.

```
CRTMQMCHL CHLNAME(CHL1) CHLTYPE(*RCVR) MQMNAME(QMGR2) +
REPLACE(*YES) TRPTYPE(*TCP) +
MSGEXIT(QMQM/AMQSXR4) 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)

CRTMQMPC 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 nesnelere yaratıldıktan sonra:

- Platforma uygun olduğunda, gönderen ve alıcı kuyruk yöneticilerine ilişkin dinleyiciyi başlatın
  - Start the channels CHL1 and REPORT
  - Alıcı kuyruk yöneticisinde, INITQ başlatma kuyruğu için tetikleme izleyicisini başlatır
3. Aşağıdaki parametreleri kullanarak, komut satırında Referans İletisi örnek programını AMQSPRM (AMQSPRMA on IBM i) komutunu çalıştırın:

**-m**

Yerel kuyruk yöneticisinin adı; varsayılan kuyruk yöneticisi varsayılan değer olarak kullanılır

**-i**

Kaynak dosyanın adı ve yeri

**-o**

Hedef dosyanın adı ve yeri

**-q**

Kuyruğun adı

**-g**

-q parametresinde tanımlanan kuyruğun var olduğu kuyruk yöneticisinin adı. Bu varsayılan değer olarak, -m parametresinde belirlenen kuyruk yöneticisi kullanılır.

**-t**

Nesne tipi

**-w**

Bekleme aralığı, yani, kural dışı durum için bekleme süresi ve alıcı kuyruk yöneticisinden COA raporları

Örneğin, önceden tanımlanan nesnelere birlikte örneği kullanmak için aşağıdaki parametreleri kullanırdınız:

```
-mQMGR1 -iInput File -oOutput File -qQR -tFLATFILE -w120
```

Bekleme süresini artırmak, büyük bir dosyanın, iletileri zamanlaşımına yatırmadan önce bir ağ üzerinden gönderilmesine olanak tanır.

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

**IBM i kullanıcıları:**  IBM işletim sisteminde aşağıdaki adımları tamamlayın:

- Aşağıdaki komutu kullanın:

```
CALL PGM(QMQM/AMQSPRM4) PARM('-mQMGR1' +
'-i/refmsg/irmsg1' +
'-o/refmsg/irmsgx' '-qQR' +
'-gQMGR1' '-tFLATFILE' '-w15')
```

This assumes that the original file `irmsg1` is in IFS directory `/refmsg` and that you want the destination file to be `irmsgx` in IFS directory `/refmsg` on the target system.

- b. Kök dizini kullanmak yerine CRTDIR komutunu kullanarak kendi dizininizi yaratın.
- c. Verileri yerleştiren programı çağırdığınızda, çıkış dosyası adının IFS adlandırma kuralını yansıtması gerektiğini unutmayın; örneğin, /test/kütükadı, TEST dizininde FILENAME adlı bir dosya yaratır.

**Not:**

**IBM i** IBM i' ta, parametreleri belirtirken sağa eğik çizgi (/) ya da tire (-) kullanabilirsiniz.

Örneğin:

```
amqsprn /i d:\files\infile.dat /o e:\files\outfile.dat /q QR
/m QMGR1 /w 30 /t FLATFILE
```

**Linux** **UNIX** UNIX and Linux platformları için, hedef dosya dizinini belirtmek için bir yerine iki ters eğik çizgi (\\) kullanmanız gerekir. Bu nedenle, **amqsprn** komutu şu şekilde görünür:

```
amqsprn -i /files/infile.dat -o e:\\files\\outfile.dat -q QR
-m QMGR1 -w 30 -t FLATFILE
```

Conference Message programının çalıştırılması aşağıdaki gibi olur:

- Başvuru İletisi, QMGR1kuyruk yöneticisine QR kuyruğuna konmaya neden oldu.
  - Kaynak dosya ve yol d:\files\infile.dat ' dir ve örnek komutun verildiği sistemde bulunur.
  - QR kuyruğu uzak bir kuyruksa, Başvuru İletisi başka bir kuyruk yöneticisine, farklı bir sistemde (e:\files\outfile.dat) ve yolu ile yaratılan bir dosyanın bulunduğu bir kuyruk yöneticisine gönderilir). Bu dosyanın içeriği kaynak dosya ile aynı.
  - amqsprn, hedef kuyruk yöneticisinden bir COA raporu için 30 saniye bekler.
  - Nesne tipi flatfile olduğundan, iletileri kuyruktan QR kuyruğundan taşımak için kullanılan kanal bunu *MsgData* alanında belirtmelidir.
4. Kanallarınızı tanımladığınızda, amqsprn olacak şekilde hem gönderen hem de alma uçlarında ileti çıkışı seçin.

**Windows** Bu, Windows üzerinde aşağıdaki şekilde tanımlıdır:

```
msgexit(' pathname\amqsprn.dll(MsgExit)')
```

**Solaris** **AIX** Bu, AIX ve Solaris üzerinde aşağıdaki gibi tanımlıdır:

```
msgexit(' pathname/amqsprn(MsgExit)')
```

Bir yol adı belirtiyorsanız, tam adı belirtin. If you omit the path name, it is assumed that the program is in the path specified in the qm.ini file (or, on IBM MQ for Windows, the path specified in the registry).

5. Kanal çıkışı, Başvuru İletisi üstbilgisini okur ve başvurduğu dosyayı bulur.
6. Daha sonra, kanal çıkışı, üstbilgiyle birlikte kanalı aşağı göndermeden önce dosyayı bölülebilir.

**Solaris** **AIX** AIX ve Solaris üzerinde, hedef dizinin grup sahibini 'mqm' olarak değiştirin, böylece örnek ileti çıkışı bu dosyayı o dizinde yaratabilir. Ayrıca, hedef dizinin izinlerini, mqm grubu üyelerinin bu gruba yazmasına izin verecek şekilde değiştirin. Dosya verileri IBM MQ kuyruklarında saklanmaz.

7. Dosyanın son bölümü alan ileti çıkışı tarafından işlendiğinde, Başvuru İletisi, amqsprn tarafından belirlenen hedef kuyruğa konmaktadır. Bu kuyruk tetiklenirse (yani, tanımlama **Trigger, InitQve 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) eriştiğinde, bir COA raporu koyma uygulamasına (amqsprn) geri gönderilir.

9. Get Reference Message Sample, amqsgm, giriş tetikleyicisi iletilerinde belirtilen kuyruktan ileti alır ve kütüğün var olup olmadığını denetler.

#### *Put Reference Message Sample tasarımı (amqsprma.c, AMQSPRM4)*

Bu konu, bir Put Reference İleti örneğine ilişkin ayrıntılı bir açıklama verir.

Bu örnek, bir dosyaya gönderme yapan ve dosyayı belirtilen bir kuyruğa koyan bir Başvuru İletisi oluşturur:

1. Bu örnek, MQCONN kullanarak yerel bir kuyruk yöneticisine bağlanır.
2. Daha sonra, rapor iletilerini almak için kullanılan bir model kuyruğunu açar (MQOPER).
3. Örnek, dosyayı taşımak için gerekli olan değerleri (örneğin, kaynak ve hedef dosya adları ve nesne tipi) içeren bir Başvuru İletisi oluşturur. Örnek olarak, IBM MQ ile verilen örnek, d:\x\file.in dosyasını QMGR1 'dan QMGR2 ' a göndermek ve aşağıdaki parametreleri kullanarak dosyayı d:\y\file.out olarak yeniden oluşturmak için bir Başvuru İletisi oluşturur:

```
amqsprm -q QR -m QMGR1 -i d:\x\file.in -o d:\y\file.out -t FLATFILE
```

Burada QR , QMGR2 üzerinde bir hedef kuyruğa gönderme yapan uzak bir kuyruk tanımlamasıdır.

**Not:** UNIX and Linux altyapılarında, hedef dosya dizinini belirtmek için, bir yerine iki ters eğik çizgi (\\) kullanın. Bu nedenle, **amqsprm** komutu şu şekilde görünür:

```
amqsprm -q QR -m QMGR1 -i /x/file.in -o d:\\y\\file.out -t FLATFILE
```

4. Başvuru İletisi, /q parametresi tarafından belirtilen kuyruğa (herhangi bir dosya verisi olmadan) yerleştirilir. Bu bir uzak kuyrukta, ileti ilgili iletim kuyruğuna konabilir.
5. Örneğin, /w parametresinde belirtilen süre (varsayılan olarak 15 saniye olarak), COA raporlarında belirtilen süre boyunca, yerel kuyruk yöneticisinde (QMGR1) yaratılan dinamik kuyruğa geri gönderilmek üzere COA raporları için bekleme süresi boyunca bekler.

#### *Reference Message Exit örneğinin tasarımı (amqsxrma.c, AMQSXRM4)*

Bu örnek, kanal tanımlamasının ileti çıkışı kullanıcı veri alanındaki nesne tipiyle eşleşen bir nesne tipine sahip Başvuru İletilerini tanımlar.

Bu iletiler için aşağıdakiler gerçekleşir:

- Gönderen ya da sunucu kanalındaki belirtilen veri uzunluğu, belirtilen dosyanın belirtilen görel konumundan, Başvuru İletisi 'nden sonra aracı arabelleğindeki boşluğa kopyalanır. Dosyanın sonuna ulaşılmamışsa, Başvuru İletisi, *DataLogicalOffset* alanını güncelledikten sonra iletim kuyruğuna geri konmaktadır.
- İstekte bulunan ya da alıcı kanalında, *DataLogicalOffset* alanı sıfırsa ve belirtilen dosya yoksa, yaratılır. Başvuru İletisi 'nin ardından gelen veriler, belirtilen dosyanın sonuna eklenir. Başvuru İletisi belirtilen dosya için son ileti değilse, atılır. Ters durumda, hedef kuyruğa konabilmek için sonuna veri eklenmeden kanal çıkışa döndürülür.

Gönderen ve sunucu kanalları için, giriş başvuru iletilerinde *DataLogicalLength* alanı sıfırsa, dosyanın geri kalan kısmı, *DataLogicalOffset* ' dan dosyanın sonuna kadar, kanal boyunca gönderilir. Bu değer sıfır değilse, yalnızca belirlenen uzunluk değeri gönderilir.

Bir hata oluşursa (örneğin, örnek bir dosyayı açamıyorsa), MQCXP. *ExitResponse* , işlenmekte olan iletilerin hedef kuyruğa devam etmek yerine, ölüm-mektup kuyruğuna konması için MQXCC\_SUPPRESS\_FUNCTION değerine ayarlıdır. MQCXP ' de bir geribildirim kodu döndürülür. *Feedback* ve iletiyi, bir rapor iletilerinin ileti tanımlayıcısının *Feedback* alanına koyan uygulamaya geri döndürüldü. Bunun nedeni, MQMD ' nin *Report* alanında MQRO\_EXCEPTION ayarını MQRO\_EXCEPTION belirleyerek kural dışı durum raporları isteğinde bulunmasıdır.

If the encoding or *CodedCharacterSetId* (CCSID) of the Reference Message is different from that of the queue manager, the Reference Message is converted to the local encoding and CCSID. Örneğimizde, amqsprm, nesnenin biçimi MQFMT\_STRING biçimidir; bu nedenle amqsxrm, veriler dosyaya yazılmadan önce nesne verilerini alan uçtaki yerel CCSID ' ye dönüştürür.

Dosya çok baytlı karakterler (örneğin, DBCS ya da Unicode) içeriyorsa, bu dosyanın MQFMT\_STRING olarak aktarılmakta olduğunu belirtmeyin. Bunun nedeni, dosyanın gönderilmesi sırasında birden çok byte 'lık bir karakterin bölünebilmesinden kaynaklanır. Bu tür bir dosyayı aktarmak ve dönüştürmek için, biçimi MQFMT\_STRING dışında bir değer olarak belirtin; böylece, Reference Message exit (Başvuru İletisi çıkışı) işlevi dönüştürmez ve aktarma tamamlandığında dosyayı alıcı uçta dönüştürmesini sağlar.

#### *Reference Message Exit örneğinin derlenmesi*

Reference Message Exit örneğini derlemek için, IBM MQ ' in kurulu olduğu altyapıya ilişkin komutu kullanın.

`MQ_INSTALLATION_PATH` , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

`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/AMQSRMA) SRCFILE(QMQMSAMP/QCSRC)  
TERASPACE(*YES *TSIFC)
```

### Not:

1. Biriminizi, IFS dosya sistemini kullanacak şekilde oluşturmak için SYSIFCOPT(\*IFSIO) seçeneğini ekleyin.
2. İş parçacıklı olmayan kanallarla kullanmak üzere programı yaratmak için şu komutu kullanın: CRTPGM PGM(MYLIB/AMQSRMA) BNDSRVPGM(QMQM/LIBMQM)
3. İş parçacıklı kanallarla kullanmak üzere programı yaratmak için şu komutu kullanın: CRTPGM PGM(MYLIB/AMQSRMA) BNDSRVPGM(QMQM/LIBMQM\_R)

## AçıkLinux

Linux

```
$ gcc -m64 -shared -fPIC -o /var/mqm/exits64/amqsxrm amqsqrma.c -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=MQ_INSTALLATION_PATH/lib64 -Wl,-rpath=/usr/lib64  
-lmqm_r
```

## AçıkSolaris

Solaris

```
$ cc -xarch=v9 -mt -G -o /var/mqm/exits64/amqsxrm amqsqrma.c -I MQ_INSTALLATION_PATH/inc  
-L MQ_INSTALLATION_PATH/lib64 -R MQ_INSTALLATION_PATH/lib64 -R/usr/lib/64 -lmqm
```

```
-lsocket  
-lnsl -ldl
```



## AçıkWindows

**Windows** IBM MQ şimdi mqm kitaplığını istemci paketleriyle birlikte sunucu paketleriyle birlikte sağlar; bu nedenle, aşağıdaki örnek mqm.vx.lib yerine mqm.lib kullanır:

```
cl amqsqrma.c /link /out:amqsxrm.dll /dll mqm.lib mqm.lib /def:amqsxrm.def
```

### İlgili kavramlar

[“Kanal-çıkış programları yazılıyor” sayfa 996](#)

Kanal çıkış programları yazmanıza yardımcı olması için aşağıdaki bilgileri kullanabilirsiniz.

*Get Reference Message Sample (amqsgrma.c, AMQSGRM4) tasarımı*

Bu konuda, Get Reference Message örneğinin tasarımı açıklanmaktadır.

Program mantığı aşağıdaki gibidir:

1. Örnek tetiklenir ve giriş tetikleyicisi ileten kuyruk ve kuyruk yöneticisi adlarını alır.
2. Daha sonra MQCONN kullanarak belirlenen kuyruk yöneticisine bağlanır ve MQOPED kullanarak belirtilen kuyruğu açar.
3. Örnek sorular, kuyruktan ileti almak için bir döngü içinde 15 saniye bekleme aralığıyla MQGET ' i yayınlar.
4. İleti bir Başvuru İletisi ise, örnek aktarılmış olan dosyanın varlığını denetler.
5. Daha sonra kuyruk kapatılır ve kuyruk yöneticisinden bağlantıyı keser.

### İstek örnek programları

İstek örnek programları, istemci/sunucu işlemlerini gösterir. Örnekler, bir sunucu programı tarafından işlenen bir hedef sunucu kuyruğuna istek iletileri yerleştiren istemcilerdir. Sunucu programının bir yanıtlama kuyruğuna yanıt iletileri göndermesini bekler.

İstek örnekleri, MQPUT çağrısını kullanarak hedef sunucu kuyruğuna bir dizi istek iletileri yerleştirdi. Bu iletiler yerel kuyruğu belirtir, SYSTEM.SAMPLE.REPLY , yerel ya da uzak bir kuyruk olabilen yanıt kuyruğu olarak. Programlar yanıt iletilerini bekler, sonra bunları görüntüler. Yanıtlar yalnızca, hedef sunucu kuyruğu bir sunucu uygulaması tarafından işleniyorsa ya da bu amaçla bir uygulama tetiklendiyse gönderilir (Sorgula, Ayarla, Echo örnek programları tetiklenecek şekilde tasarlanmıştır). C örneği 1 dakika bekler (COBOL örneği 5 dakika bekler), ilk yanıt için gelen ilk yanıt (bir sunucu uygulamasının tetiklenmesine izin vermek için) ve sonraki yanıtlar için 15 saniye bekler, ancak her iki örnek de yanıt almadan sona erebilir. İstek örnek programlarının adları için bkz. [“Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094](#) .

*İstek örnek programlarının çalıştırılması*

### amqsreq0.c, amqsreq ve amqsreqc örnekleri çalıştırılıyor

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 adıdır; replyqueue ise yanıt kuyruğunun adıdır.

Kuyruk yöneticisinin adını atarsanız, kuyruğun varsayılan kuyruk yöneticisinin iyesi olduğu varsayılır. Yanıt kuyruğunun adını atlarsanız, varsayılan yanıt kuyruğu sağlanır.

## amq0req0.cbl örneğinin çalıştırılması

COBOL sürümünün herhangi bir parametresi yok. Bu, varsayılan kuyruk yöneticisine bağlanır ve çalıştırdığınız zaman sizden sorulur:

```
Please enter the name of the target server queue
```

Program, girişini StdIn ögesinden alır ve her bir satırı, bir istek iletisinin içeriği olarak her metin satırını alarak hedef sunucu kuyruğuna ekler. Boş bir satır okunduğunda program sona erer.

## AMQSREQ4 örneğinin çalıştırılması

C programı, boş bir zaman sonlandırma girişi içeren stdin ' den (klavye) veri olarak iletiler oluşturur. Program üç değiştirge alır: Hedef kuyruğun adı (gerekli), kuyruk yöneticisi adı (isteğe bağlı) ve yanıtlama kuyruğu adı (isteğe bağlı). Kuyruk yöneticisi adı belirlenmezse, varsayılan kuyruk yöneticisi kullanılır. Herhangi bir yanıt kuyruğu belirlenmezse, SYSTEM.SAMPLE.REPLY kuyruğu kullanıldı.

Burada, yanıt kuyruğunu belirten, ancak kuyruk yöneticisi varsayılan değerinin belirlenmesine olanak vermek için C örnek programının nasıl çağrılacağı bir örneği yer alıyor:

```
CALL PGM(QMQM/AMQSREQ4) PARM('SYSTEM.SAMPLE.LOCAL' ' ' 'SYSTEM.SAMPLE.REPLY')
```


**Not:** Kuyruk adlarının büyük ve küçük harfe duyarlı olduğunu unutmayın. Örnek dosya yaratma programı AMQSAMP4 tarafından yaratılan tüm kuyruklar, büyük harfli karakterlerde yaratılmış adlara sahiptir.

## AMQOREQ4 örneğinin çalıştırılması

COBOL programı, klavyeden verileri kabul ederek iletiler oluşturur. Programı başlatmak için, programı çağırın ve parametre olarak hedef kuyruğunuzun adını belirtin. Program klavyeden bir arabelleğe giriş kabul eder ve her metin satırı için bir istek iletisi oluşturur. Klavye üzerine boş bir satır girdiğinizde program durur.

*Tetikleme kullanılarak İstek örneği çalıştırılıyor*

Örnek, tetikleme, Sorgu, Ayar ya da Yankı örnek programlarından biri ile kullanılırsa, giriş satırı, tetiklenen programın erişmesini istediğiniz kuyruğun kuyruk adı olmalıdır.

 *Running the Request sample using triggering on UNIX, Linux, and Windows*

UNIX, Linux, and Windows' ta, bir oturumda RUNMQTRM tetikleme izleme programını başlatın ve ardından amqsreq programını başka bir oturumda başlatın.

Örnekleri tetiklemeyi kullanarak çalıştırmak için:

1. Bir oturumda RUNMQTRM tetikleme izleme programını başlatır (başlatma kuyruğu SYSTEM.SAMPLE.TRIGGER kullanılabilir.)
2. amqsreq programını başka bir oturumda başlatın.
3. Hedef sunucu kuyruğu tanımladığınızdan emin olun.

İletileri yerleştirmek için istek örneği için hedef sunucu kuyruğu olarak kullanmak üzere kullanabileceğiniz örnek kuyruklar şunlardır:

- SYSTEM.SAMPLE.INQ -Sorgula ile ilgili örnek program için
- SYSTEM.SAMPLE.SET -Küme örnek programı için
- SYSTEM.SAMPLE.ECHO -Yankı örnek programı için

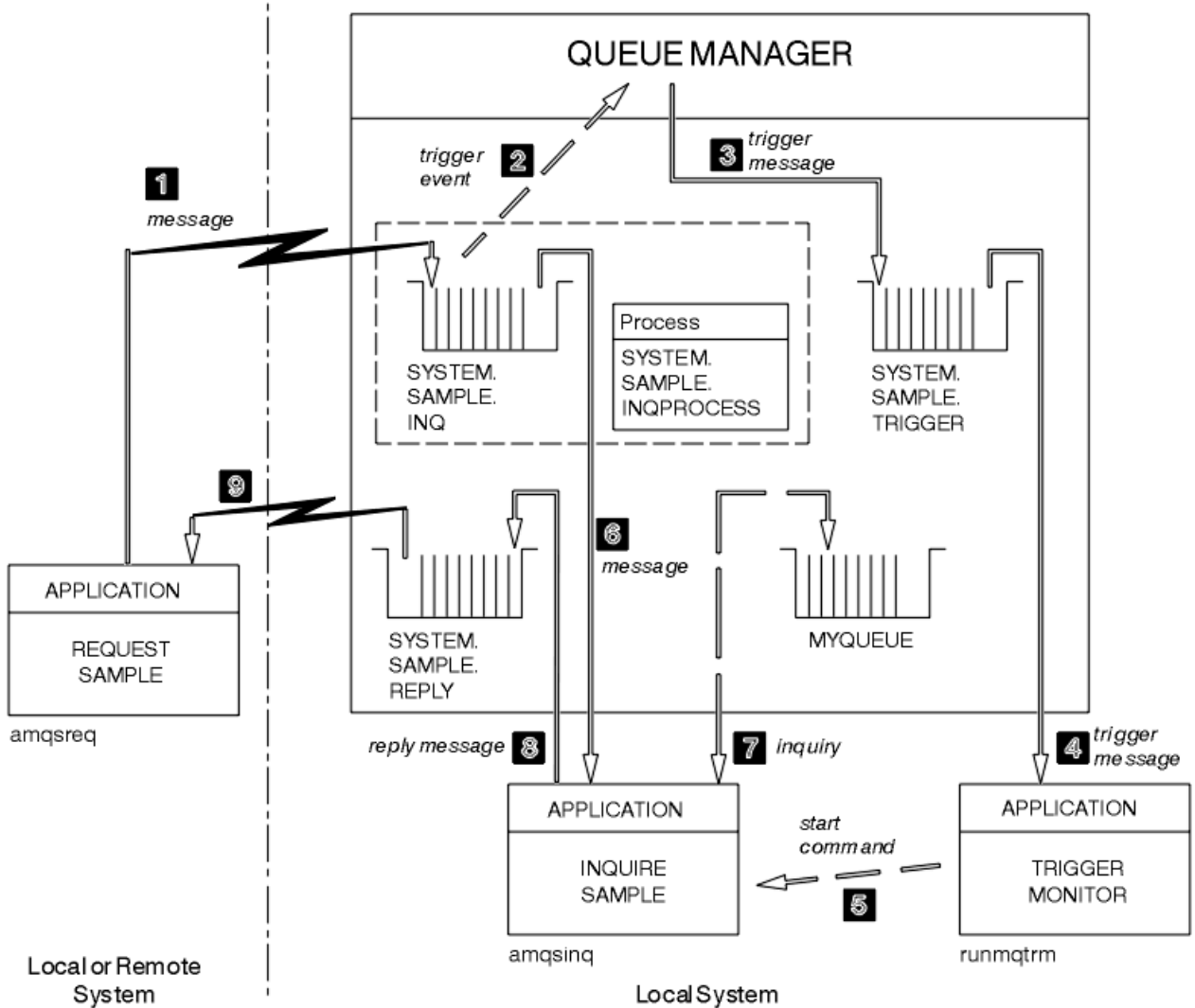
Bu kuyruklar FIRST tetikleyicisi tipine sahiptir; bu nedenle, İstek örneğini çalıştırmadan önce kuyruklarda önceden iletiler varsa, sunucu uygulamaları gönderdiğiniz iletiler tarafından tetiklenmez.

4. Kullanmak için, Sorgula, Set ya da Echo örnek programı için bir kuyruk tanımladığınızdan emin olun.

Bu, istek örneği bir ileti gönderdiğinde tetikleme izleyicinin 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 dosya ile RUNMQSC komutunu kullanın.

Şekil 137 sayfa 1147 , İsteme ve Sorgunun örneklerini birlikte nasıl kullanacağını gösterir.



Şekil 137. Tetikleme kullanarak istek ve sorgulama örnekleri

Şekil 137 sayfa 1147 ' da İstek örneği, iletileri hedef sunucu kuyruğuna ( SYSTEM.SAMPLE.INQve sorgulamak için kuyruğun, MYQUEUEUE. Alternatively, you can use one of the sample queues defined when you ran amqscos0.tst, or any other queue that you have defined, for the Inquire sample.

**Not:** Şekil 137 sayfa 1147 içindeki sayılar olayların sırasını gösterir.

İsteği ve Sorgula sorgulama örneklerini çalıştırmak için şu tetiklemeyi kullanın:

1. Kullanmak istediğiniz kuyrukların tanımlandığından emin olun. Örnek kuyrukları tanımlamak için amqscos0.tstkomutunu çalıştırın ve bir kuyruk MYQUEUEUE ögesini tanımlayın.
2. RUNMQTRM tetikleme izleyicisini çalıştırın:

```
RUNMQTRM -m qmanage:name -q SYSTEM.SAMPLE.TRIGGER
```

### 3. İstek örneğini çalıştır

```
amqsreq SYSTEM.SAMPLE.INQ
```

**Not:** Tetiklenecek olanları süreç nesnesi tanımlar. İstemci ve sunucu aynı altyapıda çalışmıyorsa, tetikleme izleyicisinin başlattığı tüm işlemler *ApplType* tanımlamalıdır; tersi durumda, sunucu varsayılan tanımlarını (yani, normalde sunucu makinesiyle ilişkili uygulama tipi) alır ve bir hataya neden olur.

Uygulama tiplerinin listesi için bkz. [ApplType](#).

### 4. Sorgulamaya ilişkin örneğinin kullanmasını istediğiniz kuyruğun adını girin:

```
MYQUEUE
```

### 5. Boş bir satır girin (İstek programını sona erdirmek için).

### 6. Bundan sonra, istek örneği, MYQUEUE ' dan elde edilen sorgulamak programı verilerini içeren bir ileti görüntüler.

Birden çok kuyruk kullanabilirsiniz; bu durumda, “4” sayfa 1148 adımındaki diğer kuyrukların adlarını girin.

Tetikleme ile ilgili daha fazla bilgi için bkz. [“Starting IBM MQ applications using triggers” sayfa 899.](#)

**IBM i**

*Running the Request sample using triggering on IBM i*

On IBM i, start the sample trigger server, AMQSERV4, in one job, then start AMQSREQ4 in another. Bu, İstek örnek programı bir ileti gönderdiğinde tetikleme sunucusunun hazır olduğu anlamına gelir.

### Not:

1. AMQSAMP4 tarafından yaratılan örnek tanımlamalar, örneklerin C sürümlerini tetikler. COBOL sürümlerini tetiklemek istiyorsanız, SYSTEM.SAMPLE.ECHOPROCESS, SYSTEM.SAMPLE.INQPROCESS ve SYSTEM.SAMPLE.SETPROCESS işlem tanımlarını değiştirin. CHGMQMPCRC komutunu kullanabilirsiniz (ayrıntılar için bkz. [Change MQ Process \(CHGMQMPCRC\)](#)) bunu yapmak ya da kendi sürümünüzü düzenlemek ve çalıştırmak için AMQSAMP4' u çalıştırın.
2. AMQSERV4 için kaynak kodu, yalnızca C dili için sağlanır. Ancak, derlenmiş bir sürüm (COBOL örnekleriyle birlikte kullanabileceğiniz) QMQM kitaplığında sağlanır.

Bu örnek sunucu kuyruklarına istek iletilerinizi koyabilirsiniz:

- SYSTEM.SAMPLE.ECHO (Yankı örnek programları için)
- SYSTEM.SAMPLE.INQ (Sorgula ilgili örnek programlar için)
- SYSTEM.SAMPLE.SET (örnek programları ayarlamak için)

SYSTEM.SAMPLE.ECHO programı Şekil 138 sayfa 1150 içinde gösterilir. Örnek verileri kullanarak, bu sunucuya C programı isteğini yayınlamaya ilişkin komut şu komutu verir:

```
CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(ECHO)')
```

**Not:** Bu örnek kuyruğun bir FIRST tetikleyicisi tipi vardır; bu nedenle, İstek örneğini çalıştırmadan önce kuyruğunda önceden iletiler varsa, sunucu uygulamaları gönderdiğiniz iletiler tarafından tetiklenmez.

Daha fazla örnek vermek istiyorsanız, aşağıdaki çeşitlendirmeleri deneyebilirsiniz:

- Ayrıntılar için AMQSTRG4 (ya da ayrıntılar için STRMQMTRM komut satırını kullanın), bkz. [Start MQ Trigger Monitor \(STRMQMTRM\)](#) Bunun yerine işi sunmak için AMQSERV4 yerine, olası iş gönderimi gecikmeleri, olan biteni takip etmeyi daha az kolaylaştırıyor olabilir.
- SYSTEM.SAMPLE.INQUIRE ve SYSTEM.SAMPLE.SET örnek programları. Örnek veri dosyası kullanılarak, bu sunuculara ilişkin C programı isteklerini yayınlamaya ilişkin komutlar şunlardır:

```
CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(INQ)')
CALL PGM(QMQMSAMP/AMQSREQ4) PARM('QMQMSAMP/AMQSDATA(SET)')
```

Bu örnek kuyruklarda, FIRST tetikleme tipi de vardır.

#### *İstek örnek programının tasarımı*

Program, iletileri koyabilmesi için hedef sunucu kuyruğunu açar. MQOO\_OUTPUT seçeneğiyle MQOPEN çağrısını kullanır. Kuyruğu açamazsa, program, MQOPED çağrısının döndürdüğü neden kodunu içeren bir hata iletisi görüntüler.

Daha sonra program, SYSTEM.SAMPLE.REPLY , böylece yanıt iletileri alabilirler. Bunun için, program MQOO\_INPUT\_EXCLUSIVE seçeneği ile MQOPEN çağrısını kullanır. Kuyruğu açamazsa, program, MQOPED çağrısının döndürdüğü neden kodunu içeren bir hata iletisi görüntüler.

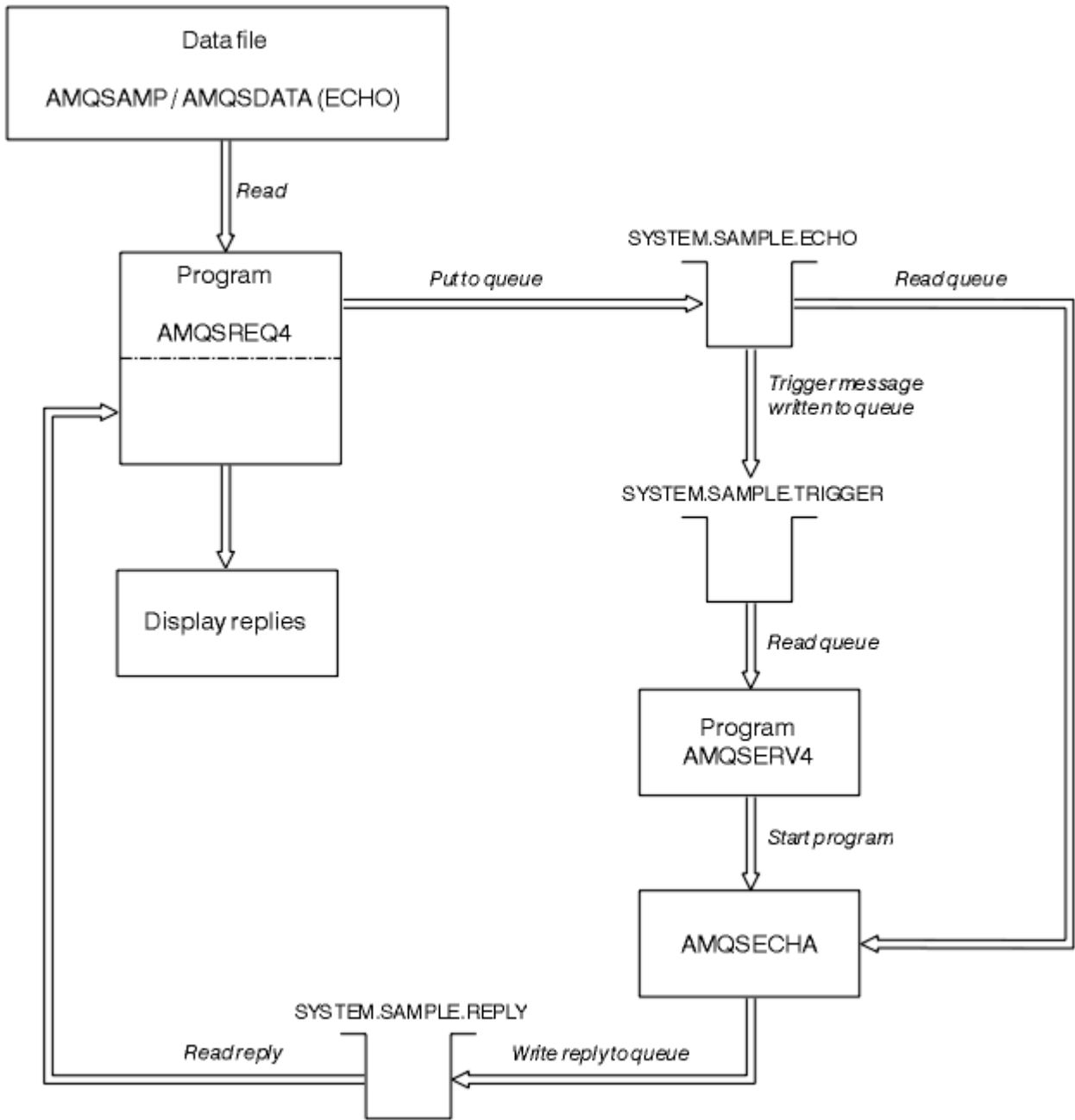
Her giriş satırı için, program metni bir arabelleğe okur ve MQPUT çağrısını kullanarak, o satırın metnini içeren bir istek iletisi yaratır. Bu çağrıda program, istek iletisine ilişkin gönderilen herhangi bir rapor iletisinin ileti verilerinin ilk 100 baytı içermesini istemek için MQRO\_EXCEPTION\_WITH\_DATA rapor seçeneğini kullanır. Program, girişin sonuna ulaşıncaya kadar ya da MQPUT çağrısının başarısız olduğu zamana kadar devam eder.

Daha sonra, program yanıtlama iletilerini kuyruktan kaldırmak için MQGET çağrısını kullanır ve yanıtlarda yer alan verileri görüntüler. MQGET çağrısı MQGMO\_WAIT, MQGMO\_CONVERT ve MQGMO\_ACCEPT\_KISALTILMIŞ seçenekleri kullanır. *WaitInterval* , COBOL sürümünde 5 dakika ve C sürümünde 1 dakika, ilk yanıt için (bir sunucu uygulamasının tetiklenmesine izin vermek için) ve sonraki yanıtlar için 15 saniye. Bu süre, kuyruğun üzerinde herhangi bir ileti yoksa, bu dönemleri bekler. Bu aralığın süresi dolmadan bir ileti gelmezse, çağrı başarısız olur ve MQRC\_NO\_MSG\_AVAILABLE neden kodunu döndürür. Çağrı ayrıca, bildirilen arabellek büyüklüğünden daha uzun iletilerin kısaltılması için MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneğini de kullanır.

Bu program, bu alanları, aldığı iletide yer alan değerlere ayarlandığından, her MQGET çağrısından sonra MQMD yapısının *MsgId* ve *CorrelId* alanlarının nasıl temizleneceğini gösterir. Bu alanların temizlenmesi, art arda gelen MQGET çağrılarının iletilerin kuyrukte tutulmakta olduğu sırayla alma çağrıları anlamına gelir.

Bu program, MQGET çağrısına ilişkin MQRC\_NO\_MSG\_AVAILABLE neden kodunu döndürünceye ya da MQGET çağrısının başarısız olduğu zamana kadar devam eder. Arama başarısız olursa, program neden kodunu içeren bir hata iletisi görüntüler.

Daha sonra, program, MQCLOSE çağrısını kullanarak hem hedef sunucu kuyruğunu, hem de yanıtlamayı kuyruğa kapatır.



Şekil 138. Örnek IBM i Client/Server (Echo) programı akış şeması

### Örnek programları ayarla

Küme örnek programları, kuyruğun **InhibitPut** özneliğini değiştirmek için MQSET çağrısını kullanarak işlemleri bir kuyruğa yerleştirmeyi engeller. Ayrıca, Set Sample programlarının tasarımıyla ilgili bilgi edinin.

Bu programların adları için bkz. “Örnek Programlar On Multiplatforms içinde gösterilen özellikler” sayfa 1094 .

Programların tetiklenen programlar olarak çalıştırılması amaçlanır, bu nedenle bunların tek girişi, sorgulanacak özneliklere sahip bir hedef kuyruğun adını içeren bir MQTMC2 (tetikleme iletisi) yapısıdır. C sürümü kuyruk yöneticisi adını da kullanır. COBOL sürümü, varsayılan kuyruk yöneticisini kullanır.

Tetikleme işleminin çalışması için, kullanmak istediğiniz Set Sample programının, SYSTEM.SAMPLE.SET kuyruğuna gelen ileteler tarafından tetiklendiğinden emin olun. To do this, specify the name of the Set sample program that you want to use in the *ApplicId* field of the process definition SYSTEM.SAMPLE.SETPROCESS. Örnek kuyruğun bir FIRST tetikleyicisi tipi vardır; İstek örneğini

çalıştırmadan önce kuyruğunda önceden iletiler varsa, Küme örneği gönderdiğiniz iletiler tarafından tetiklenmez.

Tanımlamayı doğru bir şekilde ayarladığınızda:

- **ULW** UNIX, Linux, and Windows sistemleri için, bir oturumda **runmqtrm** programını başlatın ve **amqsreq** programını başka bir oturumda başlatın.
- **IBM i** IBM için, bir oturumda AMQSERV4 programını başlatın ve ardından başka bir oturum içinde AMQSREQ4 programını başlatın. AMQSERV4 yerine AMQSTRG4 ' yi kullanabilirsiniz, ancak olası iş gönderimi gecikmeleri, gerçekleşenleri takip etmeyi daha az kolaylaştırabilirdi.

İstek örnek programlarını kullanarak, her biri yalnızca bir kuyruk adı içeren istek iletilerini SYSTEM.SAMPLE.SET kuyruğuna yollamak için kullanın. Her istek ileti için, Set Sample Programs (Örnek Programları Ayarla), belirtilen kuyruk üzerinde işlem engellenmiş olan bir doğrulama içeren bir yanıt ileti gönderir. Yanıtlar, istek iletilerinde belirtilen yanıtlama kuyruğuna gönderilir.

### Set Sample programının tasarımı

Program, tetikleme ileti yapısında adı belirtilen kuyruğu açarken, bu işlem başlatıldığında iletileceği bir kuyruğa yer alan kuyruğu açar. (For clarity, we will call this the *istek kuyruğu*.) Program, bu kuyruğu paylaşılan giriş için açmak için MQOPEN çağrısını kullanır.

Program, bu kuyruktan iletileri kaldırmak için MQGET çağrısını kullanır. Bu çağrı, 5 saniye bekleme süresi ile MQGMO\_ACCEPT\_TRUNCATED\_MSG ve MQGMO\_WADET seçeneklerini kullanır. Program, bir istek ileti olup olmadığını görmek için her iletinin tanımlayıcısını sınar; değilse, program iletiyi atar ve bir uyarı ileti görüntüler.

İstek kuyruğundan kaldırılan her istek ileti için, program kuyruğun adını okur (*hedef kuyruk* adını arayacağız). verilerde bulunur ve MQOO\_SET seçeneğiyle MQOPER çağrısını kullanarak kuyruğu açar. Program daha sonra, hedef kuyruğun **InhibitPut** özneliğinin değerini MQQA\_PUT\_INHIBITED olarak ayarlamak için MQSET çağrısını kullanır.

MQSET çağrısı başarılı olursa, program yanıt kuyruğuna yanıt ileti koymak için MQPUT1 çağrısını kullanır. Bu ileti, PUT inhibited dizisini içerir.

MQAUT ya da MQSET çağrısı başarısız olursa, program yanıt kuyruğuna report ileti koymak için MQPUT1 çağrısını kullanır. Bu rapor iletilerinin ileti tanımlayıcısının *Feedback* alanında, başarısız olan buna bağlı olarak, MQOPEN ya da MQSET çağrısının döndürdüğü neden kodudur.

MQSET çağrısından sonra program, MQCLOSE çağrısını kullanarak hedef kuyruğu kapatır.

İstek kuyruğunda bir ileti kalmadığında, program o kuyruğu kapatır ve kuyruk yöneticisinden bağlantıyı keser.

### TLS örnek programı

AMQSSLC, MQCONNX çağrısında TLS istemci bağlantısı bilgilerini sağlamak için MQCNO ve MQSCO yapılarının nasıl kullanılacağını gösteren örnek bir C programıdır. Bu, istemci MQI uygulamasının, istemci bağlantı kanalı ve TLS ayarlarını bir istemci kanal tanımlama çizelgesi (CCDT) olmadan yürütme sırasında sağlamasına olanak sağlar.

Bir bağlantı adı belirtilirse, program bir MQCD yapısında istemci bağlantısı kanalı tanımlaması oluşturur.

Anahtar havuzu dosyasının kök adı belirtilirse, program bir MQSCO yapısı oluşturur; OCSP yanıtlayıcı URL 'si de sağlanırsa, program bir kimlik doğrulama bilgisi kaydı MQAIR yapısını oluşturur.

Daha sonra, program MQCONNX komutunu kullanarak kuyruk yöneticisine bağlanır. Bağlantı kuruldu ve bağlı olduğu kuyruk yöneticisinin adını yazdırır.

Bu programın bir MQI istemci uygulaması olarak bağlanması amaçlanır. Ancak, olağan bir MQI uygulaması olarak bağlantılandırılabilir. Daha sonra, yerel bir kuyruk yöneticisine bağlanır ve istemci bağlantısı bilgilerini yoksayar.

AMQSSLC aşağıdaki değişirgeleri kabul eder; bunların tümü isteğe bağlıdır:

**-m QmgrName**

Bağlanılacak kuyruk yöneticisinin adı

**-c ChannelName**

Kullanılacak kanalın adı

**-x ConnName**

Sunucu bağlantısı adı

TLS parametreleri:

**-k KeyReposStem**

Anahtar havuzu dosyasının kök adı. Bu, .kdb soneki olmadan dosyanın tam yoludur. Örneğin:

```
/home/user/client  
C:\User\client
```

**-s CipherSpec**

Kuyruk yöneticindeki SVRCONN kanal tanımlamasındaki SSLCIPH ' ye karşılık gelen TLS kanalı CipherSpec dizgisi.

**-f**

Yalnızca FIPS 140-2 sertifikalı algoritmaların kullanılması gerektiğini belirtir.

**-b VALUE1[,VALUE2...]**

Yalnızca Suite B uyumlu algoritmaların kullanılması gerektiğini belirtir. Bu değıştirge, řu değerlerden birinin ya da birkaçının virgülle ayrılmış bir listesidir: NONE,128\_BIT,192\_BIT. Bu değerler, MQSUITEB ortam değışkeni ile aynı anlamı ve istemci yapılandırma dosyasındaki eşdeğer EncryptionPolicySuiteB ayarına sahip olur.

**-p İlkesi**

Kullanılacak sertifika doğrulama ilkesini belirtir. Bu, aşağıdaki değerlerden biri olabilir:

**HERHANGİ BİRİ**

Güvenli yuva kitaplığı tarafından desteklenen sertifika geçerlilik denetimi ilkelerinin her birini uygulayın ve herhangi bir ilkenin sertifika zincirini geçerli olarak kabul etmesi durumunda sertifika zincirini kabul edin. Bu ayar, modern sertifika standartlarıyla uyumlu olmayan eski dijital sertifikalar ile geriye dönük uyumluluk üst sınırı için kullanılabilir.

**RFC5280**

Yalnızca RFC 5280 uyumlu sertifika geçerlilik denetimi ilkesini uygulayın. Bu ayar ANY ayarından daha katı geçerlilik denetimi sağlar, ancak bazı eski dijital sertifikaları reddeder.

Varsayılan değer ANY değeridir.

**-l CertLabel**

Güvenli bağlantı için kullanılacak sertifika etiketi.

**Not:** Değeri küçük harfli karakterler kullanarak belirtmeniz gerekir.

OCSP sertifikası iptal parametresi:

**-o URL**

OCSP Yanıtlayıcı URL 'si

*TLS örnek programının çalıştırılması*

TLS örnek programını çalıştırmak için önce TLS ortamınızı ayarlamanız gerekir. Daha sonra, bir dizi parametre belirterek, örneği komut satırından çalıştırıyorsunuz.

**Bu görev hakkında**

Aşağıdaki yönergeler, örnek programı kişisel sertifikalar kullanarak çalıştırır. Komut, örneğin, CA sertifikalarını kullanabilir ve bir OCSP yanıtlayıcısı kullanarak durumlarının denetlenmesini sağlar. Örnek içindeki yönergelere bakın.



## Yordam

1. QM1adını taşıyan bir kuyruk yöneticisi yaratın. Daha fazla bilgi için bkz. [crtmqm](#).
2. Kuyruk yöneticisi için bir anahtar havuzu yaratın. Daha fazla bilgi için bakınız: [Setting up a key Repository on UNIX, Linux, and Windows](#).
3. İstemci için bir anahtar havuzu yaratın. Call it *clientkey.kdb*.
4. Kuyruk yöneticisi için kişisel bir sertifika yaratın. Daha fazla bilgi için bakınız: [Creating a self-signad personal certificate on UNIX, Linux, and Windows](#).
5. İstemci için kişisel bir sertifika yaratın.
6. Sunucu anahtar havuzundan kişisel sertifikayı alın ve istemci havuzuna ekleyin. Daha fazla bilgi için bkz. [Extracting the public part of a self-signed certificate from a key repository on UNIX, Linux, and Windows](#)ve [UNIX, Linux ya da Windows sistemlerinde CA sertifikası \(ya da kendinden onaylı bir sertifikenin genel bölümü\) bir anahtar havuzuna eklenmesi](#).
7. İstemci anahtar havuzundan kişisel sertifikayı alın ve sunucu anahtar havuzuna ekleyin.
8. MQSC komutunu kullanarak bir sunucu bağlantı kanalı yaratın:

```
DEFINE CHANNEL(QM1SVRCONN) CHLTYPE(SVRCONN) TRPTYPE(TCP)
SSLCPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
```

Ek bilgi için bkz. [Sunucu bağlantı kanalı](#)

9. Kuyruk yöneticinde bir kanal dinleyicisi tanımlayın ve başlatın. Ek bilgi için [DEFINE LISTENER](#) ve [START LISTENER](#)başlıklı konuya bakın.
10. Aşağıdaki komutu kullanarak örnek programı çalıştırın:

```
AMQSSSLC -m QM1 -c QM1SVRCONN -x localhost
-k "C:\Program Files\IBM\MQ\clientkey" -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 sorgular.
3. Kuyruk yöneticisini kapatır.
4. Kuyruk yöneticisinden bağlantıyı keser.

Örnek program başarıyla çalıştırılırsa, çıkışı aşağıdaki örneğe benzer şekilde 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 URL 'si belirtirseniz, aşağıdaki iletiyi alırsınız:

```
MQCONN ended with reason code 2553
```

Neden kodlarının bir listesi için bkz. [API tamamlama ve neden kodları](#).

## Tetikleme örnek programları

Tetikleme örneğinde sağlanan işlev, **runmqtrm** programındaki tetikleyici izleyicide sağlanan bir altkümedir.

Bu programların adları için bkz. "[Örnek Programlar On Multiplatforms içinde gösterilen özellikler](#)" sayfa 1094 .

## Tetikleme örneğinin tasarımı

Tetikleme örneği programı, MQOO\_INPUT\_AS\_Q\_DEF seçeneğiyle MQOPEN çağrısını kullanarak başlatma kuyruğunu açar. Sınırsız bekleme aralığı belirterek, MQGET çağrısını MQGMO\_accept\_truncated\_msg ve MQGMO\_WAWT seçenekleriyle kullanarak, başlatma kuyruğundan iletiler alır. Program, iletileri sırayla almak için her MQGET çağrısından önce *MsgId* ve *CorrelId* alanlarını temizler.

Başlatma kuyruğundan bir ileti alındığında, program iletinin büyüklüğünü denetleyerek, bu iletinin bir MQTM yapısıyla aynı boyutta olduğundan emin olmak için iletiyi sınar. Bu sınıma başarısız olursa, program bir uyarı görüntüler.

Geçerli tetikleyici iletiler için, tetikleme örneği şu alanlardaki verileri kopyalar: *ApplicId*, *EnvrData*, *Version*, ve *ApplType*. Bu alanların son ikisi sayısaldir; bu nedenle program, IBM i, UNIX, Linux, and Windows sistemleri için MQTMC2 yapısında kullanılacak karakter değiştirmeleri yaratır.

Tetikleme örneği, tetikleme iletinin *ApplicId* alanında belirtilen uygulamaya bir başlangıç komutu verir ve bir MQTMC2 ya da MQTMC (tetikleme iletinin karakter sürümü) yapısını geçirir.

- ▶ **ULW** In UNIX, Linux, and Windows systems, the *EnvrData* field is used as an extension to the invoking command string.
- ▶ **IBM i** IBM i' ta, iş önceliği ya da iş tanımlaması gibi, iş gönderme değiştirmeleri olarak kullanılır.

Son olarak, program başlatma kuyruğunu kapatır.

## Ending the triggering sample programs on IBM i

### ▶ IBM i

Bir tetikleme izleme programı, sysrequest seçeneği 2 (ENDRQS) tarafından ya da tetikleyici kuyruğundan alıkonabilecek bir program tarafından sona erdirilebilir.

Örnek tetikleyici kuyruğu kullanılırsa, komut şöyle olur:

```
CHGMQM QNAME('SYSTEM.SAMPLE.TRIGGER') QMQNAME GETENBL(*NO)
```

**Önemli:** Bu kuyrukda yeniden tetiklemeden önce, aşağıdaki komutu girmeniz gerekir:

```
CHGMQM QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*YES)
```

### Tetikleme örnek programlarının çalıştırılması

Bu konu, tetikleme örneği programlarının çalıştırılmasına ilişkin bilgiler içerir.

## amqstrg0.c, amqstrg ve amqstrgc örneklerinin çalıştırılması

Program 2 parametre alır:

1. Başlatma kuyruğunun adı (gerekli)
2. Kuyruk yöneticisinin adı (isteğe bağlı)

Kuyruk yöneticisi belirtilmediyse, varsayılan değer olarak varsayılan değer olarak bağlanır. A sample initiation queue will have been defined when you ran amqscos0.tst; the name of that queue is SYSTEM.SAMPLE.TRIGGER, and you can use it when you run this program.

**Not:** Bu örnekteki işlev, runmqtrm programında sağlanan tam tetikleme işlevinin bir alt kümesidir.

## AMQSTRG4 örneğinin çalıştırılması

### IBM i

Bu, IBM i ortamı için bir tetikleyici izleyicidir. Başlatılacak her uygulama için tek bir IBM i işi teslim edilir. Bu, her bir tetikleyici iletilisiyle ilişkilendirilmiş ek bir işlemenin olduğu anlamına gelir.

AMQSTRG4 (QCSRC içinde) iki değiştirge alır: Hizmet vermek üzere olduğu başlangıç kuyruğunun adı ve kuyruk yöneticisinin adı (isteğe bağlı). AMQSAMP4 (QCLSRC içinde), bir örnek başlatma kuyruğu (SYSTEM.SAMPLE.TRIGGER, örnek programları denediğinizde kullanabileceğiniz bir tetikleyiciye sahip olabilir.

Tetikleme kuyruğunu örnek olarak kullanarak, şu komutu verin:

```
CALL PGM(QMQM/AMQSTRG4) PARM('SYSTEM.SAMPLE.TRIGGER')
```

Diğer bir seçenek olarak, ayrıntılar için, CL eşdeğeri STRMQMTRM 'yi (STRMQMTRM) kullanabilirsiniz. Bkz. [Start MQ Trigger Monitor \(STRMQMTRM\)](#).

## AMQSERV4 örneğinin çalıştırılması

### IBM i

Bu, IBM i ortamı için bir tetikleme sunucusudur. Her bir tetikleme iletilisi için, bu sunucu belirlenen uygulamayı başlatmak için kendi işinde başlatma komutunu çalıştırır. Tetikleyici sunucusu CICS işlemlerini çağırabilir.

AMQSERV4 iki değiştirge alır: hizmet vermek üzere olduğu başlatıcı kuyruğunun adı ve kuyruk yöneticisinin adı (isteğe bağlı). AMQSAMP4 , bir örnek başlatma kuyruğu (SYSTEM.SAMPLE.TRIGGER, örnek programları denediğinizde kullanabileceğiniz bir tetikleyiciye sahip olabilir.

Örnek tetikleme kuyruğunun kullanılması, komutun yayınına aşağıdaki komutu verir:

```
CALL PGM(QMQM/AMQSERV4) PARM('SYSTEM.SAMPLE.TRIGGER')
```

### *Tetikleme sunucusunun tasarımı*

Tetikleme sunucusunun tasarımı, tetikleme izleyicisine benzer birkaç kural dışı durumla aynı

Tetikleme sunucusunun tasarımı tetikleme izleyicisine benzer; tetikleme sunucusu dışında:

- MQAT\_CICS uygulamalarının yanı sıra MQAT\_OS400 uygulamalarını da sağlar.
- **IBM i** Bir IBM i işi göndermek yerine IBM i uygulamalarını kendi işinde çağırır (ya da CICS uygulamalarını başlatmak için STRCICSUSAR 'ı kullanır).
- For CICS applications, substitutes the *EnvData*, for example, to specify the CICS region, from the trigger message in the STRCICSUSR command.
- Paylaşılan girişe ilişkin başlatma kuyruğunu açar; böylece, birden çok tetikleyici sunucu aynı anda çalışabilir.

**Not:** AMQSERV4 tarafından başlatılan programlar, tetikleme sunucusunu durdurduğu için MQDISC çağrısını kullanmamalıdır. AMQSERV4 tarafından başlatılan programlar MQCONN çağrısını kullanarak başlatıldıysa, MQRC\_ALREADY\_CONNECTED neden kodunu alır.

Smokin için Put ve Get örnek programları hakkında bilgi edinin ve sunucu ortamını SMOKIN ' de oluşturma.

## Başlamadan önce

Bu örnekleri çalıştırmadan önce, sunucu ortamını oluşturmanız gerekir.

## Bu görev hakkında

**Not:** Bu kısım boyunca ters eğik çizgi (\) karakteri, uzun komutları birden çok satıra bölmek için kullanılır. Bu karakteri girmeyin. Her komutu tek bir satır olarak girin.

Farklı altyapılar için IBM MQ için sunucu ortamının oluşturulmasına ilişkin bilgiler.

## Başlamadan önce

Çalışan bir TUXEDO ortamınız olduğu varsayılır.

IBM MQ for AIX (32 bit) için sunucu ortamı nasıl oluşturulacağı.

## Yordam

1. Sunucu ortamının oluşturulduğu bir dizin (örneğin, APPDIR) oluşturun ve bu dizindeki tüm komutları yürütür.
2. Aşağıdaki ortam değişkenlerini dışa aktarın; burada TUXDIR , SMOKIN için kök dizin ve MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder:

```
$ 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. Add the following line to the TUXEDO file udataobj/RM:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqmx -lmqm
```

4. Şu 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/amqstxvx.c \
-f MQ_INSTALLATION_PATH/lib/libmqm.a \
-r MQSeries_XA_RMI -s MPUT1:MPUT \
-s MGET1:MGET \
-v -bsh
$ buildserver -o MQSERV2 -f MQ_INSTALLATION_PATH/samp/amqstxvx.c \
-f MQ_INSTALLATION_PATH/lib/libmqm.a \
-r MQSeries_XA_RMI -s MPUT2:MPUT \
-s MGET2:MGET \
-v -bsh
$ 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 gereken şekilde makine adının, çalışma dizinlerinin ve kuyruk yöneticisine ilişkin ayrıntıları ekleyin:

```
$ tmloadcf -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

#### 6. TLOGDEVICE ' yi yaratın:

```
$tmadmin -c
```

Bir bilgi istemi görüntülenir. Bu komut isteminde şunu girin:

```
> crd1 -z /APPDIR/TLOG1
```

#### 7. Kuyruk yöneticisini başlat:

```
$ stmqm
```

#### 8. Smokin Başlat:

```
$ tmboot -y
```

## Sonraki adım

Şimdi, iletileri bir kuyruğa koymak ve kuyruktan almak için, doputs ve dogets programlarını kullanabilirsiniz.

**AIX** *Building the server environment for AIX (64-bit)*  
IBM MQ for AIX (64 bit) için sunucu ortamı nasıl oluşturulacağı.

## Yordam

1. Sunucu ortamının oluşturulduğu bir dizin (örneğin, APPDIR) oluşturun ve bu dizindeki tüm komutları yürütür.
2. Aşağıdaki ortam değişkenlerini dışa aktarın; burada TUXDIR , SMOKIN için kök dizini temsil eder ve MQ\_INSTALLATION\_PATH , IBM MQ ' in 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. Add the following line to the TUXEDO file udataobj/RM:

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: -lmqma64 -lmqm
```

4. Şu 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/amqstxvx.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/amqstxvx.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 gereken şekilde makine adının, çalışma dizinlerinin ve kuyruk yöneticisine ilişkin ayrıntıları ekleyin:

```
$ tmloadcf -y MQ_INSTALLATION_PATH/samp/ubbstxcx.cfg
```

6. TLOGDEVICE ' yi yaratın:

```
$tmadmin -c
```

Bir bilgi istemi görüntülenir. Bu komut isteminde şunu girin:

```
> crdl -z /APPDIR/TLOG1
```

7. Kuyruk yöneticisini başlat:

```
$ stmqm
```

8. Smokin Başlat:

```
$ tmboot -y
```

## Sonraki adım

Şimdi, iletileri bir kuyruğa koymak ve kuyruktan almak için, doputs ve dogets programlarını kullanabilirsiniz.

**Solaris** *Building the server environment for Solaris (32-bit)*  
IBM MQ for Solaris (32 bit) için sunucu ortamı nasıl oluşturulacağı.

## Bu görev hakkında

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

## Yordam

1. Sunucu ortamının oluşturulduğu bir dizin (örneğin, APPDIR) oluşturun ve bu dizindeki tüm komutları yürütür.
2. Aşağıdaki ortam değişkenlerini dışa aktarın; burada TUXDIR , TUXEDO için kök dizindir:

```
$ export CFLAGS="-I /APPDIR"  
$ export FIELDTBLS=amqstxvx.flds  
$ export VIEWFILES=amqstxvx.V  
$ export SHLIB_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib  
$ export LD_LIBRARY_PATH=$TUXDIR/lib:MQ_INSTALLATION_PATH/lib:/lib
```

3. Aşağıdakileri SMOKIN kütüğüne ( udataobj/RM ) ekleyin (RM, MQ\_INSTALLATION\_PATH/lib/libmqmcs ve MQ\_INSTALLATION\_PATH/lib/libmqmzse) içermelidir.

```
MQSeries_XA_RMI:MQRMIXASwitchDynamic: \  
MQ_INSTALLATION_PATH/lib/libmqmxa.a MQ_INSTALLATION_PATH/lib/libmqm.so \  
/opt/tuxedo/lib/libtux.a MQ_INSTALLATION_PATH/lib/libmqmcs.so \  
MQ_INSTALLATION_PATH/lib/libmqmzse.so
```

4. Şu komutları çalıştırın:

```
$ mkfldhdr amqstxvx.flds
```

```

$ viewc      amqstvx.v
$ buildtms   -o MQXA -r MQSERIES_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
  -f MQ_INSTALLATION_PATH/lib/libmqm.so \
  -r MQSERIES_XA_RMI -s MPUT1:MPUT \
  -s MGET1:MGET \
  -v -bshm
  -l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
  -f MQ_INSTALLATION_PATH/lib/libmqm.so \
  -r MQSERIES_XA_RMI -s MPUT2:MPUT \
  -s MGET2:MGET \
  -v -bshm
  -l -ldl
$ buildclient -o doputs -f amqstxpx.c \
  -f MQ_INSTALLATION_PATH/lib/libmqm.so \
  -f MQ_INSTALLATION_PATH/lib/libmqmzse.co \
  -f MQ_INSTALLATION_PATH/lib/libmqmcs.so
$ buildclient -o dogets -f amqstxgx.c \
  -f MQ_INSTALLATION_PATH/lib/libmqm.so
  -f MQ_INSTALLATION_PATH/lib/libmqmzse.co \
  -f MQ_INSTALLATION_PATH/lib/libmqmcs.so

```

5. ubbstxcx.cfg dosyasını düzenleyin ve gereken şekilde makine adının, çalışma dizinlerinin ve kuyruk yöneticisine ilişkin ayrıntıları ekleyin:

```
$ tmloadcf -y ubbstxcx.cfg
```

6. TLOGDEVICE ' yi yaratın:

```
$tmadmin -c
```

Bir bilgi istemi görüntülenir. Bu komut isteminde şunu girin:

```
> crdl -z /APPDIR/TLOG1
```

7. Kuyruk yöneticisini başlat:

```
$ stmqm
```

8. Smokin Başlat:

```
$ tmboot -y
```

## Sonraki adım

Şimdi, iletileri bir kuyruğa koymak ve kuyruktan almak için, doputs ve dogets programlarını kullanabilirsiniz.

**Solaris** *Building the server environment for Solaris (64-bit)*  
IBM MQ for Solaris (64 bit) için sunucu ortamı nasıl oluşturulacağı.

## Bu görev hakkında

MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduğu üst düzey dizini temsil eder.

## Yordam

1. Sunucu ortamının oluşturulduğu bir dizin (örneğin, APPDIR) oluşturun ve bu dizindeki tüm komutları yürütür.
2. Aşağıdaki ortam değişkenlerini dışa aktarın; burada TUXDIR , TUXEDO için kök dizindir:

```
$ export CFLAGS="-I /APPDIR"
```

```
$ export FIELDTBLS=amqstxvx.flds
$ export VIEWFILES=amqstxvx.V
$ export SHLIB_PATH=$TUXDIR/lib: MQ_INSTALLATION_PATH/lib:/lib64
$ export LD_LIBRARY_PATH=$TUXDIR/lib64: MQ_INSTALLATION_PATH/lib64:/lib64
```

3. Aşağıdakileri SMOKIN kütüğüne ( udataobj/RM ) ekleyin (RM, MQ\_INSTALLATION\_PATH/lib/libmqmcs ve MQ\_INSTALLATION\_PATH/lib/libmqmzse) içermelidir.

```
MQSERIES_XA_RMI:MQRMIXASwitchDynamic: \
MQ_INSTALLATION_PATH/lib64/libmqma64.a MQ_INSTALLATION_PATH/lib64/libmqm.so \
/opt/tuxedo/lib64/libtux.a MQ_INSTALLATION_PATH/lib64/libmqmcs.so \
MQ_INSTALLATION_PATH/lib64/libmqmzse.so
```

4. Şu komutları çalıştırın:

```
$ mkfldhdr amqstxvx.flds
$ viewc amqstxvx.v
$ buildtms -o MQXA -r MQSERIES_XA_RMI
$ buildserver -o MQSERV1 -f amqstxsx.c \
-f MQ_INSTALLATION_PATH/lib64/libmqm.so \
-r MQSERIES_XA_RMI -s MPUT1:MPUT \
-s MGET1:MGET \
-v -bshm
-l -ldl
$ buildserver -o MQSERV2 -f amqstxsx.c \
-f MQ_INSTALLATION_PATH/lib64/libmqm.so \
-r MQSERIES_XA_RMI -s MPUT2:MPUT \
-s MGET2:MGET \
-v -bshm
-l -ldl
$ buildclient -o doputs -f amqstxpx.c \
-f MQ_INSTALLATION_PATH/lib64/libmqm.so \
-f MQ_INSTALLATION_PATH/lib64/libmqmzse.co \
-f MQ_INSTALLATION_PATH/lib64/libmqmcs.so
$ buildclient -o dogets -f amqstxgx.c \
-f MQ_INSTALLATION_PATH/lib64/libmqm.so
-f MQ_INSTALLATION_PATH/lib64/libmqmzse.co \
-f MQ_INSTALLATION_PATH/lib64/libmqmcs.so
```

5. ubbstxcx.cfg dosyasını düzenleyin ve gereken şekilde makine adının, çalışma dizinlerinin ve kuyruk yöneticisine ilişkin ayrıntıları ekleyin:

```
$ tmloadcf -y ubbstxcx.cfg
```

6. TLOGDEVICE ' yi yaratın:

```
$tmadmin -c
```

Bir bilgi istemi görüntülenir. Bu komut isteminde şunu girin:

```
> crdl -z /APPDIR/TLOG1
```

7. Kuyruk yöneticisini başlat:

```
$ stmqm
```

8. Smokin Başlat:

```
$ tmboot -y
```

## Sonraki adım

Şimdi, iletileri bir kuyruğa koymak ve kuyruktan almak için, doputs ve dogets programlarını kullanabilirsiniz.



## Bu görev hakkında

**Not:** Aşağıdaki dizinde yer alan *DEĞERFLER* olarak tanımlanan alanları dizin yollarına çevirin:

Çizelge 171. Dizin yollarında değişiklik yapmak için alanlar	
Alan	Dizin yolu
<i>MQMDIR</i>	IBM MQ kurulduğunda belirtilen dizin yolu (örneğin, g: \Program Files\IBM\MQ).
<i>TUXDIR</i>	SMOKIN kurulurken belirtilen dizin yolu (örneğin, f: \tuxedo).
<i>APPDIR</i>	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\amqstxn.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 139. IBM MQ for Windows için ubbstxcn.cfg dosyası örneği

**Not:** Change the machine name *MachineName*, and the directory paths, to match your installation. Ayrıca, *MYQUEUEMANAGER* kuyruk yöneticisi adını bağlanmak istediğiniz kuyruk yöneticisi adına çevirin.

The sample ubbconfig file for IBM MQ for Windows is listed in [Şekil 139 sayfa 1161](#). Bu, IBM MQ örnek dizininde ubbstxcn.cfg olarak sağlanır.

IBM MQ for Windows için sağlanan örnek makefile (bkz. [Şekil 140 sayfa 1162](#)), ubbstxmn.makolarak adlandırılır ve IBM MQ örnek 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)\amqstpx.c \
-f $(MQMLIB)\mqm.lib -v $(INC) $(DBG)
$(TUXDIR)\bin\buildclient -o dogets -f $(MQMSAMP)\amqstgx.c \
-f $(MQMLIB)\mqm.lib $(INC) -v $(DBG)
$(TUXDIR)\bin\tmloadcf -y $(APPDIR)\ubbstxcn.cfg
```

Şekil 140. IBM MQ for Windows için örnek SMOKIN 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. Bu dosyaların her birini, kuruluşunuzda kullanılan dizin adlarını ve dizin yollarını ayarlamak için düzenleyin.
4. Bağlanmak istediğiniz makine adına ve kuyruk yöneticisine ilişkin ayrıntıları eklemek için ubbstxcn.cfg (bkz. [Şekil 139 sayfa 1161](#)) ögesini düzenleyin.
5. Add the following line to the TUXEDO file `TUXDIR`udataobj\rm:

```
MQSERIES_XA_RMI;MQRMIXASwitchDynamic;MQMDIR\tools\lib\mqmxa.lib MQMDIR\tools\lib\mqm.lib
```

Yeni girişin, dosyada bir satır olması gerekir.

6. Aşağıdaki ortam değişkenlerini ayarlayın:

```
TUXDIR=TUXDIR
TUXCONFIG=APPDIR\tuxconfig
FIELDTBLS=MQMDIR\tools\c\samples\amqstxvx.fld
LANG=C
```

7. SMOKIN için bir TLOG aygıtı oluşturun.

Bunu yapmak için `tmadmin -ckomutunu` çağırın ve şu komutu girin:

```
cdl -z APPDIR\TLOG
```

8. Geçerli dizini `APPDIR` olarak ayarlayın ve örnek makefile `amqstxmn.mak` 'ı dış proje makefile olarak çağırın. Örneğin, Microsoft Visual C++ ile aşağıdaki komutu verin:

```
msvc amqstxmn.mak
```

Tüm örnek programları oluşturmak için **oluştur** seçeneğini belirleyin.

**Windows** *Building the server environment for Windows (64-bit)*  
IBM MQ for Windows (64 bit) için sunucu ortamı nasıl oluşturulacağı.

## Bu görev hakkında

**Not:** Aşağıdaki dizinde yer alan *DEĞERFLER* olarak tanımlanan alanları dizin yollarına çevirin:

<i>Çizelge 172. Dizin yollarında değişiklik yapmak için alanlar</i>	
<b>Alan</b>	<b>Dizin yolu</b>
<i>MQMDIR</i>	IBM MQ kurulduğunda belirtilen dizin yolu (örneğin, <code>g:\Program Files\IBM\MQ</code> ).
<i>TUXDIR</i>	SMOKIN kurulurken belirtilen dizin yolu (örneğin, <code>f:\tuxedo</code> ).
<i>APPDIR</i>	Örnek uygulama için kullanılacak dizin yolu (örneğin, <code>f:\tuxedo\apps\mqapp</code> ).

```

*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;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 141. IBM MQ for Windows için ubbstxcn.cfg dosyası örneği

**Not:** Change the machine name *MachineName*, and the directory paths, to match your installation. Ayrıca, *MYQUEUEMANAGER* kuyruk yöneticisi adını bağlamak istediğiniz kuyruk yöneticisi adına çevirin.

The sample ubbconfig file for IBM MQ for Windows is listed in [Şekil 141 sayfa 1164](#). Bu, IBM MQ örnek dizininde ubbstxcn.cfg olarak sağlanır.

The sample makefile (see [Şekil 142 sayfa 1165](#)) supplied for IBM MQ for Windows is called ubbstxmn.mak, and is held in the IBM MQ samples directory.

```

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 142. IBM MQ for Windows için örnek SMOKIN 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. Bu dosyaların her birini, kuruluşunuzda kullanılan dizin adlarını ve dizin yollarını ayarlamak için düzenleyin.
4. Edit ubbstxcn.cfg (see [Şekil 141 sayfa 1164](#)) to add details of the machine name and the queue manager that you want to connect to.
5. Add the following line to the TUXEDO file `TUXDIR\udataobj\rm`

```
MQSERIES_XA_RMI;MQRMIXASwitchDynamic;MQMDIR\tools\lib64\mqmxa64.lib
MQMDIR\tools\lib64\mqm.lib
```

Yeni girişin, dosyada bir satır olması gerekir.

6. Aşağıdaki ortam değişkenlerini ayarlayın:

```
TUXDIR=TUXDIR
TUXCONFIG=APPDIR\tuxconfig
FIELDTBLS=MQMDIR\tools\c\samples\amqstxvx.fld
LANG=C
```

7. SMOKIN için bir TLOG aygıtı oluşturun. Bunu yapmak için `tadmin -ckomutunu` çağırın ve komutu girin:

```
crdl -z APPDIR\TLOG
```

8. Geçerli dizini *APPDIR* olarak ayarlayın ve örnek makefile `amqstxmn.mak` 'ı dış proje makefile olarak çağırın. Örneğin, Microsoft Visual C++ ile aşağıdaki komutu verin:

```
msvc amqstxmn.mak
```

Tüm örnek programları oluşturmak için **oluştur** seçeneğini belirleyin.

Windows UNIX

*SMOKIN için örnek sunucu programı*

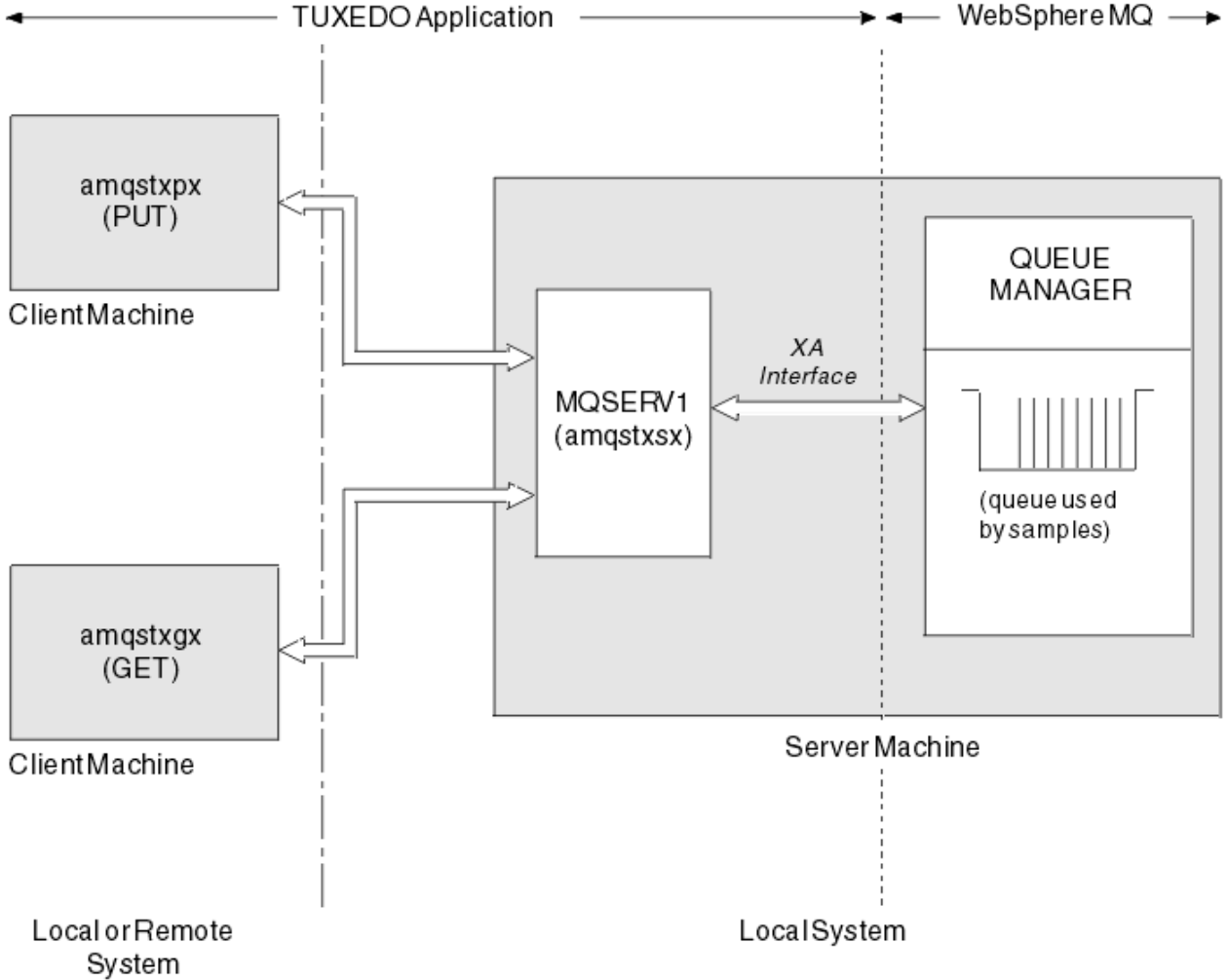
Örnek sunucu programı (`amqstxsx`), Put (`amqstxpx.c`) ve Get (`amqstxgx.c`) örnek programlarıyla çalışacak şekilde tasarlanmıştır. Örnek sunucu programı, SMOKIN başlatıldığında otomatik olarak çalıştırılır.

**Not:** SMOKIN başlatılmadan önce kuyruk yöneticinizi başlatmanız gerekir.

Örnek sunucu iki SMOKIN hizmetleri, MPUT1 ve MGET1:sağlar.

- MPUT1 hizmeti, PUT örneğiyle yönlendirilir ve SMOKIN tarafından denetlenen bir iş birimine ileti koymak için syncpoint 'te MQPUT1 kullanır. PUT numunesi tarafından sağlanan QName ve Message Text parametrelerinin yer aldığı parametrelere neden olur.
- MGET1 hizmeti açılır ve her ileti için kuyruğu her zaman kapatır. Get Sample tarafından sağlanan QName ve Message Text parametrelerinin yer aldığı parametrelere neden olur.

Herhangi bir hata iletisi, neden kodları ve durum iletileri SMOKIN günlük dosyasına yazılır.



Şekil 143. Smokin numunesi nasıl birlikte çalışıyor?

Windows

UNIX

### SMOKIN için örnek program koy

Bu örnek, kaynak yöneticisi olarak SMOKIN kullanarak syncpointleme işlevini göstererek, kuyruklara birden çok kez ileti koymanıza olanak sağlar.

The sample server program `amqstxsx` must be running for the put sample to succeed; the server sample program connects to the queue manager and uses the XA interface. Örnek girişi çalıştırmak için:

- `doputs -n queuename -b batchsize -c tranccount -t message`

Örneğin:

- `doputs -n myqueue -b 5 -c 6 -t "Hello World"`

This puts 30 messages onto the queue named `myqueue`, in six batches, each with five messages in it. Herhangi bir sorun varsa, bir ileti kümesini dışarıda yedekler; tersi durumda, bu ileti bunları kesinleştirir.

Herhangi bir hata iletisi SMOKIN günlük dosyasına yazılır ve `stderr` 'e yazılır. Herhangi bir neden kodu `stderr` 'e yazılır.

Windows

UNIX

### SMOKIN için örnek al

Bu örnek, toplu iş kuyruğundan ileti almanıza olanak sağlar.

The sample server program `amqstxsx` must be running for the Get sample to succeed; the sample server program connects to the queue manager and uses the XA interface. Örneği çalıştırmak için şu komutu girin:

- `dogets -n queuename -b batchsize -c tranccount`

Örneğin:

- `dogets -n myqueue -b 6 -c 4`

Bu, her biri dört ileti içeren altı toplu iş kuyruğunda `myqueue` adlı kuyruğun üzerinden 24 ileti alır. Bu işlemi, `myqueue` 'ta 30 ileti koyan bir örnek örneğinden sonra çalıştırırsanız, `myqueue` ' ta yalnızca altı iletiniz olur. Toplu iş ve toplu iş büyüklüğünün sayısı, iletilerin yerleştirilip alınması arasında değişiklik gösterebilir.

Herhangi bir hata iletisi SMOKIN günlük dosyasına yazılır ve `stderr` 'e yazılır. Herhangi bir neden kodu `stderr` 'e yazılır.

Windows

### Windows üzerindeki SSPI güvenlik çıkışının kullanılması

This topic describes how to use the SSPI channel-exit programs on Windows systems. Sağlanan çıkış kodu iki biçimde olur: nesne ve kaynak.

## Nesne kodu

Nesne kodu dosyası `amqrspin.dll` olarak adlandırılır. For both client and server, it is installed as a standard part of IBM MQ for Windows in the `MQ_INSTALLATION_PATH/exits/INSTALLATION_NAME` folder. 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 kanala destek sağlamak için SVRCONN kanalına aşağıdaki bilgileri girin:

```
SCYDATA('remote_principal_name')
```

Burada *remote\_principal\_name* , *DOMAIN\kullanıcı* formunun içinde yer alıyor. Güvenli kanal, yalnızca uzak birincil kullanıcının adı *remote\_principal\_name* ile eşleşiyorsa kurulur.

To use the supplied channel-exit programs between systems that operate within a Kerberos security domain, create a **servicePrincipalName** for the queue manager.

## Kaynak kodu

Çıkış kaynak kodu dosyası *amqsspın.c* olarak adlandırılır. It is in *C:\Program Files\IBM\MQ\Tools\c\Samples*.

Kaynak kodu değiştirirseniz, değiştirilen kaynağı yeniden derlemeniz gerekir.

İlgili platforma ilişkin diğer kanal çıkışlarıyla aynı şekilde derlenir ve bağladınız; ancak, derleme sırasında SSPI üstbilgilerinin ve SSPI güvenlik kitaplıklarının, önerilen ilişkili kitaplıklarla birlikte, bağlantı sırasında erişilmesine gerek vardır.

Before you execute the following command, make sure that *c1.exe*, and the Visual C++ library, and the *include* folder are available in your path. Örneğin:

```
c1 /VERBOSE /LD /MT /Ipath_to_Microsoft_platform_SDK\include
/Ipath_to_IBM_MQ\tools\c\include amqsspın.c /DSECURITY_WIN32
-link /DLL /EXPORT:SCY_KERBEROS /EXPORT:SCY_NTLM STACK:8192
```

**Not:** Kaynak kod, izleme ya da hata işleme için herhangi bir hüküm içermiyor. Kaynak kodu değiştirir ve kullanır, kendi izleme ve hata işleme yordamlarınızı da ekleyebilirsiniz.

## Uzak kuyruklar kullanarak örnekleri çalıştırma

Bağlantılı kuyruk yöneticilerindeki örnekleri çalıştırarak uzak kuyruklama gösterebilirsiniz.

*amqscos0.tst* programı, başka bir uzak kuyruk yöneticisi kullanan bir uzak kuyruğun (SYSTEM.SAMPLE.REMOTE) yerel tanımlamasını sağlar. Bu örnek tanımlamasını kullanmak için, kullanmak istediğiniz ikinci kuyruk yöneticisinin adını değiştirin. Ayrıca, iki kuyruk yöneticiniz arasında bir ileti kanalı da ayarlamanız gerekir; bunun nasıl yapacağınıza ilişkin bilgi edinmek için [Kanalların tanımlanması](#) başlıklı konuya bakın.

İstek örnek programları, gönderdikleri iletilerin *ReplyToQMGr* alanına kendi yerel kuyruk yöneticisi adını koyar. Sorgula ve Set örnekleri, yanıt iletilerini, işlendikleri istek iletilerinin *ReplyToQ* ve *ReplyToQMGr* alanlarında belirtilen kuyruğa ve ileti kuyruğu yöneticisine gönderir.

## Küme Kuyruğu İzleme örnek programı (AMQSCLM)

This sample uses the built-in IBM MQ cluster workload balancing features to direct messages to instances of queues that have consuming applications attached. Bu otomatik yön, hiçbir tüketen uygulamanın eklenmediği bir küme kuyruğunun yönetim ortamında iletilerin birikmesini önler.

## Genel Bakış

Farklı kuyruk yöneticilerindeki aynı kuyruk için birden çok tanımlaması olan bir küme ayarlayabilirsiniz. Bu yapılandırma, artan kullanılabilirlik ve iş yükü dengelemesinin avantajını sağlar. Ancak, eklenen uygulamaların durumuna dayalı olarak bir küme genelinde iletilerin dağıtımını dinamik olarak değiştirmek için IBM MQ 'e yerleşik bir yetenek yoktur. Bu nedenle, iletilerin işlenmesini sağlamak için her zaman tüketen bir uygulama her zaman kuyruğun her örneğine bağlanmalıdır.

Küme kuyruğu izleme örnek programı, bağlı uygulamaların durumunu izler. Program, yerleşik iş yükü dengeleme yapılandırmasını, iletileri tüketen uygulamaları tüketerek kümelenmiş bir kuyruğun eşgörünümlerine yönlendirmek için dinamik olarak ayarlar. Belirli durumlarda bu program, bir uygulamanın her zaman bir kuyruğun her örneğine bağlı olması gerisini rahatlatılabilir için kullanılabilir. Ayrıca, bir kuyruğun somut örneğinde kuyruğa yollanan ve hiçbir uygulama ekli olmayan iletileri yeniden göndermektedir. İletilerin yeniden çevrilmesi, iletilerin geçici olarak kapatılan bir tüketici uygulama etrafında yönlendirilmesine olanak sağlar.



Bu program, uygulamaların sık sık eklenmesi ve alıkoyması yerine uzun süredir çalışmakta olan uygulamaların kullanıldığı bir yerde kullanılmak üzere tasarlanmıştır.

The cluster queue monitoring sample program is the compiled executable program of the C sample file amqsc1ma.c.

Kümelere ve iş yüküne ilişkin daha fazla bilgi, İş yükü yönetimi için kümelerin kullanılması başlıklı konuda bulunabilir.

*AMQSCLM: Örneği kullanmak için tasarım ve planlama*

Küme kuyruğu izleme örnek programının nasıl çalıştığı hakkında bilgi, örnek kaynak için bir sistem ayarlanırken göz önünde bulundurulması gereken noktalar ve örnek kaynak kodunda yapılacak değişiklikler.

## Tasarım

Küme kuyruğu izleme örnek programı, bağlı uygulamaları tüketen yerel kümelenmiş kuyrukları izler. Program, kullanıcı tarafından belirlenen kuyrukları izler. Kuyruğun adı belirli bir ad olabilir; örneğin, APP.TEST01 ya da soysal. Soysal adlar, PCF ' ye (Programmable Command Format; Programlar Komut Biçimi) uyan bir biçimde olmalıdır. Soysal ad örnekleri: APP.TEST\* ya da APP\*.

İzlenecek yerel bir kuyruğun somut örneğinin sahibi olan bir kümedeki her kuyruk yöneticisi, bir küme kuyruğu izleme örnek programının bir yönetim ortamının bu programa bağlanmasını gerektirir.

## Dinamik ileti yöneltmesi

Küme kuyruğu izleme örnek programı, kuyruğun herhangi bir tüketiciye sahip olup olmadığını belirlemek için bir kuyruğun **IPPROCS** (giriş işlemi sayısı için açık) değerini kullanır. 0 'dan büyük bir değer, kuyruğun en az bir uygulama tüketen uygulama olduğunu gösterir. Bu tür kuyruklar etkindir. 0 değeri, kuyruğun bağlı olmayan programların olmadığını gösterir. Bu tür kuyruklar etkin değildir.

For a clustered queue with multiple instances in a cluster, IBM MQ uses the cluster workload priority property **CLWLPRTY** of each queue instance to determine which instances to send messages to. IBM MQ , iletileri en yüksek **CLWLPRTY** değerine sahip bir kuyruğun kullanılabilir eşgörünümüne gönderir.

Küme kuyruğu izleme örneği programı, yerel **CLWLPRTY** değerini 1 değerine ayarlayarak bir küme kuyruğunu etkinleştirir. The program deactivates a cluster queue by setting its **CLWLPRTY** value to 0.

IBM MQ kümeleme teknolojisi, kümelenmiş bir kuyruğun güncellenen **CLWLPRTY** özelliğini kümedeki tüm ilgili kuyruk yöneticilerine dağıtabiliyor. Örneğin,

- Kuyruğa ileti koyan, bağlı bir uygulamaya sahip kuyruk yöneticisi.
- Aynı kümede aynı adı içeren bir yerel kuyruk sahibi olan bir kuyruk yöneticisi.

Yayma işlemi, kümenin tam havuz kuyruğu yöneticileri kullanılarak yapılır. Küme kuyruğuna ilişkin yeni iletiler, küme içinde en yüksek **CLWLPRTY** değerine sahip eşgörünümlere yönlendirilir.

## Kuyruğa yollanmış ileti aktarımı

**CLWLPRTY** değerinin dinamik olarak değiştirilmesi, yeni iletilerin yöneltmesini etkiler. Bu dinamik değişiklik, ekli bir tüketiciye sahip olmayan bir kuyruk örneğinde önceden kuyruğa alınan iletileri ya da değiştirilen bir **CLWLPRTY** değeri kümeye geçirilmeden önce iş yükü dengeleme mekanizmasından geçen iletiler etkilemez. Sonuç olarak, iletiler etkin olmayan bir kuyruğun üzerinde kalır ve tüketen bir uygulama tarafından işlenmez. Bunu çözmek için, küme kuyruğu izleme örnek programı, tüketiciye sahip olmayan yerel bir kuyruktan ileti alabilir ve bu iletileri, tüketicilerin bağlı olduğu kuyruğun uzak eşgörünümlerine gönderebilir.

Küme kuyruğu izleme örnek programı, iletileri işlem dışı bir yerel kuyruktan bir ya da daha çok etkin uzak kuyruğa aktarır ( **MQGET** kullanarak) ve iletileri koymak ( **MQPUT** ' un kullanılması) Aynı kümelenmiş kuyruğa. This transfer causes the IBM MQ cluster workload management to select a different target instance, based on a higher YAZD1RMA value than that of the local queue instance. İleti aktarımı sırasında ileti kalıcılığı ve bağlam korunur. İleti sırası ve bağ tanımlama seçenekleri korunmaz.

## Planlama

Küme kuyruğu izleme örnek programı, uygulamaların tüketilmesinde bir değişiklik olduğunda küme yapılandırmasını değiştirir. Değişiklikler, küme kuyruğu izleme örnek programının kuyrukları izleme, kümedeki tam havuz kuyruğu yöneticilerine iletilmekte olduğu kuyruk yöneticilerinden iletilir. Tüm havuz kuyruğu yöneticileri, yapılandırma güncelleştirmelerini işler ve bunları kümedeki ilgili tüm kuyruk yöneticilerine yeniden gönderir. İlgili kuyruk yöneticileri, aynı adı (küme kuyruğu izleme örnek programının bir örneği çalıştırıldığı) ve bir uygulamanın, son 30 gün içinde iletileri koymak için küme kuyruğunu açtıkları kuyruk yöneticilerine sahip olan kuyruk yöneticilerini içerir.

Değişiklikler, kümeden zamanuyumsuz olarak işlenir. Bu nedenle, her değişiklikten sonra, kümedeki farklı kuyruk yöneticilerinin bir süre yapılandırma için farklı görünümleri olabilir.

Küme kuyruğu izleme örnek programı, yalnızca uygulamaların sık sık bağlanma ya da ayırma (örneğin, uzun süredir çalışan uygulamalar) kullanan sistemler için uygundur. Uygulamaların tüketilmesinin yalnızca kısa süreler için eklendiği sistemleri izlemek için kullanıldığında, yapılandırma güncelleştirmelerini dağıtma işlemi sırasında oluşan gecikme süresi kümedeki kuyruk yöneticilerinin, tüketicilerin bağlı olduğu kuyrukların yanlış görünümüne sahip olmasına neden olabilir. Bu gecikme, yanlış yönlendirilmiş iletiler ile sonuçlanabilir.

Birçok kuyruk izlenirken, tüm kuyruklar boyunca bağlı tüketicilerde göreceli olarak düşük bir değişiklik oranı, küme içindeki küme yapılandırma trafiğini artırabilir. Artan küme yapılandırma trafiği, aşağıdaki kuyruk yöneticilerinden birinde ya da daha fazlasında aşırı yük ile sonuçlanabilir.

- Küme kuyruğu izleme örnek programının çalışmakta olduğu kuyruk yöneticileri
- Tüm havuz kuyruğu yöneticileri
- Kuyruğa ileti koyan, bağlı bir uygulamaya sahip kuyruk yöneticisi
- Aynı kümede aynı adı içeren bir yerel kuyruk sahibi olan bir kuyruk yöneticisi

Tam havuz kuyruğu yöneticilerindeki işlemci kullanımı değerlendirilmelidir. Ek işlemci kullanımı, tam havuz kuyruğu SYSTEM.CLUSTER.COMMAND.QUEUE. Bu kuyrukta ileti oluşturuyorsa, bu, tam havuz kuyruğu yöneticilerinin sistemdeki küme yapılandırma değişikliği hızına yetişemediğinden emin olun.

Birçok kuyruk, küme kuyruğu izleme örnek programı tarafından izlenirken, örnek program ve kuyruk yöneticisi tarafından gerçekleştirilen bir çalışma miktarı vardır. Bu iş, bağlı tüketicilerde herhangi bir değişiklik olmadığında da gerçekleştirilir. -i bağımsız değişkeni, izleme çevriminin sıklığını azaltarak yerel sistemde örnek programın işlemci kullanımını azaltmak için değiştirilebilir.

Aşırı etkinliği algılamaya yardımcı olmak için, küme kuyruğu izleme örnek programı, yoklama aralığı, geçen işleme süresi ve yapılandırma değişikliklerinin sayısı için ortalama işleme süresini bildiriyor. Raporlar, hangisi daha erken olursa olsun, bir bilgi iletisinde, **CLM0045I**, her 30 dakikada bir ya da her 600 yoklama aralığı içinde teslim edilir.

## Küme kuyruğu izleme kullanım gereksinimleri

Küme kuyruğu izleme örnek programının gereksinimleri ve kısıtlamaları vardır. Bu kısıtlamaların bazılarını değiştirmek için sağlanan örnek kaynak kodunu, bu kodun nasıl kullanılabileceği konusunda değiştirebilirsiniz. Bu bölümde listelenen örnekler, yapılabilen ayrıntılı değişikliklerle ilgili olarak sıralanabilir.

- Küme kuyruğu izleme örnek programı, uygulamaların kullanıldığı ya da bağlı olmadığı kuyrukları izlemek için kullanılmak üzere tasarlanmıştır. Sistem sık sık bağlanma ve ayırma işlemi yapan uygulamaları tüketiyorsa, örnek program tüm küme boyunca aşırı küme yapılandırma etkinliği oluşturabilir. Bu, kümedeki kuyruk yöneticilerinin performansı üzerinde etkili olabilir.
- Küme kuyruğu izleme örnek programı, temeldeki IBM MQ sistemi ve küme teknolojisine bağlıdır. İzlenmekte olan kuyrukların sayısı, izleme sıklığı ve her bir kuyruğun durumunun değişme sıklığı, genel sistemdeki yükü etkiler. İzlenecek kuyruklar ve izlemenin yoklama aralığı seçilirken bu etkenlerin göz önünde bulundurulması gerekir.

- Küme kuyruğu izleme örnek programının bir eşgörünümü, izlenecek bir kuyruğun somut örneğinin sahibi olan kümedeki her kuyruk yöneticisine bağlı olmalıdır. Bu örnek programı, kuyrukların sahibi olmayan kümeden kuyruk yöneticilerine bağlamak gerekmez.
- Küme kuyruğu izleme örnek programı, gereken tüm IBM MQ kaynaklarına erişmek için uygun bir yetkiyle çalıştırılmalıdır. Örneğin,
  - Bağlanılacak kuyruk yöneticisi
  - SYSTEM.ADMIN.COMMAND.QUEUE
  - İleti aktarımı gerçekleştirildiğinde izlenecek tüm kuyruklar
- Komut sunucusu, küme kuyruğu izleme örnek programı bağlı her kuyruk yöneticisi için çalışır durumda olmalıdır.
- Küme kuyruğu izleme örnek programının her bir eşgörünümü, bağlı olduğu kuyruk yöneticinde yerel (kümelenmemiş) bir kuyruk için dışlayıcı kullanımı gerektirir. Bu yerel kuyruk, örnek programı denetlemek ve kuyruk yöneticisinin komut sunucusuna yapılan sorgulardan yanıt iletilerini almak için kullanılır.
- Küme kuyruğu izleme örnek programının tek bir eşgörünümü tarafından izlenilecek tüm kuyruklar aynı kümede olmalıdır. Bir kuyruk yöneticisinin izleme gerektiren birden çok kümede kuyrukları varsa, örnek programın birden çok örneği gereklidir. Her yönetim ortamının denetim ve yanıt iletileri için yerel bir kuyruk olması gerekir.
- İzlenecek tüm kuyrukların tek bir kümede olması gerekir. Küme ad listesini kullanmak için konfigürasyonu tanımlanmış kuyruklar izlenmez.
- İletilerin etkin olmayan kuyruklardan aktarılabilmesini sağlamak isteğe bağlıdır. Bu, küme kuyruğu izleme örnek programının eşgörünümü tarafından izlenmekte olan tüm kuyruklara uygulanır. Yalnızca, izlenmekte olan kuyrukların bir alt kümesi ileti aktarımı için etkinleştirilmişse, küme kuyruğu izleme örnek programının iki eşgörünümü gerekir. Bir örnek programda ileti aktarımı etkinleştirildi, diğerinde ise ileti aktarımı devre dışı bırakıldı. Örnek programın her bir eşgörünümü, denetim ve yanıt iletileri için yerel bir kuyruğa gerek duyar.
- IBM MQ küme iş yükü dengelemesi, varsayılan olarak, iletilerin bağlı olduğu kuyruk yöneticisinde bulunan kümelenmiş kuyrukların eşgörünümlerine ileti gönderir. Bu durum, yerel kuyruk aşağıdaki durumlarda etkin değilken devre dışı bırakılmalıdır:
  - Uygulamaların, izlenmekte olan etkin olmayan bir kuyruğun eşgörünümlerine sahip kuyruk yöneticilerine bağlanması
  - Kuyruğa alınan iletiler, etkin olmayan kuyruklardan etkin kuyruklara aktarılıyor.

The local workload balancing preference on the queue can be disabled statically, through setting the **CLWLUSEQ** value to HER. Yerel kuyruklar içeren bu yapılanış iletilerinde, yerel ve uzak kuyruk eşgörünümlerine, yerel tüketen uygulamalar olsa bile, iş yükünü dengelemek için dağıtılır. Alternatively, the cluster queue monitoring sample program can be configured to temporarily set the **CLWLUSEQ** value to HER while the queue has no attached consumers which results in only local messages going to local instances of a queue while that queue is active.

- IBM MQ sistemi ve uygulamaları, izlenilecek kuyruklar ya da kullanılmakta olan kanallar için **CLWLPRTY** kullanılmamalıdır. Ters durumda, küme kuyruğu izleme örnek programının **CLWLPRTY** kuyruk özneliklerine ilişkin işlemleri istenmeyen etkilerden olabilir.
- Küme kuyruğu izleme örnek programı, bir rapor dosyaları kümesine ilişkin çalıştırma zamanı bilgilerini günlüğe 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, bir kuyruk yöneticisine yerel olarak bağlı olarak ya da bir kanal üzerinden bağlı bir istemci olarak çalıştırılabilir. Kuyruk yöneticisi çalışırken bu örnek çalıştırılmalıdır; yerel olarak çalışırken, kuyruk yöneticisiyle birlikte örneği otomatik olarak başlatmak ve durdurmak için kuyruk yöneticisi hizmeti olarak 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 örneği, dışlayıcı iç kullanım için yerel olmayan bir kuyruğa alma kuyruğuna gereksinim duyar. Kuyruğun adını seçebilirsiniz. Örnek, AMQSCLM .CONTROL .QUEUE adını kullanır. Örneğin, Windows üzerinde, aşağıdaki **MQSC** komutunu kullanarak bu kuyruğu oluşturabilirsiniz:

```
DEFINE QLOCAL (AMQSCLM.CONTROL.QUEUE)
```

You can leave the values of **MAXDEPTH** and **MAXMSGL** as default.

2. Hata ve bilgi iletisi günlükleri için bir dizin oluşturun.

Örnek, rapor dosyalarına tanımlama iletileri yazar. Dosyaların saklanacak bir dizin seçmeniz gerekir. Örneğin, Windows üzerinde, aşağıdaki komutu kullanarak bir dizin yaratabilirsiniz:

```
mkdir C:\AMQSCLM\irpts
```

Örnek tarafından oluşturulan rapor dosyaları aşağıdaki adlandırma kuralına sahiptir:

```
QmgrName.ClusterName.RPTOn.LOG
```

3. (İsteğe bağlı) Küme kuyruğu izleme örneğini IBM MQ hizmeti olarak tanımlayın.

Kuyrukları izlemek için, örnek her zaman çalışır durumda olmalıdır. Küme kuyruğu izleme örneğinin her zaman yürütülmesini sağlamak için, örneği kuyruk yöneticisi hizmeti olarak tanımlayabilirsiniz. Örneği hizmet olarak tanımlamak, AMQSCLM 'nin kuyruk yöneticisi başlatıldığında başlatıldığı anlamına gelir. Küme kuyruğu izleme örneğini bir 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\irpts') +
  stdout ('C:\AMQSCLM\irpts\+QMNAME+.TSTCLUS.stdout.log') +
  stderr ('C:\AMQSCLM\irpts\+QMNAME+.TSTCLUS.stderr.log')
```

Tanımlama	Tanım
<b>service</b>	Hizmet adını belirtir. Hizmet adını seçebilirsiniz.
<b>descr</b>	Hizmetin metin tanımlamasını belirler.
<b>control</b>	Hizmetin kuyruk yöneticisiyle aynı anda başlatıldığını ve durduğunu gösterir.
<b>servtype</b>	Bu kuyruk yöneticisi için bir kerede yalnızca bir yönetim ortamı anlamına gelen bir sunucu hizmeti nesnesini belirtir.
<b>startcmd</b>	Programın yerini ve adını belirler.
<b>startarg</b>	Örneğe ilişkin bağımsız değişkenleri belirtir. + QMNAME + kullanımını not edin. Kuyruk yöneticisinin adı otomatik olarak yerine konur.
<b>stdout</b>	Standart çıkışın yeniden yönlendirileceği tam olarak nitelenmiş dosya adı. Örnek, yalnızca bu örneğin sonlandığı doğrulayan iletiler için bu dosyaya yazar. Standart hata dosyası, örnek sonlandırma işleminin daha önceki bir aşamasında önceden kapandığı için bu örnek bunu yapar.

Tanımlama	Tanım
<b>stderr</b>	Standart hata çıkışının yeniden yönlendirileceği tam olarak nitelenmiş dosya adı. Örnek, örnek olarak sona erdirilmeden önce hata iletilerine ilişkin standart hata dosyasına yazar.

### Bu görev hakkında

Bu görev, küme kuyruğu izleme örneğini farklı şekillerde başlamanıza ve durdurmanıza olanak sağlar. Ayrıca, örneği, izlenmekte olan kuyruklara ilişkin istatistik bilgilerini içeren rapor dosyaları oluşturan bir kipte çalıştırmanıza da olanak sağlar.

Örnek program aşağıdaki komutu kullanarak çalıştırılabilir.

```
AMQSCLM -m QMgrName -c ClusterName (-q QNameMask | -f QListFile) -r MonitorQName  
[-l ReportDir] [-t] [-u ActiveVal] [-i Interval] [-d] [-s] [-v]
```

Bu çizelge, küme kuyruğu izleme örneğiyle birlikte kullanılabilir bağımsız değişkenleri ve her biri hakkında ek bilgi içeren bağımsız değişkenleri listeler.

Bağımsız Değişken	Değişken	Ek Bilgi
-m	QMGRName	İzlenecek kuyruk yöneticisi.
-c	ClusterName	İzlenecek kuyrukları içeren küme.
-q	QNameMask	İzlemek için kuyruklar ya da kuyruklar. Sonda bir * , sıfır ya da daha fazla sondaki karakterlerle eşleşen adlara sahip tüm kuyrukları izler.
-f	QListFile	İzlenecek kuyruk adlarının ya da kuyruk adı maskelerinin listesini içeren dosyanın tam yolu ve dosya adı. Dosya her satır için bir kuyruk adı/maske içermelidir. You can specify -q or -f, but not both.
-r	MonitorQName	Özel olarak, örnek tarafından kullanılan yerel kuyruk.
-l	ReportDir	Günlüğe kaydedilecek bilgi iletilerinin bir paket kaydırma kümesinde saklanacak dizin yolu <sup>9</sup> rapor dosyaları.
-t		(İsteğe bağlı) Kuyruğa alınan iletilerin etkin olmayan yerel kuyruklardan etkin kuyruklara aktarılmasını sağlar. Etkinleştirilmezse, yalnızca kümeye giren yeni iletiler bir kuyruğun etkin eşgörünümlerine dinamik olarak yönetilir.
-u	ActiveVal	(İsteğe bağlı) İzlenen bir kuyruk örneğinin <b>CLWLUSEQ</b> özelliğini etkinlik dışı olduğunda otomatik olarak ANY olarak ve etkin olduğunda <b>ActiveVal</b> değerine otomatik olarak değiştirir. <b>ActiveVal</b> , LOCAL ya da QMGRolabilir. Bu bağımsız değişken, uygulamaların aynı kuyruk yöneticisine bağlanması ya da ileti aktarımının etkinleştirildiği bir sistemde ayarlanmazsa, izlenen kuyrukların <b>CLWLUSEQ</b> değeri HERya da kuyruk yöneticisi HERdeğerine sahip MMGR olmalıdır.
-i	Interval	(İsteğe bağlı) Monitörün kuyrukları denetleyen saniye cinsinden zaman aralığı. Varsayılan değer 300 saniyedir (5 dakika).
-d		(İsteğe bağlı) Ek tanımlama çıkışı etkinleştirir. Hata ayıklama çıkışı, sistem ilk kez yapılandırılırken ya da örnek kodla çalışırken yararlı olabilir.
-s		(İsteğe bağlı) Aralık başına en düşük istatistiksel çıkışı etkinleştirir.
-v		(İsteğe bağlı) Rapor dosyalarının yanı sıra, rapor bilgilerini standard out' e (log) günlüğe kaydet.

Bağımsız değişken listesi örnekleri:

```
-m QMGR1 -c CLUS1 -f c:\QList.txt -r CLMQ -l c:\amqsc1m\1rpts -s
-m QMGR2 -c CLUS1 -q ABC* -r CLMQ -l c:\amqsc1m\1rpts -i 600
-m QMGR1 -c CLUSDEV -q QUEUE.* -r CLMQ -l c:\amqsc1m\1rpts -t -u QMGR -d
```

Örnek kuyruk listesi dosyası:

```
Q1
QUEUE.*
ABC
ABD
```

<sup>9</sup> Her kuyruk yöneticisi ve kuyruk birleşimi için, tam olarak üzerine yazıldığında, sabit büyüklükteki bir günlük dosyası oluşturulur. Günlüğe kaydedici her zaman aynı dosyaya yazar ve dosyanın önceki iki sürümünü de tutar.

## Yordam

1. Küme kuyruğu izleme örneğini başlatın. Örneği aşağıdaki yollardan birini kullanarak başlatabilirsiniz:

- Uygun kullanıcı yetkilendirmeleriyle bir komut istemi kullanın.
- Örnek bir IBM MQ hizmeti olarak yapılandırıldıysa, MQSC **START SERVICE** komutunu kullanın.

Her iki durumda da bağımsız değişken listesi aynıdır.

Örnek, program başlatıldıktan sonra 10 saniye boyunca kuyrukları izlemeyi başlatmaz. Bu gecikme, uygulamaların önce izlenen kuyruklara bağlanmasını, kuyruğun etkin durumunda gereksiz değişikliklerin yapılmasını önlemenizi sağlar.

2. Küme kuyruğu izleme örneğini durdurun. Kuyruk yöneticisi durdurulduğunda, durduğunda, susturulurken ya da kuyruk yöneticisiyle bağlantı kesilirse, örnek otomatik olarak durur. Kuyruk yöneticisini sonlandırmaksızın örneği durdurmanın yolları vardır:

- Get (Alma) işlevini geçersiz kılmak için yalnızca örnek tarafından kullanılan yerel kuyruğu yapılandırın.
- Send a message with a **CorrelId** of "KüMEYI DURDUR MONITOR\0\0\0\0", to the local queue used exclusively by the sample.
- Örnek işlemi sona erdirin. Bu, kalıcı olmayan iletilerin etkin kuyruklara aktarılmamasına neden olabilir. Ayrıca, sonlandırma işleminden sonra bir kaç saniye açık tutulan örnek tarafından kullanılan yerel kuyrukta da sonuçlanabilir. Bu durum, küme kuyruğu izleme örneğinin yeni bir örneğini hemen başlatmasını önler.

Örnek bir IBM MQ hizmeti olarak başlatıldıysa, **STOP SERVICE** hiçbir etkisine sahip değildir.

Kuyruk yöneticisinde yapılandırılmış bir **STOP SERVICE** mekanizması olarak tanımlanan sonlandırma yöntemlerinden birini kullanmak mümkündür.

## Sonraki adım

Örneğe ilişkin durumu denetleyin.

Raporlama etkinleştirilmişse, durum için rapor dosyalarını inceleyebilirsiniz. En güncel rapor dosyasını gözden geçirmek için aşağıdaki komutu kullanın:

```
QMGRName.ClusterName.RPT01.LOG
```

Eski rapor dosyalarını incelemek için aşağıdaki komutları kullanın:

```
QMGRName.ClusterName.RPT02.LOG  
QMGRName.ClusterName.RPT03.LOG
```

Rapor dosyaları, yaklaşık 1 MB ' lik bir boyut üst sınırına yetişir. RPT01 dosyası doldurulduğunda, yeni bir RPT01 dosyası oluşturulur. Eski RPT01 dosyası RPT02olarak yeniden adlandırıldı. RPT02 , RPT03olarak yeniden adlandırıldı. Eski RPT03 atılır.

Örnek, aşağıdaki durumlarda bilgi iletileri yaratır:

- Başlatma sırasında
- sona erdirme sırasında
- when it marks a queue **ACTIVE** or **INACTIVE**
- Etkin olmayan bir kuyruktan etkin bir yönetim ortamına ya da eşgörünümlere ileti isteğinde bulunduğu

Örnek, dikkat gerektiren bir sorunu bildirmek için *CLMnnnnE* hata iletisini yaratır.

Örnek raporlar her 30 dakikada bir, yoklama aralığı başına ortalama işleme süresi ve geçen işleme süresi bildirir. Bu bilgiler CLM0045İletisinde tutulur.

When statistical messages are enabled **-s**, the sample reports the following statistical information about each queue check:

- Kuyrukların işlenmesi için geçen süre (milisaniye cinsinden)
- Denetlenen kuyrukların sayısı
- Yapılan etkin/etkin olmayan değişiklik sayısı
- Aktarılan ileti sayısı

Bu bilgiler CLM0048Iiletisinde raporlanır.

Rapor dosyaları hata ayıklama kipinde hızla büyüyebilir ve hızlı bir şekilde paketleyebilir. Bu durumda, tek tek dosyalar için 1 MB ' lik büyüklük sınırı aşılabılır.

*AMQSCLM: Sorun Giderme*

Aşağıdaki kısımlarda, örnek kullanılırken karşılaşılabılır senaryolar hakkında bilgiler yer alır. Bir senaryoya ilişkin olası açıklamalarla ilgili bilgiler ve bu senaryoya nasıl çözülebileceği ile ilgili seçenekler sağlanır.

### **Senaryo: AMQSCLM başlatılmaz**

**Olası açıklama:** Sözdizimi yanlış.

**Yapılması gereken:** Doğru sözdizimi için standart hata çıkışını denetleyin

**Olası açıklama:** Kuyruk yöneticisi kullanılamıyor.

**Yapılması gereken:** İleti tanıtıcısı CLM0010E için rapor dosyasına bakın.

**Olası açıklama:** Rapor dosyası ya da dosyaları açılmıyor ya da oluşturulamıyor.

**Yapılması gereken:** Başlatma sırasında hata iletileri için standart hata çıkışını denetleyin.

### **Senaryo: AMQSCLM, bir kuyruğu ETKİN ya da DEVREDİŞİ olarak değiştirmiyor**

**Olası açıklama:** Kuyruk, izlenecek kuyruklar listesinde yok.

**İşlem:** **-q** ve **-f** parametre değerlerini denetleyin.

**Olası açıklama:** Kuyruk, doğru kümede yerel bir kuyruk değil.

**Yapılması gereken:** 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:** Bir tüketici olmadığı için, kuyruk INACTIVE (Etkin), **CLWLPRTY** = 0 olarak bırakılır. Diğer bir seçenek olarak, en az 1 tüketici olduğu için ETKİN **CLWLPRTY** > =1 olarak bırakılır.

**Yapılması gereken:** Tüketim uygulamalarının kuyruğa bağlı olup olmadığını denetleyin.

**Olası açıklama:** Kuyruk yöneticisinin komut sunucusu çalışmıyor.

**Yapılması gereken:** Hata olup olmadığını görmek için rapor dosyalarını denetleyin.

### **Senaryo: İletiler INACTIVE kuyrukları etrafında yönlendirilmiyor**

**Olası açıklama:** İletiler, doğrudan etkin olmayan kuyruğa sahip olan kuyruk yöneticisine doğrudan doğrulanır ve kuyruğun **CLWLUSEQ** değeri ANY(ANY) değildir ve **-u** bağımsız değişkeni AMQSCLM için kullanılmıyorsa, bu bağımsız değişken kullanılır.

**Eylem:** İlgili kuyruk yöneticisinin **CLWLUSEQ** değerini denetleyin ya da AMQSCLM için **-u** bağımsız değişkeninin kullanıldığından emin olun.

**Olası açıklama:** Kuyruk yöneticilerindeki herhangi bir etkin kuyruk yok. İletiler, bir kuyruk etkin duruma gelinceye kadar tüm etkin olmayan kuyruklar boyunca dengeli bir şekilde iş yüküne sahip olur.

**Yapılması gereken:** Tüm kuyruk yöneticilerindeki kuyrukların durumunu denetleyin.



**Olası açıklama:** İletiler, küme içindeki farklı bir kuyruk yöneticisine, etkin olmayan kuyruğa sahip olan bir kuyruk yöneticisine yerleştirilir ve güncellenen **CLWLPRTY** değeri 0, koyma uygulamasının kuyruk yöneticisine yayılmaz.

**Yapılması gereken:** İzlenen kuyruk yöneticisi ile tam havuz kuyruk yöneticisi arasındaki küme kanallarının çalışır durumda olup olmadığını denetleyin. Koyma kuyruk yöneticisi ve tam havuz kuyruk yöneticisi arasındaki kanalların çalışır durumda olup olmadığını denetleyin. İzlenen, koyulan ve tam havuz kuyruk yöneticilerine ilişkin hata günlüklerini denetleyin.

**Olası açıklama:** Uzak kuyruk eşgörünümleri etkin (**CLWLPRTY=1**), ancak yerel kuyruk yöneticisinden gelen küme gönderen kanalı çalışmadığı için, iletiler o kuyruk eşgörünümlerine yöneltilemez.

**Yapılması gereken:** Yerel kuyruk yöneticisinden uzak kuyruk yöneticisine ya da yöneticilere, kuyruğun etkin bir eşgörünümüyle birlikte, küme gönderen kanallarının durumunu denetleyin.

## **Senaryo: AMQSCLM, etkin olmayan bir kuyruktan ileti aktarmıyor**

**Olası açıklama:** İleti aktarımı etkinleştirilmedi ( **-t** ).

**Yapılması gereken:** İleti aktarımlarının etkinleştirildiğini doğrulayın ( **-t** ).

**Olası açıklama:** Kuyruk, izlenecek kuyruklar listesinde yer almıyor.

**İşlem:** **-q** ve **-f** parametre değerlerini denetleyin.

**Olası açıklama:** AMQSCLM, kümedeki bu ya da aynı kuyruğun somut örneklerinin sahip olduğu diğer kuyruk yöneticileri için çalışmamaktadır.

**İşlem:** AMQSCLM ' yi başlatın.

**Olası açıklama:** Kuyruğun **CLWLUSEQ** = LOCAL ya da **CLWLUSEQ** = QMGROlduğu ve **-u** bağımsız değişkeninin tanımlı olmadığı bir kuyruk vardır.

**Yapılması gereken:** **-u** parametresini ayarlayın ya da kuyruğu ya da kuyruk yöneticisi yapılanışını ANYolarak değiştirin.

**Olası açıklama:** Kümede kuyruğun etkin somut örneği yok.

**Eylem:** Kuyruğun eşgörünümlerini **CLWLPRTY** değerini 1 ya da daha büyük bir değer ile denetleyin.

**Olası açıklama:** Uzak kuyruk eşgörünümlerinin tüketiciler ( **IPPROCS** > = 1) olmasına karşın, AMQSCLM bu uzak yönetim ortamlarını izlemediği için, bu kuyruk yöneticilerindeki ( **CLWLPRTY** = 0) etkin değildir.

**Eylem:** AMQSCLM ' nin bu kuyruk yöneticilerinde çalıştırıldığından ve/veya kuyruk, **-q** ve **-f** parametre değerlerini denetleyerek izlenecek kuyruklar listesinde olduğundan emin olun.

**Olası açıklama:** Uzak kuyruk örnekleri etkin ( **CLWLPRTY** = 1), ancak yerel kuyruk yöneticinde etkin değil ( **CLWLPRTY** = 0) olarak görülür. 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 bağlı olduğundan emin olun. Tam havuz kuyruğu yöneticilerinin doğru şekilde çalıştığından emin olun. Tüm havuz kuyruğu yöneticileri ve izlenen kuyruk yöneticileri arasındaki kanalların çalışır durumda olup olmadığını denetleyin.

**Olası açıklama:** İletiler kesinleştirilmedi, bu nedenle yeniden denenemez.

**Yapılması gereken:** Gönderme uygulamasının doğru çalışıp çalışmadığını denetleyin.

**Olası açıklama:** AMQSCLM, iletilerin kuyruğa alındığı yerel kuyruğa erişime sahip değildir.

**İşlem:** AMQSCLM ' in kuyruğa erişmek için yeterli yetkisi bulunan bir kullanıcı olarak çalışıp çalışmadığını denetleyin.

**Olası açıklama:** Kuyruk yöneticisinin komut sunucusu çalışmıyor.

**Yapılması gereken:** Kuyruk yöneticisinin komut sunucusunu başlatın.

**Olası açıklama:** AMQSCLM bir hata saptadı.

**Yapılması gereken:** Hata olup olmadığını görmek için rapor dosyalarını denetleyin.

**Olası açıklama:** Uzak kuyruk eşgörünümleri etkin (CLWLPRTY=1), ancak yerel kuyruk yöneticisinden gelen küme gönderen kanalı çalışmadığı için, iletiler kuyruk örneklerine aktarılamıyor. Bu genellikle amqsclm rapor günlüğünde bir CLM0030W uyarısı ile birlikte gönderilir.

**Yapılması gereken:** Yerel kuyruk yöneticisinden uzak kuyruk yöneticisine ya da yöneticilere, kuyruğun etkin bir eşgörünümlüyle birlikte, küme gönderen kanallarının durumunu denetleyin.

### **ULW** **Bağlantı Uç Noktası Araması örnek programı (CEPL)**

IBM MQ Connection Endpoint Lookup örneği, IBM MQ kullanıcılarına Tivoli Directory Server gibi bir LDAP havuzundan bağlantı tanımlarını alma olanağı sunan, basit ve güçlü bir çıkış modülü sağlar.

CEPL 'yi kullanabilmek için Tivoli Directory Server v6.3 Client kurulu olmalıdır.

Bu örneği kullanmak için desteklenen platformlarda çalışan bir IBM MQ denetimi bilgisine sahip olması gerekir.

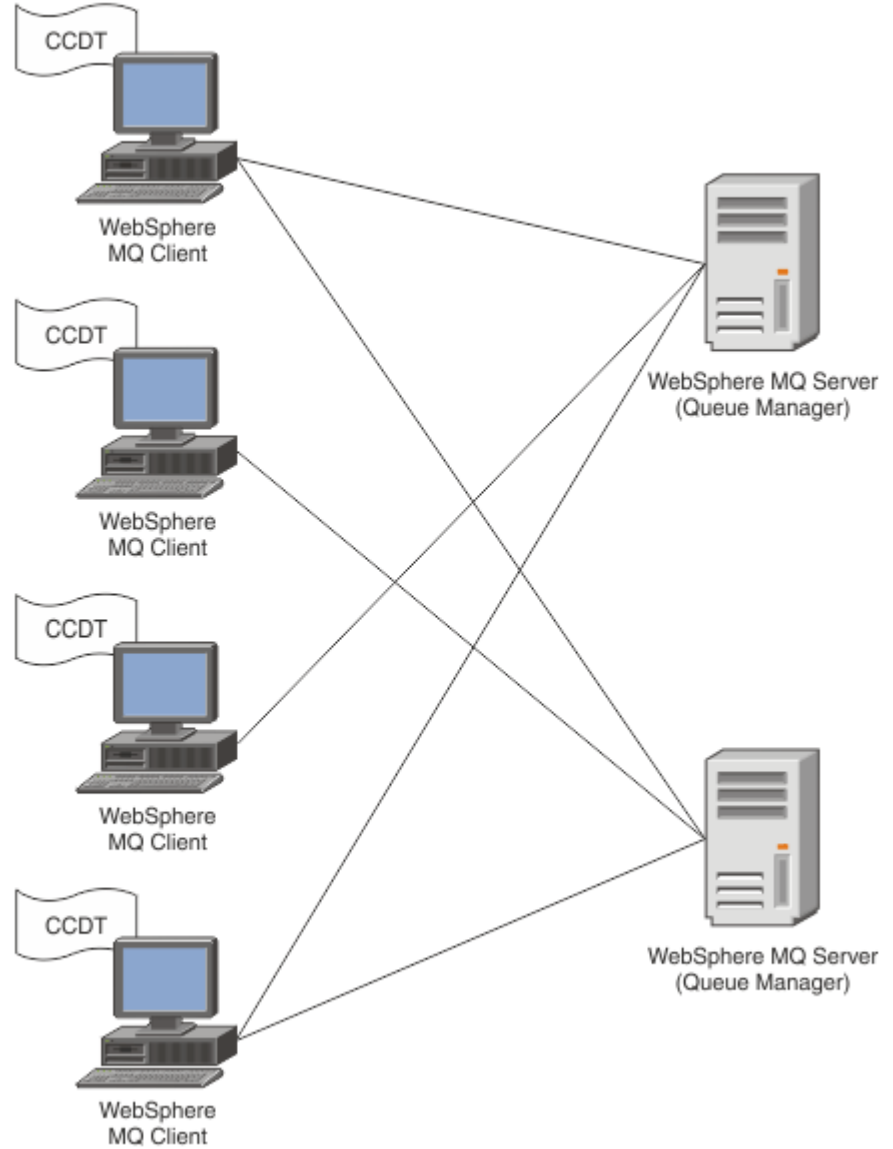
**Solaris** **Linux** **Windows** **AIX** Giriş

Bir LDAP (Lightweight Directory Access Protocol; LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) dizini gibi bir genel havuzu, istemci bağlantısı tanımlamalarını bakım ve yönetime yardımcı olacak şekilde saklamak için

Client Connection Definition Table (CCDT) aracılığıyla bir Kuyruk Yöneticisine bağlantı kurmak için IBM MQ Client uygulamasını kullanma.

CCDT, standart IBM MQ MQSC Administration arabirimi aracılığıyla yaratılır. Tanımın içindeki veriler Kuyruk Yöneticisi ile sınırlı olmasa da, kullanıcının istemci bağlantısı tanımlamaları yaratabilmek için bir Kuyruk

Yöneticisine bağlanması gerekir. Oluşturulan CCDT dosyasının istemci makineleri ve uygulamalar arasında



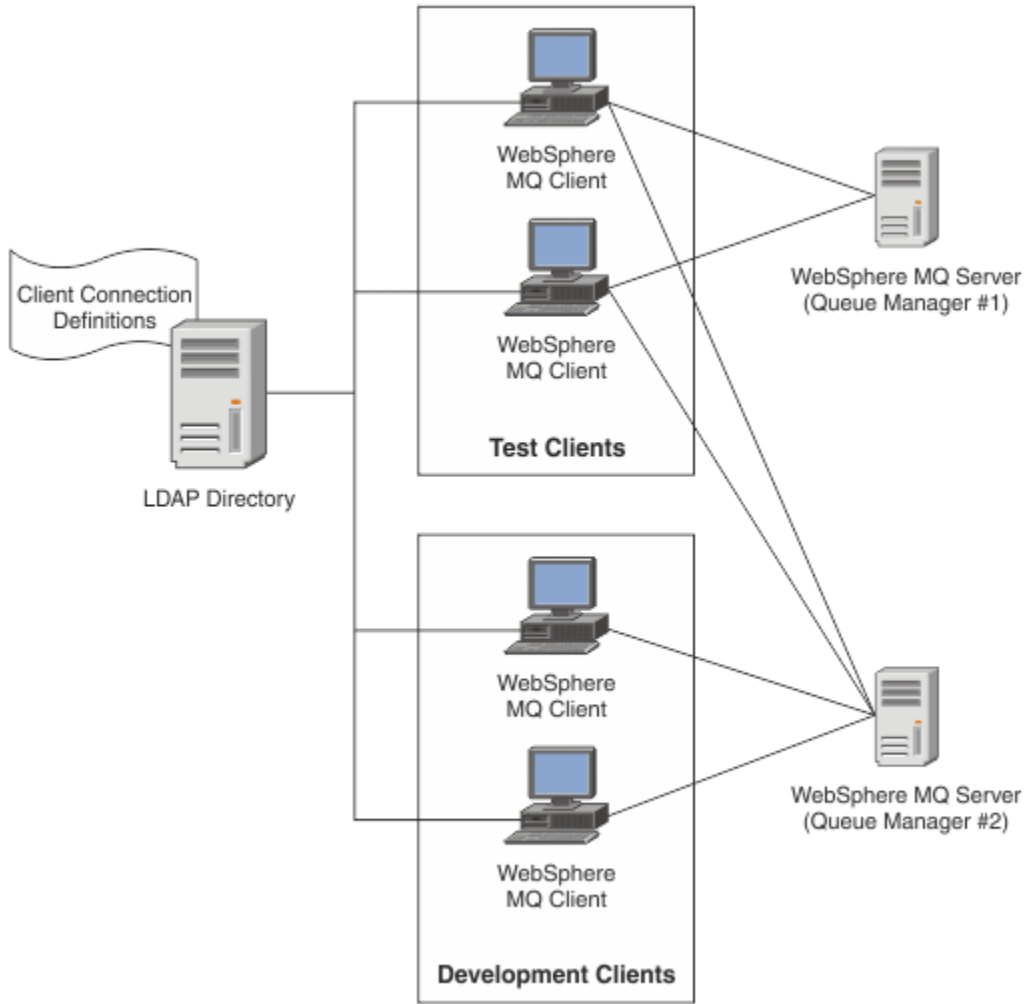
el ile dağıtılması gerekir.

CCDT dosyasının her bir IBM MQ istemcisine dağıtılması gerekir. Binlerce müşterinin yerel ya da küresel olarak bulunabildiği durumlarda, kısa sürede bakımı ve yönetimi zor hale gelir. Her bir istemcinin uygun istemci tanımlarına sahip olduğundan emin olmak için daha esnek bir yaklaşıma gerek vardır.

Bu tür bir yaklaşım, istemci bağlantısı tanımlamalarının LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) dizini gibi genel bir havuzda saklamasıdır. Ayrıca, bir LDAP dizini ek güvenlik, izin oluşturma ve arama olanakları da sağlayabilir; böylece, her bir istemci, yalnızca kendileriyle ilgili bağlantı tanımlarına erişebilir.

LDAP dizini, belirli kullanıcı grupları için yalnızca belirli tanımların kullanılabilir olması için yapılandırılabilir. Örneğin, Test İstemcileri hem Kuyruk Yöneticisi.hem de #2'a

erişebilir; ancak Geliştirme İstemcileri yalnızca Kuyruk Yöneticisi #2 ' e erişebilir.



Çıkış modülü, kanal tanımlamalarını almak için bir LDAP havuzunu (örneğin, IBM Tivoli Directory Server) arayabiliyor. Bu bağlantı tanımlarını kullanarak, bir IBM MQ istemcisi uygulaması kuyruk yöneticisiyle bağlantı kurabilir.

Çıkış modülü, bir LDAP havuzundan gelen MQCONN/MQCONNX çağrısı sırasında kanal tanımlamasının elde edilebilmesini sağlayan bir ön bağlanma çıkış modülüdür.

Çıkış modülü ve şema aşağıdaki gibi uygulanabilir:

- Var olan CCDT dosya tabanlı teknolojiyi kullanarak bir beceri tabanı kurmuş olan ve yönetim ve dağıtım maliyetlerini hafifletmek isteyen müşteriler.
- Müşteri bağlantısı tanımlarını dağıtmak için kendi istiflerini önceden kullanan mevcut müşteriler.
- Şu anda herhangi bir istemci bağlantısı çözümü kullanmayan ve IBM MQ tarafından sunulan özellikleri kullanmak isteyen yeni ya da var olan müşteriler.
- Herhangi bir güncel LDAP iş mimarisine yerleşik ileti modelini doğrudan kullanmak veya ayarlamak isteyen yeni veya var olan müşteriler.

#### **ULW** Desteklenen ortamlar

Connection Endpoint Lookup örneğini çalıştırmadan önce, desteklenen bir işletim sistemi ve ilgili yazılıma sahip olduğuna doğrulayın.

IBM MQ Connection Endpoint Lookup için örnek program, aşağıdaki yazılımları gerektirir:

- IBM WebSphere MQ 7.0'a da üstü

- Tivoli Directory Server V6.3 İstemcisi ya da üstü

Desteklenen işletim sistemleri:

1. **Windows** Windows (7/8/2008/2012)
2. **Solaris** Solaris (SPARC ve x86-64)
3. **AIX** AIX
4. **Linux** Linux
  - RHEL v4 ve System püzerinde v5
  - SUSE v9 ve System püzerinde 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** z/OS
- **IBM i** IBM i

**ULW** *Kuruluş ve yapılandırma*

Çıkış modülü ve bağlantı uç noktası şeması kuruluyor ve yapılandırılıyor.

## Çıkış modülü kurulumu

IBM MQkuruluşu sırasında, çıkış modülü `tools/samples/c/preconnexit/bin` altında kurulur. 32 bit altyapılar için, çıkış modülünün kullanılabilmesi için `exit/installation_name/` 'a kopyalanması gerekir. 64 bit altyapılar için, çıkış modülünün `exit64/installation_name/` kullanılabilmesi için kopyalanması gerekir.

## Bağlantı Uç Noktası şeması kurulumu

Çıkış, Bağlantı Uç Noktası şemasını ( `ibm-amq.schema`) kullanır. Çıkış kullanılabilmesi için, şema dosyasının herhangi bir LDAP sunucusuna aktarılması gerekir. Şemayı içe aktardıktan sonra, özniteliklere ilişkin değerler eklenmelidir.

Burada, Bağlantı Uç Noktası şemasının içe aktarılmasına ilişkin bir örnek vardır. Örnek, IBM Tivoli Directory Server (ITDS) kullanıldığını varsayar.

- Ensure that IBM Tivoli Directory Server is running, then copy or FTP the `ibm-amq.schema` file to the ITDS server.
- ITDS sunucusunda, şemayı ITDS deposuna kurmak için şu komutu girin; burada `LDAP ID` ve `LDAP password`, LDAP sunucusunun kök ayırt edici adı ve paroladır:  
`ldapadd -D "LDAP ID" -w "LDAP password" -f ibm-amq.schema`
- Komut penceresinde, aşağıdaki komutu girin ya da doğrulama için şemaya göz atmak için bir üçüncü kişi aracı kullanın:

```
ldapsearch objectclass=ibm-amqClientConnection
```

Şema dosyasının içe aktarılmasına ilişkin ek bilgi için LDAP Sunucusu belgelerinize bakın.

## Yapılandırma

İstemci yapılandırma dosyasına (örneğin, `mqclient.ini`) PreConnect adı verilen yeni bir bölüm eklenmelidir. PreConnect kısmı aşağıdaki anahtar sözcükleri içerir:

Modül: API çıkış kodunu içeren modülün adı. Bu alan modülün tam yolunu içeriyorsa, olduğu gibi kullanılır. Ters durumda, IBM MQ kuruluşundaki `exit` ya da `exit64` klasörü aranır.

İşlev: İşlevsel giriş noktasının, PreConnect çıkış kodunu içeren kitaplığa ilişkin adı. İşlev tanımlaması MQ\_PRECONNECT\_EXIT prototipine uygun olur.

Veri: Kanal tanımlarını içeren LDAP havuzunun URI 'si.

Aşağıdaki kod parçası, `mqclient.ini` dosyasında gerekli olan değişikliklere bir örnektir.

```
PreConnect:
Module=amqclcelp
Function=PreConnectExit
Data=ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
Sequence=1
```

## ULW Çıkışa ve şemaya genel bakış

Sözdizimi ve kuyruk yöneticisiyle bağlantı kurmak için kullanılan parametreler.

IBM MQ 9.1 , çıkış modülündeki bir giriş noktası için aşağıdaki sözdizimini tanımlar.

```
void MQENTRY MQ_PRECONNECT_EXIT ( PMQNX pExitParms
                                   , PMQCHAR pQMgrName
                                   , PPMQCN ppConnectOpts
                                   , PMQLONG pCompCode
                                   , PMQLONG pReason)
```

MQCONN/X çağırısı yürütmesi sırasında, IBM MQ C İstemcisi işlev sözdiziminin somutlamasını içeren çıkış modülünü yükler. Daha sonra kanal tanımlamalarını almak için bir çıkış işlevini çağırır. Daha sonra, alınan kanal tanımlamaları kuyruk yöneticisiyle bağlantı kurmak için kullanılır.

## Parametreler

### pExitParms

Tip: PMQNX giriş/çıkış

PreConnection çıkış değiştirgesi yapısı. Yapı, çıkışa ilişkin çağırıcı tarafından ayrılır ve sürdürür.

```
struct tagMQNX
{
  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. On input, this parameter is the filter string supplied to the MQCONN API call through the **QMgrName** parameter. Bu alan boş, açık ya da belirli genel arama karakterleri içerebilir. Alan, çıkışa göre değiştirilir. Çıkış MQXR\_TERM ile çağırıldığında, parametre NULL olur.

### ppConnectSeçenekleri

Tip: ppConnectGiriş/çıkış

MQCONNX 'in işlemini denetleyen seçenekler. Bu, MQCONN API çağırısının işlemini denetleyen MQCNO bağlantı seçenekleri yapısına bir işaretir. Çıkış MQXR\_TERM ile çağırıldığında, parametre NULL

olur. MQI istemcisi, çıkışa her zaman, uygulama tarafından sağlanmamış olsa da, çıkışa bir MQCNO yapısı sağlar. Bir uygulama MQCNO yapısı sağlıyorsa, istemci bunu değiştirdiği yerden çıkışa geçirmek için bir yineleme yapar. İstemci MQCNO ' nun sahipliğini korur. MQCNO ile gönderme yapılan bir MQCD, dizi aracılığıyla sağlanan bağlantı tanımlarına göre öncelik kazanır. İstemci, kuyruk yöneticisine bağlanmak için MQCNO yapısını kullanıyor ve diğerleri yok sayılıyor.

### pCompKodu

Tip: PMQXX\_ENCODE\_CASE\_ONE long giriş/çıkış

Tamamlanma kodu. Çıkışların tamamlanma kodunu alan bir MQUZE işaretçisi. Bu değer aşağıdaki değerlerden biri olmalıdır:

- MQCC\_OK -Tamamlama tamamlandı
- MQCC\_UYARI -Uyarı (kısmi tamamlama)
- MQCC\_FAILED -Arama başarısız oldu

### pReason

Tip: PMQXX\_ENCODE\_CASE\_ONE long giriş/çıkış

Neden niteleyici pCompkodu. Çıkış neden kodunu alan bir MQUZE işaretçisi. Tamamlanma kodu MQCC\_OK ise, geçerli tek değer: MQRC\_NONE -(0, x'000') Raporlamak için herhangi bir neden yok.

Tamamlanma kodu MQCC\_FAILED ya da MQCC\_UYARI ise, çıkış işlevi neden kodu alanını geçerli bir MQRC\_\* değerine ayarlayabiliyor.

## ULW

### MQ LDAP Bağlam Bilgileri

Çıkış, bağlam bilgileri için aşağıdaki veri yapısını kullanır.

### MQNLDPCTX

MQNLDPCTX yapısı aşağıdaki C prototipine sahiptir.

```
typedef struct tagMQNLDPCTX MQNLDPCTX;
typedef MQNLDPCTX MQPOINTER PMQNLDPCTX;

struct tagMQNLDPCTX
{
    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 */
};
```

Solaris

Linux

Windows

AIX

Bağlantı uç noktası arama çıkışını oluşturmak için

örnek kod

You can use the sample code snippets for compiling the source on AIX, Linux, Solaris, and Windows.

## Kaynak derleniyor

Kaynağı herhangi bir LDAP istemci kitaplıkla (örneğin, IBM Tivoli Directory Server V6.3 istemci kitaplıkları) derleyebilirsiniz. Bu belgede, Tivoli Directory Server V6.3 istemci kitaplıklarını kullandığınızı varsayar.

**Not:** Bağlantı öncesi çıkış kitaplığı, aşağıdaki LDAP sunucularıyla desteklenir:

- IBM Tivoli Directory Server V6.3
- Novell eDirectory V8.2

Aşağıdaki kod parçacıklar, çıkışlar nasıl derleneceğini açıklar:

## Windows Compiling the exit on the Windows platform

Çıkış kaynağını derlemek için aşağıdaki parçacığı kullanabilirsiniz:

```
CC=cl.exe
LL=link.exe
CCARGS=/c /I. /DWIN32 /W3 /DNDEBUG /EHsc /D_CRT_SECURE_NO_DEPRECATED /Zl

# The libraries to include
LDLIBS=ws2_32.lib Advapi32.lib libibldapstatic.lib libibldapdbgstatic.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 derleyicisi ile derlenmiş IBM Tivoli Directory Server V6.3 Client kitaplıklarını kullanıyorsanız, IBM Tivoli Directory Server V6.3 Client kitaplıklarını Microsoft Visual Studio 2012ya da üstü derleyici ile derlerken uyarılar alabilirsiniz.

## Solaris Linux AIX Compiling the exit on AIX, Linux, or Solaris

Aşağıdaki kod parçacığı, çıkış kaynağının Linux' ta derlenmesi için. Bazı derleyici seçenekleri AIX ya da Solarisüzerinde farklı olabilir.

```
#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 gönderir, ancak yalnızca bir kitaplık tipini kullanabilirsiniz. Bu komut dosyası, statik kitaplıkları kullandığınızı varsayar.

```
#Use static libraries.
LDLIBS=-L$(TDSL) -libibldapstatic

CFLAGS=-I. -I$(MQMI) -I$(TDSI)

all:amqlcepl

amqlcepl: amqlcel0.c
$(CC) -o cepl amqlcel0.c -shared -fPIC $(XFLAG) $(CFLAGS) $(LDLIBS)
```

## ULW PreConnect çıkış modülünün çağırılması

PreConnect çıkış modülü, üç farklı neden koduyla çağırılabilir: MQXR\_INIT neden kodu, LDAP sunucusuna yönelik bir bağlantı başlatmak ve kurmak için, MQXR\_PRECONNECT neden kodu, bir LDAP sunucusundan kanal tanımlarını almak için neden kodu ya da çıkış temizlenmek üzere MQXR\_TERM neden kodu.

### MQXR\_INIT

Bu çıkış, bir LDAP sunucusuyla bağlantı kurulması ve kurulmasının sağlanması için MQXR\_INIT neden koduyla çağırılır.

MQXR\_INIT çağırısından önce, MQNXP yapısının pExitDataPtr alanı, mqclient.ini dosyasındaki (yani LDAP) PreConnect kısmında bulunan Veri özniteliğinden veri yerleştirilir.



LDAP URL adresi, arama için en az protokol, anasistem adı, kapı numarası ve temel DN ' den oluşur. Çıkış, pExitDataPtr alanında bulunan LDAP URL adresini ayrıştırır, bir MQNLDAPCTX LDAP Lookup Context yapısı ayırır ve buna uygun olarak veri yerleştirir. Bu yapının adresi, pExitUserAreaPtr alanında saklanır. LDAP URL sonuçlarını doğru olarak ayrıştırılmazsanız, MQCC\_FAILED hatası oluştu.

Bu noktada, çıkış, **MQNLDAPCTX** parametrelerini kullanarak LDAP sunucusuna bağlanır ve bağlanır. Sonuçta elde edilen 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 adı içeriyorsa, arama, belirtilen kuyruk yöneticisi adıyla **ibm-amQueueManagerName** LDAP öznitelik değerinin eşleşeceği tüm kanal tanımlarını döndürür.

**QMgrName** parametresi '\*' ya da ' ise ' (boşluk), **ibm-amIsClientDefault Connection** uç noktası özniteliğinin TRUEolarak ayarlanana ilişkin tüm kanal tanımlarını döndürür.

Başarılı bir aramadan sonra, çıkış bir ya da bir MQCD tanımı dizisi hazırlar ve çağıranı geri döndürür.

### **MQXR\_TERM**

Çıkış temizlenmek olduğunda, çıkış bu neden koduyla çağrılır. Bu temizleme işlemi sırasında, çıkış LDAP sunucusundan kesilir ve MQNLDAPCTX yapısı, işaretçi dizisi ve tüm MQCD ' ler de içinde olmak üzere, çıkışa göre ayrılan ve bakımı yapılan tüm belleği serbest bırakır. Diğer alanlar varsayılan değerlere ayarlanır. The **pQMgrName** and **ppConnectOpts** exit parameters are unused during an exit with the MQXR\_TERM reason code and may be BOŞ.

### **İlgili kavramlar**

[İstemci yapılandırma dosyasınınPreConnect kısmı](#)

### **ULW** LDAP şemaları

İstemci bağlantı verileri, LDAP (LDAP (Lightweight Directory Access Protocol; Temel Dizin Erişimi Protokolü) dizini adı verilen bir genel havuzda saklanır. Bir IBM MQ istemcisi, bağlantı tanımlarını almak için bir LDAP dizini kullanır. LDAP dizini içindeki IBM MQ istemcisi bağlantı tanımlarının yapısı LDAP şeması olarak bilinir. LDAP şeması, öznitelik tipi tanımlamalarının, nesne sınıfı tanımlamalarının ve sunucunun, bir süzgeç ya da öznitelik değeri değerlendirmesinin bir girişin öznitelikleriyle eşleşip eşleşmediğini ve işlemlerin izin, ekleme ve değiştirme işlemlerine izin verilip verilmeyeceğini belirlemek için kullandığı diğer bilgiler toplamlıdır.

### **Verileri LDAP dizininde saklama**

İstemci bağlantı tanımları, bağlantı noktası olarak bilinen dizin ağacı içindeki belirli bir dalın altında bulunur. Bir LDAP dizinindeki diğer tüm düğümler gibi, bağlantı noktasının ilişkili bir Ayırt Edici Adı (DN) vardır. Bu düğümü, dizin üzerinde yaptığınız tüm sorguların başlangıç noktası olarak kullanabilirsiniz. İstemci bağlantı tanımlarının bir alt kümesini döndürmek için LDAP dizini sorgulanırken süzgeç uygulamayı kullanın. Dizin ağacının diğer bölümlerinde verilen izinlere dayalı olarak alt ağaçlara erişimi kısıtlayabilirsiniz; örneğin, kullanıcılara, bölümlere ya da gruplara.

### **Kendi özniteliklerinizi ve sınıflarınızı tanımlama**

LDAP şemasını değiştirerek istemci kanalı tanımlamasını saklayın. Tüm LDAP veri tanımlamaları nesne ve öznitelik gerektirir. Nesnelere ve özniteliklere, nesneyi ya da özniteliği benzersiz bir şekilde tanımlayan bir nesne tanıtıcısı (OID) numarasıyla tanımlanır. Bir LDAP şemasının içindeki tüm sınıflar, doğrudan ya da dolaylı olarak üst nesneden devralır. İstemci kanalı tanımlaması nesnesi, üst nesnenin özniteliklerini içerir. Tüm LDAP veri tanımlamaları nesne ve öznitelik gerektirir:

- Nesne tanımlamaları, LDAP özniteliklerinin derlemeleridir.
- Öznitelikler LDAP veri tipleridir.

Her özniteliğin tanımı ve bu özelliklerin normal IBM MQ özellikleriyle nasıl eşleneceği [LDAP özniteliklerinde](#) açıklanmıştır.

Tanımlanan LDAP öznitelikleri IBM MQ ' a özeldir ve doğrudan istemci bağlantısı özellikleriyle eşlenir.

### IBM MQ Client Channel Dizin Dizgi Öznitelikleri

The character string attributes with their mapping to IBM MQ properties are listed in the following table. Öznitelikler, directoryString (UTF-8 kodlamalı Unicode) değerlerini, bir alt küme olarak IA5/ASCII içeren bir değişken bayt kodlama sistemi olan değerleri tutabilir. Sözdizimi, nesne tanıtıcısı numarası (OID) ile belirtilir.

Çizelge 173. IBM MQ istemci kanal dizini dizgi öznitelikleri		
LDAP Özniteliği	Tanım	IBM MQ Özellik
<u>CN</u>	Kanal adından ve tanımlayıcı kuyruk yöneticisi adına sahip ortak ad.	
<u>ibm-amqChannelAdı</u>	Kanal tanımlamasının adı.	Kanal
<u>ibm-amqConnectionAdı</u>	İletişim bağlantısı tanıtıcısı.	ADı
<u>ibm-amqDescription</u>	Kanal açıklaması.	TASARIMLA
<u>ibm-amqLocalAdresi</u>	Kanala ilişkin yerel iletişim adresi.	KAPSAYICI
<u>ibm-amqModeAdı</u>	LU 6.2 kip adı.	MODENAME
<u>ibm-amqPassword</u>	Kullanılabilecek parola.	Parola
<u>ibm-amqQueueManagerName</u>	Bir IBM MQ istemcisi uygulamasının bağlantı isteyebileceği kuyruk yöneticisi ya da kuyruk yöneticisi grubunun adı.	QMNAME
<u>ibm-amqSecurityExitUserVerileri</u>	Güvenlik çıkışa geçirilen kullanıcı verileri.	SCYDATA
<u>ibm-amqSecurityExitName</u>	Kanal güvenliği çıkışıyla çalıştırılacak çıkış programının adı.	SCYEXIT
<u>ibm-amqSslCipherSpec</u>	TLS bağlantısı için tek bir CipherSpec .	SSLCIPH
<u>ibm-amqSslPeerName</u>	Bir IBM MQ kanalının diğer ucundaki eş kuyruk yöneticisinden ya da istemciden gelen sertifikana ilişkin ayırt edici adı (DN) denetler.	SSLPEER
<u>ibm-amqTransactionProgramName</u>	Hareket programı adı.	TADı
<u>ibm-amqUserTanıtıcısı</u>	Uzak MCA ile güvenli bir SNA oturumu başlatma girişiminde bulunduğu MCA tarafından kullanılacak kullanıcı kimliği.	USERID

### IBM MQ istemci bağlantısı tamsayı öznitelikleri

Önceden tanımlanmış değerlere sahip öznitelikler (örneğin, bir sıralı tip) standart tamsayılar olarak depolanır. Bu değerler LDAP dizininde tamsayı değerleri olarak saklanır ve ilişkili sabit ad kullanılarak değil.

Çizelge 174. IBM MQ istemci kanalı izin tamsayı öznitelikleri		
LDAP Özniteliği	Tanım	IBM MQ Özellik
<u>ibm-amqConnectionAffinity</u>	Aynı kuyruk yöneticisi adı üzerinden birden çok kez bağlanan istemci uygulamalarının aynı istemci kanalını kullanıp kullanmayacağını belirler.	BENZERLIK

<i>Çizelge 174. IBM MQ istemci kanalı dizin tamsayı öznitelikleri (devamı var)</i>		
<b>LDAP Özniteliği</b>	<b>Tanım</b>	<b>IBM MQ Özellik</b>
<a href="#">ibm-amqClientChannelWeight</a>	İstemci bağlantı kanalı tanımlamasının kullanıldığı etki alanı ağırlıklandırma.	CLNTWGHT
<a href="#">ibm-amqHeartBeatInterval</a>	İletim kuyruğunda ileti olmadığına, gönderen MCA ' dan geçirilecek sağlıklı işletim bildirim akışları arasındaki yaklaşık süre.	HBNT
<a href="#">ibm-amqKeepAliveInterval</a>	Bir kanala ilişkin zaman aşımı değeri.	KAINT
<a href="#">ibm-amqMaximumMessageLength</a>	Kanalda iletilebilecek ileti uzunluğu üst sınırı.	MAXMSGL
<a href="#">ibm-amqSharingKonusmaları</a>	Her bir TCP/IP kanalı yönetim ortamını paylaşan etkileşim sayısı üst sınırı.	SHARECNV
<a href="#">ibm-amqTransportTipi</a>	Kullanılacak iletim tipi.	TRPTYPE

#### **IBM MQ istemci kanalı boolean özniteliği**

Bu Boole özniteliği herhangi bir IBM MQ özelliği ile eşlenmez. Bu özniteliğin sözdizimi, bir boole değeri gösterir.

<i>Çizelge 175. IBM MQ istemci kanalı boolean özniteliği</i>	
<b>LDAP Özniteliği</b>	<b>Tanım</b>
<a href="#">ibm-amqIsClientDefault</a>	Bu Boole özniteliği, <a href="#">ibm-amqQueueManagerName</a> özniteliği tanımlı olmayan girişlerin aranması sorununu çözmek için tanımlanır.

#### **IBM MQ istemci kanal listesi öznitelikleri**

IBM MQ özellikleri, LDAP dizini içinde tek değerli, virgülle ayrılmış liste özniteliği olarak depolanır. Öznitelikler, diğer dizin dizesi öznitelikleriyle aynı şekilde tanımlanır. IBM MQ özelliklerine eşlemeleriyle birlikte liste öznitelikleri aşağıdaki tabloda açıklanmıştır.

<i>Çizelge 176. IBM MQ istemci kanal listesi öznitelikleri</i>		
<b>LDAP Özniteliği</b>	<b>Tanım</b>	<b>IBM MQ Özellik</b>
<a href="#">ibm-amqHeaderSıkıştırması</a>	Kanal tarafından desteklenen üstbilgi veri sıkıştırma tekniklerini içeren bir listedir.	KARMAŞIK
<a href="#">ibm-amqMessageSıkıştırması</a>	Kanal tarafından desteklenen ileti veri sıkıştırma tekniklerinin listesi.	MSG
<a href="#">ibm-amqSendExitUserVerileri</a>	Gönderme çıkışa geçirilen kullanıcı verileri.	SENDDATA
<a href="#">ibm-amqSendExitUserAdı</a>	Kanal gönderme çıkışıyla çalıştırılacak çıkış programının adı.	SENDEXIT
<a href="#">ibm-amqReceiveExitUserVerileri</a>	Alma çıkışa geçirilen kullanıcı verileri.	RVDATA
<a href="#">ibm-amqReceiveExitName</a>	Kanal tarafından çalıştırılacak kullanıcı çıkış programının adı, kullanıcı çıkışı alır.	RCVEXIT

#### **ULW Ortak Ad**

Ortak ad (CN), kanal adından ve tanımlayıcı kuyruk yöneticisi adından oluşur.

Bu önceden var olan bir öznitedir.

CN ' nin biçimi şöyledir:

```
CN=CHANNEL_NAME(DEFINING_Q_MGR_NAME)
```

Örneğin:

```
CN=TC1(QM_T1)
```

Bu öznitelik için yalnızca bir değer belirtebilirsiniz.

Bu öznitelik bir dizgi öznitelidir ve değerler büyük/küçük harfe duyarlı değildir. Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, alt şemada kullanılan, bir alt dizgiyle (örneğin, CN=jim \* bir öznitelige sahip CN=jim \* gibi) bir arama süzgecindeki özniteliğin davranışını belirten ve bir ya da daha fazla joker karakter içeren eşleştirme kuralıdır.

**ULW** *ibm-amqChannelAdı*

Bu öznitelik, kanal tanımlamasının adını belirtir.

Bu özniteliğin, büyük/küçük harf duyarlı olmayan en fazla 20 karakteri içeren tek bir dizgi değeri vardır. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, alt şemada kullanılan, bir alt dizgiyi kullanan ve bir ya da daha fazla joker karakter içeren bir arama süzgecindeki özniteliğin davranışını belirleyen eşleşen bir kuralıdır.

**ULW** *ibm-amqDescription*

Bu LDAP özniteliği kanal tanımlamasını sağlar.

Bu özniteliğin, büyük/küçük harf duyarlı olmayan tek bir dizgi değeri en çok 64 byte olmalıdır. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuralıdır.

**ULW** *ibm-amqConnectionAdı*

Bu LDAP özniteliği, iletişim bağlantı tanıtıcısıdır. Bu kanal tarafından kullanılacak iletişim bağlantılarını belirler.

Bu özniteliğin en çok 264 karakteri olan tek bir dizgi değeri vardır; bu, büyük/küçük harfe duyarlı değildir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuralıdır.

**ULW** *ibm-amqLocalAdresi*

Bu öznitelik, kanala ilişkin yerel iletişim adresini belirtir.

Bu özniteliğin en çok 48 karakterden oluşan tek bir dizgi değeri vardır; bu, büyük/küçük harfe duyarlı değildir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuralıdır.

**ULW** *ibm-amqModeAdı*

Bu öznitelik, LU 6.2 bağlantılarıyla birlikte kullanılmak içindir. Bir iletişim oturumu ayırma işlemi gerçekleştirildiğinde bağlantının oturum özellikleri için ek tanım sağlar.

Bu öznitelik, büyük/küçük harf duyarlı olmayan, tam olarak 8 karakterden oluşan tek bir dizgi değerine sahiptir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuralıdır.

### **ULW** *ibm-amqPassword*

Bu LDAP özniteliği, uzak MCA ile güvenli bir LU 6.2 oturumu başlatma girişimi sırasında MCA tarafından kullanılabilir bir parola belirtir.

Bu özniteliğin en çok 12 hanesi olan tek bir tamsayı değeri vardır. Bu, önceden var olan bir öznitelik değildir.

### **ULW** *ibm-amqQueueManagerName*

Bu öznitelik, bir IBM MQ istemcisi uygulamasının bağlantı isteyebileceği kuyruk yöneticisi ya da kuyruk yöneticisi grubunun adını belirtir.

Bu özniteliğin en çok 48 karakterden oluşan tek bir dizgi değeri vardır; bu, büyük/küçük harfe duyarlı değildir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksa, alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

### **İlgili başvurular**

[“ibm-amqIsClientDefault” sayfa 1191](#)

Bu Boole özniteliği, *ibm-amqQueueManagerName* özniteliğinin tanımlanmadığı arama girdilerinin sorununu çözer.

### **ULW** *ibm-amqSecurityExitUserVerileri*

Bu LDAP özniteliği, güvenlik çıkışa geçirilen kullanıcı verilerini belirtir.

Bu öznitelik, büyük/küçük harf duyarlı olmayan, en çok 999 karakterden oluşan tek bir dizgi değerine sahiptir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksa, alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

### **ULW** *ibm-amqSecurityExitName*

Bu LDAP özniteliği, kanal güvenliği çıkışıyla çalıştırılacak çıkış programının adını belirtir.

Kanal güvenlik çıkışı yoksa, boş bırakın.

Bu öznitelik, büyük/küçük harf duyarlı olmayan, en çok 999 karakterden oluşan tek bir dizgi değerine sahiptir. Bu öznitelik, çıkış öncesi bir öznitelik değildir.

Alt dizgi eşleştirme yoksa, alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

### **ULW** *ibm-amqSslCipherSpec*

Bu LDAP özniteliği, TLS bağlantısı için tek bir CipherSpec belirtir.

Bu özniteliğin, büyük/küçük harf duyarlı olmayan tek bir dizgi değeri en çok 32 karakter olmalıdır. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksa, alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

### **ULW** *ibm-amqSslPeerName*

Bu LDAP özniteliği, bir IBM MQ kanalının diğer ucundaki eşdüzey kuyruk yöneticisinden ya da istemciden alınan sertifikana ilişkin ayırt edici adı (DN) denetlemek için kullanılır.

Bu LDAP özniteliğinin, büyük/küçük harf duyarlı olmayan tek bir dizgi değeri en fazla 1024 byte olmalıdır. Bu önceden var olan bir şey değil.

Alt dizgi eşleştirme yoksa, alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

**ULW** *ibm-amqTransactionProgramName*

Bu LDAP özniteliği, hareket programı adını belirtir. Bu, LU 6.2 bağlantılarıyla kullanılmak içindir.

Bu özniteliğin, büyük/küçük harf duyarlı olmayan, en çok 64 karakterden oluşan tek bir dizgi değeri vardır. Bu önceden var olan bir şey değil.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

**ULW** *ibm-amqUserTanıtıcısı*

Bu LDAP özniteliği, uzak MCA ile güvenli bir SNA oturumu başlatma girişiminde bulunulduğunda MCA tarafından kullanılacak kullanıcı kimliğini belirtir.

Bu özniteliğin tek bir dizgi değeri tam 12 karakter (büyük/küçük harfe duyarlı değildir). Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

**ULW** *ibm-amqConnectionAffinity*

Bu LDAP özniteliği, aynı kuyruk yöneticisi adını kullanarak birden çok kez bağlantı yapan istemci uygulamalarının aynı istemci kanalını kullanıp kullanmadığını belirler.

Bu özniteliğin tek bir tamsayı değeri vardır. Bu, önceden var olan bir öznitelik değildir.

**ULW** *ibm-amqClientChannelWeight*

Bu LDAP özniteliği, hangi istemci bağlantısı kanal tanımlamasının kullanıldığını etkileyen bir ağırlıklandırma belirtir.

İstemci kanalı ağırlıklandırma özniteliği, birden çok uygun tanımlama kullanılabilir olduğunda istemci kanalı tanımlarının seçilmesini önlemek için kullanılır.

Bu özniteliğin tek bir tamsayı değeri vardır. Bu, önceden var olan bir öznitelik değildir.

**ULW** *ibm-amqHeartBeatInterval*

Bu LDAP özniteliği, iletim kuyruğunda herhangi bir ileti olmadığında, gönderen MCA ' dan geçirilecek sağlıklı işletim bildirimleri akışları arasındaki yaklaşık süreyi belirtir.

Bu özniteliğin tek bir tamsayı değeri vardır. Bu, önceden var olan bir öznitelik değildir. Varsayılan değer 1 'dir. Varsayılan değer, yürürlükteki MQSERVER ortam değişkeni işleminde ayarlıdır.

**ULW** *ibm-amqKeepAliveInterval*

Bu LDAP özniteliği, bir kanala ilişkin bir zaman aşımı değeri belirtmek için kullanılır.

Bu özniteliğin değeri, kanala ilişkin canlı tutma (Keepalive) zamanlaması belirterek, iletişim yığınının geçirilir. Bu seçeneği, her kanal için farklı bir canlı tutma değeri belirtmek için kullanabilirsiniz.

Bu özniteliğin tek bir tamsayı değeri vardır. Bu, önceden var olan bir öznitelik değildir.

**ULW** *ibm-amqMaximumMessageLength*

Bu LDAP özniteliği, kanalda iletilebilecek bir ileti uzunluğu üst sınırını belirtir.

Bu özniteliğin varsayılan değeri, yürürlükteki MQSERVER ortam değişkeni işlemine göre 104857600 'tür. Bu özniteliğin tek bir tamsayı değeri var ve bu öznitelik önceden var olan bir öznitelik değil.

**ULW** *ibm-amqSharingKonuşmaları*

Bu LDAP özniteliği, her bir TCP/IP kanalı yönetim ortamını paylaşan etkileşim sayısı üst sınırını belirtir.

Bu özniteliğin tek bir tamsayı değeri vardır. Bu öznitelik, önceden var olan bir öznitelik değil.

ULW

*ibm-amqTransportTipi*

Bu LDAP özniteliği, kullanılacak iletim tipini belirtir.

Bu özniteliğin tek bir tamsayı değeri vardır. Bu, önceden var olan bir öznitelik değildir.

ULW

*ibm-amqIsClientDefault*

Bu Boole özniteliği, *ibm-amqQueueManagerName* özniteliğinin tanımlanmadığı arama girdilerinin sorununu çözer.

Önyükleme öncesi çıkış birimleri genellikle, arama ölçütü olarak, LDAP sunucularında *ibm-amqQueueManagerName* özniteliğinin değerini içeren LDAP sunucularını arar. Bu tür bir sorgu, *ibm-amqQueueManagerName* öznitelik değerinin, MQCONN/X çağrısında belirlenen kuyruk yöneticisinin adıyla eşleştiği tüm girişleri döndürür. Ancak, istemci kanal tanımlama çizelgeleri (CCDT) kullanılırken, bir MQCONN/X çağrısında kuyruk yöneticisi adını boş olarak ayarlayabilir ya da adın önekini yıldız (\*) ile önleyebilirsiniz. Kuyruk yöneticisinin adı boş bırakılırsa, istemci varsayılan kuyruk yöneticisine bağlanır. Adın başına bir yıldız işareti (\*) eklenirse, istemci kuyruk yöneticisini bağlar.

Benzer şekilde, bir girişteki *ibm-amqQueueManagerName* özniteliği tanımsız bırakılabilir. Bu durumda, bu uç nokta bilgilerini kullanan istemcinin herhangi bir kuyruk yöneticisine bağlanabilmesi 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, tanımlı olmayan bir öznitelik değerinde arama yapılmaz. Örneğin, bir giriş, *ibm-amqQueueManagerName* dışındaki bağlantı bilgilerini içeriyorsa, arama sonuçları bu girişi içermeyecekti. Bu sorunu aşmak için *ibm-amqIsClientDefault* öğesini ayarlayabilirsiniz. Bu bir Boole öznitelidir ve tanımlanmamışsa, FALSE değerinin olduğu varsayılır.

*ibm-amqQueueManagerName* ' in tanımlanmadığı ve aramanın bir parçası olması beklenen girişler için, *ibm-amqIsClientDefault* değerini TRUE olarak ayarlayın. MQCONN/X çağrısında kuyruk yöneticisi adı olarak boş ya da yıldız işareti (\*) belirtildiğinde, ön bağlanma çıkışı, *ibm-amqIsClientDefault* öznitelik değerinin TRUE olarak ayarlandığı tüm girişler için LDAP sunucusunu arar.

**Not:** *ibm-amqIsClientDefault* TRUE olarak ayarlandıysa, *ibm-amqQueueManagerName* özniteliğini ayarlamaz ya da tanımlamayın.

### İlgili başvurular

[“ibm-amqQueueManagerName” sayfa 1189](#)

Bu öznitelik, bir IBM MQ istemcisi uygulamasının bağlantı isteyebileceği kuyruk yöneticisi ya da kuyruk yöneticisi grubunun adını belirtir.

ULW

*ibm-amqHeaderSıkıştırma*

Bu LDAP özniteliği, kanal tarafından desteklenen üstbilgi veri sıkıştırma tekniklerinin bir listesidir.

Bu özniteliğin büyüklük üst sınırı 48 karakterdir. Bu, önceden var olan bir öznitelik değildir.

Bu öznitelik için yalnızca bir değer belirtebilirsiniz.

Bu liste özniteliği, virgülle ayrılmış bir biçim kullanılarak dizin dizgileri olarak belirtilir. Örneğin, **ibm-amqHeaderCompression** için belirtilen değer 0 , bu değer NONE olarak eşlenir. İstemci tarafından izin verilen üst sınırı aşan değerler dikkate alınmaz. Örneğin, *ibm-amqHeaderCompression*, listede en çok 2 tamsayı içerir.

ULW

*ibm-amqMessageSıkıştırma*

Bu LDAP özniteliği, kanal tarafından desteklenen ileti veri sıkıştırma tekniklerinin bir listesidir.

Bu özniteliğin büyüklük üst sınırı 48 karakterdir. Bu, önceden var olan bir öznitelik değildir.

Bu öznitelik birden çok değeri desteklemiyor.

Bu liste özniteliği, virgülle ayrılmış bir biçim kullanılarak dizin dizgileri olarak belirtilir. Örneğin, bu öznitelik için belirtilen değer 1,2,4 'tür ve bu, temeldeki sıkıştırma sırası RLE, ZLIBFAST ve ZLIBSTHH ile eşlenir.

İstemci tarafından izin verilen üst sınırı aşan değerler dikkate alınmaz. Örneğin, `ibm-amqMessageSıkıştırması`, listede en çok 16 tamsayı içerir.

#### **ULW** `ibm-amqSendExitUserVerileri`

Bu LDAP özniteliği, gönderme çıkışa geçirilen kullanıcı verilerini belirtir.

Bu LDAP özniteliğinin, büyük ve küçük harfe duyarlı olmayan tek bir dizgi değeri en fazla 999 karakter içerir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

**Not:** `ibm-amqSendExitName` ve `ibm-amqSendExitUserData` ' in çift olarak eşitlenmesi gerekir. Kullanıcı verileri, çıkış adıyla uyumlulaştırılmalıdır. Bu nedenle, biri belirtilirse, diğeri veri içermese de, diğeryandan da simetrik olarak belirtilmiş olmalıdır.

#### **ULW** `ibm-amqSendExitName`

Bu LDAP özniteliği, kanal gönderme çıkışıyla çalıştırılacak çıkış programının adını belirtir.

Bu öznitelik, büyük/küçük harf duyarlı olmayan, en çok 999 karakterden oluşan tek bir dizgi değerine sahiptir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

**Not:** `ibm-amqSendExitName` ve `ibm-amqSendExitUserData` , çiftler halinde eşitlenmiş olmalıdır. Kullanıcı verileri, çıkış adıyla eşitlenmiş olmalıdır. Bu nedenle, biri belirtilirse, diğeri veri içermese de simetrik olarak belirtilmiş olmalıdır.

#### **ULW** `ibm-amqReceiveExitUserVerileri`

Bu LDAP özniteliği, alma çıkışa geçirilen kullanıcı verilerini belirtir.

Bir dizi alma çıkışı çalıştırabilirsiniz. Bir dizi çıkışa ilişkin kullanıcı verileri dizgisi virgülle, boşluklarla ya da her ikisiyle birbirinden ayrılır.

Bu öznitelik, büyük/küçük harf duyarlı olmayan, en çok 999 karakterden oluşan tek bir dizgi değerine sahiptir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.

**Not:** `ibm-amqReceiveExitName` ve `ibm-amqReceiveExitUserData` , çiftler halinde eşitlenmiş olmalıdır. Kullanıcı verileri, çıkış adıyla eşitlenmiş olmalıdır. Bu nedenle, biri belirtilirse, diğeri veri içermese de simetrik olarak belirtilmiş olmalıdır.

#### **ULW** `ibm-amqReceiveExitName`

Bu LDAP özniteliği, kanal tarafından çalıştırılacak kullanıcı çıkış programının adını belirtir.

Bu öznitelik, art arda çalıştırılacak programların adlarının bir listesidir. Herhangi bir kanal alma kullanıcı çıkışı yürürlükte değilse, boş bırakın.

Bu öznitelik, büyük/küçük harf duyarlı olmayan, en çok 999 karakterden oluşan tek bir dizgi değerine sahiptir. Bu, önceden var olan bir öznitelik değildir.

Alt dizgi eşleştirme yoksayıldı. Alt dizgi eşleştirme, bir arama süzgecindeki özniteliğin davranışını belirleyen, alt şemada kullanılan bir eşleşen kuraldır.



**Not:** **ibm-amqReceiveExitName** ve **ibm-amqReceiveExitUserData** , çiftler halinde eşitlenmiş olmalıdır. Kullanıcı verileri, çıkış adıyla eşitlenmiş olmalıdır. Bu nedenle, biri belirtilirse, diğeri veri içermese de, simetrik olarak da belirtilmiş olmalıdır.

## z/OS için örnek programların kullanılması

IBM MQ for z/OS ile teslim edilen örnek yordamsal uygulamalar, İleti Kuyruğu Arabirimi 'nin (MQI) tipik kullanımları göstermektedir.

### Bu görev hakkında

IBM MQ for z/OS also provides sample data-conversion exits, described in [“Veri dönüştürme çıkışları yazılıyor” sayfa 1014.](#)

Örnek uygulamaların tümü kaynak biçiminde sağlanır; birden çok uygulama yürütülebilir biçimde de sağlanır. Kaynak modüller, program mantığını açıklayan sözde kodu içerir.

**Not:** Bazı örnek uygulamaların temel pano tabanlı arabirimleri olsa da, uygulamalarının görünüşünün ve hislerinin nasıl tasarlanması gerektiğini göstermeyi amaçlamazlar. Programlanabilir olmayan uçbirimler için pano tabanlı arabirimlerin nasıl tasarlanabileceği hakkında daha fazla bilgi için *SAA Common User Access: Basic Interface Design Guide* (SC26-4583) belgesine ve eki (GG22-9508) belgesine bakın. Bu bilgiler, uygulama içinde ve diğer uygulamalar arasında tutarlı olan uygulamaları tasarlamaya yardımcı olacak yönergeler sağlar.

### Yordam

- Örnek programlar hakkında daha fazla bilgi almak için aşağıdaki bağlantıları kullanın:
  - [“z/OS için örnek uygulamalarda gösterilen özellikler” sayfa 1194](#)
  - [“z/OS üzerinde toplu iş ortamı için örnek uygulamalar hazırlama ve çalıştırma” sayfa 1200](#)
  - [“z/OS üzerinde TSO ortamı için örnek uygulamalar hazırlama” sayfa 1203](#)
  - [“Preparing the sample applications for the CICS environment on z/OS” sayfa 1205](#)
  - [“Preparing the sample application for the IMS environment on z/OS” sayfa 1208](#)
  - [“The Put samples on z/OS” sayfa 1209](#)
  - [“The Get samples on z/OS” sayfa 1211](#)
  - [“The Browse sample on z/OS” sayfa 1214](#)
  - [“z/OS üzerindeki Print Message \(Yazdırma İletisi\) örneği” sayfa 1216](#)
  - [“z/OS üzerinde Kuyruk Öznitelikleri örneği” sayfa 1219](#)
  - [“The Mail Manager sample on z/OS” sayfa 1220](#)
  - [“The Credit Check sample on z/OS” sayfa 1227](#)
  - [“The Message Handler sample on z/OS” sayfa 1237](#)
  - [“z/OS üzerinde Zamanuyumsuz Koyma örneği” sayfa 1241](#)
  - [“The Batch Asynchronous Consumption sample on z/OS” sayfa 1242](#)
  - [“z/OS üzerinde CICS Zamanuyumsuz Tüketimi ve Yayınlama/Abone Olma örneği” sayfa 1244](#)
  - [“z/OS üzerindeki Yayınlama/Abone Olma örneği” sayfa 1246](#)
  - [“z/OS üzerinde Set ve Sorgula ileti özelliği örneği” sayfa 1248](#)

### İlgili görevler

[“Örnek Programların çoklu Platformlar Üzerinde Kullanılması” sayfa 1093](#)

Bu örnek yordamsal programlar ürünle birlikte teslim edilir. Örnekler, C ve COBOL 'de yazılır ve Message Queue Interface (MQI)' in tipik kullanımları gösterir.

## **z/OS** z/OS için örnek uygulamalarda gösterilen özellikler

Bu bölümde, örnek uygulamaların her birinde gösterilen MQI özellikleri özetlenmektedir, her bir örnek yazıldığı programlama dilleri ve her bir örnek çalıştırıldığı ortam özetlenir.

### **z/OS** Örnekleri z/OS üzerine koy

Put örnekleri, MQPUT çağrısını kullanarak iletilerin kuyruğa nasıl konacağını gösterir.

Uygulama bu MQI çağrılarını kullanıyor:

- MQCONN
- MQOPEN
- MQPUT
- MQCLOSE
- MQDISC

Program COBOL ve C ' de teslim edilir ve toplu iş ve CICS ortamında çalışır. Toplu iş uygulaması için bkz. [Çizelge 179 sayfa 1201](#) ve CICS uygulaması için [Çizelge 186 sayfa 1206](#) .

### **z/OS** Get samples on z/OS

Alma örnekleri, MQGET çağrısını kullanarak bir kuyruktan iletilerin nasıl gönderileceğini gösterir.

Uygulama bu MQI çağrılarını kullanıyor:

- MQCONN
- MQOPEN
- MQGet
- MQCLOSE
- MQDISC

Program COBOL ve C ' de teslim edilir ve toplu iş ve CICS ortamında çalışır. Toplu iş uygulaması için bkz. [Çizelge 179 sayfa 1201](#) ve CICS uygulaması için [Çizelge 186 sayfa 1206](#) .

### **z/OS** z/OS' ta göz atma örneği

Göz At (Browse) örneği, bir iletiyi bulmak için Göz At seçeneğinin nasıl kullanılacağını gösterir, yazdırın ve kuyruklardaki iletileri adım adım ilerler.

Uygulama bu MQI çağrılarını kullanıyor:

- MQCONN
- MQOPEN
- İletilere göz atmak için MQGET
- MQCLOSE
- MQDISC

Program COBOL, çevirici, PL/I ve C dillerinde teslim edilir. Uygulama, toplu iş ortamında çalışır. Toplu iş uygulaması için bkz. [Çizelge 180 sayfa 1201](#) .

### **z/OS** Print Message Sample on z/OS

Print Message Sample, bir iletinin kuyruktan nasıl kaldırılacağı ve iletinin, ileti tanımlayıcısının tüm alanlarıyla birlikte iletiyle nasıl yazdırılacağı gösterilir. İsteğe bağlı olarak, her iletiyle ilişkili tüm ileti özelliklerinin tümünü görüntüleyebilir.

Kaynak modüldeki iki satırdan açıklama karakterlerini kaldırarak, programı, bir kuyrukta bulunan iletileri kaldırmak yerine göz atmasını sağlamak üzere değiştirebilirsiniz. Bu program, bir kuyruğa ileti yerleştiren bir uygulamayla ilgili sorunların tanınması için kullanılabilir.

Uygulama bu MQI çağrılarını kullanıyor:

- MQCONN
- MQOPEN
- İletilerin kuyruktan kaldırılması için MQGET (göz atma seçeneği ile)
- MQCLOSE
- MQDISC
- MQCRTMH
- MQDLTMH
- MQINQMP

Program C dilinde teslim edilir. Uygulama, toplu iş ortamında çalışır. Toplu iş uygulaması için bkz. [Çizelge 181 sayfa 1202](#) .

#### z/OS üzerinde Kuyruk Öznitelikleri örneği

Kuyruk Öznitelikleri örneği, IBM MQ for z/OS nesne özniteliklerinin değerlerinin nasıl sorgulanmasını ve ayarlanmasını gösterir.

Uygulama bu MQI çağrılarını kullanıyor:

- MQOPEN
- MQINQ
- MQSET
- MQCLOSE

Program COBOL, çevirici ve C dillerinde teslim edilir. Uygulama, CICS ortamında çalışır. CICS uygulaması için bkz. [Çizelge 187 sayfa 1206](#) .

#### Mail Manager sample on z/OS

Mail Manager örneği kullanılırken dikkat edilmesi gereken noktalar.

Mail Manager örneği şu teknikleri gösterir:

- Diğer ad kuyruklarının kullanılması
- Geçici dinamik kuyruk yaratmak için bir model kuyruğu kullanılması
- Yanıtlama kuyruklarının kullanılması
- CICS ve toplu iş ortamlarında eşitleme noktalarının kullanılması
- Sistem komutu giriş kuyruğuna komut gönderilmesi
- Dönüş kodlarının sınanması
- Uzak kuyruk yöneticilerine, uzak kuyruğun yerel tanımlamasını kullanarak ve iletileri uzak kuyruk yöneticisinde doğrudan adlandırılan bir kuyruğa koyarak, iletileri uzak kuyruk yöneticilerine gönderme

Uygulama bu MQI çağrılarını kullanıyor:

- MQCONN
- MQOPEN
- MQPUT1
- MQGet
- MQINQ
- MQCMIT
- MQCLOSE
- MQDISC

Uygulamanın üç sürümü sağlanır:

- COBOL ' da yazılmış bir CICS uygulaması
- COBOL ' da yazılmış bir TSO uygulaması
- C içinde 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 184 sayfa 1203](#) ve CICS uygulaması için [Çizelge 188 sayfa 1206](#) .

#### *Credit Check sample on z/OS*

Bu bilgiler, Credit Check Sample uygulamasını kullanırken dikkate alınacak noktaları içerir.

Credit Check Sample, bu teknikleri sergileyen bir program grubudur:

- Birden çok ortamda çalışan bir uygulama geliştirilmesi
- Geçici dinamik kuyruk yaratmak için bir model kuyruğu kullanılması
- İlti tanıtıcısı kullanılması
- Bağlam bilgilerinin ayarlanması ve geçirilmesi
- İleti önceliğinin ve kalıcılarının kullanılması
- Programları tetiklemeyi kullanarak başlatma
- Yanıtlama kuyruklarının kullanılması
- Diğer ad kuyruklarının kullanılması
- Kuyruk-harf kuyrukunun kullanılması
- Ad listesi kullanılması
- Dönüş kodlarının sınanması

Uygulama bu MQI çağrılarını kullanıyor:

- MQOPEN
- MQPUT
- MQPUT1
- Tarama ve ileti alma, bekleme ve sinyal seçeneklerini kullanma ve belirli bir iletiyi alma için MQGET
- MQINQ
- MQSET
- MQCLOSE

Örnek, bağımsız bir CICS uygulaması olarak çalıştırılabilir. Ancak, hem CICS hem de IMS ortamlarında sağlanan olanakları kullanan bir ileti kuyruklama uygulamasının nasıl tasarlanması olduğunu göstermek için, bir modül de IMS toplu ileti işleme programı olarak sağlanır.

CICS programları C ve COBOL ' de teslim edilir. Tek IMS programı C içinde teslim edilir.

CICS uygulaması için bkz. [Çizelge 189 sayfa 1207](#) , IMS uygulaması için [Çizelge 191 sayfa 1209](#) .

#### *The Message Handler sample on z/OS*

İleti İşleyici örneği, bir kuyruktaki iletilere göz atmanızı, iletileri iletmenizi ve silmenizi sağlar.

Uygulama bu MQI çağrılarını kullanıyor:

- MQCONN
- MQOPEN
- MQINQ
- MQPUT1
- MQCMIT
- MQBACK

- MQGet
- MQCLOSE
- MQDISC

Program C ve COBOL programlama dillerinde teslim edilir. Uygulama TSO altında çalışır. TSO uygulaması için bkz. Çizelge 185 sayfa 1204.

**z/OS** z/OS üzerinde kuyruğa alma çıkış örnekleri dağıtıldı

Dağıtılmış kuyruğa alma çıkış örneklerinin kaynak programları tablosu.

Dağıtılan kuyruğa alma çıkış örneklerinin kaynak programlarının adları aşağıdaki tabloda listelenir:

<i>Çizelge 177. Dağıtılmış kuyruğa alma çıkış örneklerinin kaynağı</i>			
Üye adı	Dil için	Tanım	Kitaplıkta sağlanan
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-düzenlenmesine neden olur.

**z/OS** z/OS üzerindeki veri dönüştürme çıkış örnekleri

Bir veri dönüştürme çıkış yordamı için bir iskelet sağlanır ve MQXCNVÇ çağrısını gösteren bir örnek IBM MQ ile birlikte gönderilir.

Veri dönüştürme çıkış örneklerinin kaynak programlarının adları aşağıdaki çizelgede listelenir:

<i>Çizelge 178. Veri dönüştürme çıkış örneklerinin kaynağı (yalnızca çevirici dili)</i>		
Üye adı	Tanım	Kitaplıkta sağlanan
CSQ4BAX8	Kaynak program	SCSQASMS
CSQ4BAX9	Kaynak program	SCSQASMS
CSQ4CAX9	Kaynak program	SCSQASMS

**Not:** Kaynak programlar, CSQASKICE ile bağlantı-düzenlenmektedir.

Ek bilgi için “Veri dönüştürme çıkışları yazılıyor” sayfa 1014 başlıklı konuya bakın.

**z/OS** z/OS üzerinde Yayınlama/Abone Olma örnekleri

Yayınlama/Abone Olma örnek programları, IBM MQ' ta yayınlama ve abone olma özelliklerinin kullanımını gösterir.

IBM MQ Yayınlama/Abone Olma arabirimine nasıl programların göstermeyi gösteren dört adet C ve iki COBOL programlama dili örnek programı vardır.

Uygulamalar bu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQPUT
- MQSUB
- MQGet
- MQCLOSE

- MQDISC
- MQCRTMH
- MQDLTMH
- MQINQMP

Public/Subscribe örnek programları C ve COBOL programlama dillerinde teslim edilir. Örnek uygulamalar toplu iş ortamında çalışır. Toplu iş uygulamaları için bkz. [Örnekle/Abone Olma örnekleri](#).

## **z/OS** **z/OS üzerindeki istemci bağlantılarını kabul etmek için 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 etmek için yapılandırabilirsiniz.

### **Başlamadan önce**

Kuyruk yöneticisinin zaten var olduğundan ve başlatıldığından emin olun. Kanal doğrulama kayıtlarının, MQSC komutu vererek önceden etkinleştirilmiş olup olmadığını saptayın:

```
DISPLAY QMGR CHLAUTH
```

**Önemli:** Bu görev, kanal doğrulama kayıtlarının etkinleştirilmesini bekler. Bu, diğer kullanıcılar ve uygulamalar tarafından kullanılan bir kuyruk yöneticisiyse, bu ayarın değiştirilmesi diğer tüm kullanıcıları ve uygulamaları etkiler. If your queue manager does not make use of channel authentication records then step 4 can be replaced with an alternate authentication method (for example a security exit) which sets the MCAUSER to the *ayrıcalıksız-kullanıcı kimliği* you will obtain in step "1" sayfa 1198.

Uygulamanızın hangi kanal adını kullanmayı beklediğini bilmeniz gerekir. Böylece, uygulamanın kanalı kullanmasına izin verilmelidir. Ayrıca, hangi nesnelere, örneğin kuyruklar ya da konular için uygulamanızın kullanılmasını beklediğinden, uygulamanızın bunları kullanmasına izin verilebilmesi için izin verileceğini de bilmeniz gerekir.

### **Bu görev hakkında**

Bu görev, kuyruk yöneticisine bağlanan bir istemci uygulaması için kullanılmak üzere ayrıcalıklı olmayan bir kullanıcı kimliği yaratır. İstemci uygulaması için erişim izni verilir; yalnızca, gereken kanalı ve bu kullanıcı kimliğini kullanarak gereksinim duyduğu kuyruğu kullanabilmelidir.

### **Yordam**

1. Kuyruk yöneticinizin çalışmakta olduğu sistemde bir kullanıcı kimliği edinin.

Bu görev için bu kullanıcı kimliği ayrıcalıklı bir yönetimle görevli kullanıcı olmamalıdır. Bu kullanıcı kimliği, istemci bağlantısının kuyruk yöneticisi üzerinde çalışacağı yetkindir.

2. Bir dinleyici programı başlatın.

- a) Kanal başlatıcınızın başlatıldığından emin olun. Yoksa, **START CHINIT** komutunu vererek başlatın.
- b) Aşağıdaki komutu girerek dinleyici programını başlatın:

```
START LISTENER TRPTYPE(TCP) PORT(nnnn)
```

Burada *nnnn* , seçtiğiniz kapı numarasıdır.

3. Uygulamanızın SYSTEM.DEF.SVRCONN daha sonra bu kanal önceden tanımlıdır. Uygulamanız başka bir kanal kullanıyorsa, MQSC komutunu kullanarak bu kanalı yaratın:

```
DEFINE CHANNEL(' channel-name ') CHLTYPE(SVRCONN) TRPTYPE(TCP) +  
DESCR('Channel for use by sample programs')
```

*kanal-adi* , kanalınızın adıdır.

4. İstemci sisteminizin yalnızca IP adresinin, bu kanalı kullanarak, MQSC komutunu vererek kanal kullanmasını sağlayacak bir kanal doğrulama kuralı yaratın:

```
SET CHLAUTH(' channel-name ') TYPE(ADDRESSMAP) ADDRESS(' client-machine-IP-address ') +
MCAUSER(' non-privileged-user-id ')
```

burada:

*kanal-adi* , kanalınızın adıdır.

*client-machine-IP-address* , istemci sisteminizin IP adresidir. Örnek istemci uygulamanız kuyruk yöneticisiyle aynı makinede çalıştırılıyorsa, uygulamanız 'localhost' ile bağlantı kuracaksa '127.0.0.1' IP adresini kullanın. Birden çok farklı istemci makinesi bağlanacaksa, tek bir IP adresi yerine bir kalıp ya da aralık kullanabilirsiniz. Ayrıntılı bilgi için [Soysal IP adresleri](#) başlıklı konuya bakın. *ayrıcılıkli olmayan-kullanıcı-kimliği* , "1" sayfa 1198adımında edindiğiniz kullanıcı kimliğidir.

5. Uygulamanız SYSTEM.DEFAULT.LOCAL.QUEUE(Kuyruk), daha sonra bu kuyruk zaten tanımlı. Uygulamanız başka bir kuyruk kullanıyorsa, MQSC komutunu vererek yaratın:

```
DEFINE QLOCAL(' queue-name ') DESCR('Queue for use by sample programs')
```

Burada *kuyruk-adi* , kuyruğunuzun adıdır.

6. Kuyruk yöneticisine bağlanmak ve sorgulamak için erişim izni verin:

- Kanal başlatıcınızın başlatıldığından emin olun. Yoksa, START CHINIT komutunu kullanarak kanal başlatıcıyı başlatın.
- 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, bu, kuyrukların kullanılmasını sağlar ve MQSC komutları vererek, kullanılacak kullanıcı kimliği tarafından kuyruğunuzu kullanarak kuyruğa alma ve iletileri alma ve iletileri alma izni 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-adi* , kuyruğunuzun adıdır.

*ayrıcılıkli olmayan-kullanıcı-kimliği* , "1" sayfa 1198adımında edindiğiniz kullanıcı kimliğidir.

8. Uygulamanız bir yayınlama/abone olma uygulamasıysa, konu kullanımı, aşağıdaki RACF komutlarını yayınlayarak, kullanılacak kullanıcı kimliği ile konularınızı kullanarak yayınlamayı 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

*ayrıcalklı olmayan-kullanıcı-kimliği* , “1” sayfa 1198adımında edindiğiniz kullanıcı kimliğidir.

Bu işlem, konu ağacındaki herhangi bir konuya *ayrıcalklısuz-kullanıcı-kimliği* erişimi verir; diğere bir seçenek olarak, **DEFINE TOPIC** kullanarak bir konu nesnesi tanımlayabilir ve yalnızca o konu nesnesinin gönderme yaptığı konu ağacının bir bölümüne erişimler verir. Daha fazla bilgi için [Konulara kullanıcı erişimini denetleme](#) başlıklı konuya bakın.

## Sonraki adım

Artık istemci uygulamanız kuyruk yöneticisine bağlanabilir ve kuyruğu kullanarak ileti alabilir ya da alabilir.

### İlgili kavramlar

**z/OS** [z/OS üzerinde IBM MQ nesnelere çalışma yetkisi](#)

### İlgili başvurular

[CHLAUTH KÜMESİ](#)

[KANAL TANIMLA](#)

[QLOCAL ' I TANIMLA](#)

[AUTHREC](#)

### **z/OS** [z/OS üzerinde toplu iş ortamı için örnek uygulamalar hazırlama ve çalıştırma](#)

Toplu iş ortamında çalışan bir örnek uygulama hazırlamak için, herhangi bir toplu iş IBM MQ for z/OS uygulamasını oluştururken uyguladığınız adımları gerçekleştirin.

Bu adımlar “z/OS toplu iş uygulamaları oluşturma” sayfa 1060 içinde listelenir.

Diğere bir seçenek olarak, bir örnek için yürütülebilir bir form sağladığımız durumlarda, bunu [thlqual.SCSQLOAD](#) yükleme kitaplığından çalıştırabilirsiniz.

**Not:** Göz At örneğinin çevirici dili sürümü, veri denetim bloklarını (DCBs) kullanır, bu nedenle RMODE (24) komutunu kullanarak bağlantıyı düzenlemeniz gerekir.

Kullanılacak kitaplık üyeleri [Çizelge 179 sayfa 1201](#), [Çizelge 180 sayfa 1201](#), [Çizelge 181 sayfa 1202](#) ve [Çizelge 182 sayfa 1202](#) ' de listelenir.

Kullanmak istediğiniz örnekler için verilen çalıştırma JCL ' yi düzenlemeniz gerekir (bkz. [Çizelge 179 sayfa 1201](#), [Çizelge 180 sayfa 1201](#), [Çizelge 181 sayfa 1202](#) ve [Çizelge 182 sayfa 1202](#) ).

Belirtilen JCL ' deki PARM deyimi, değiştirmeniz gereken bazı değiştiregeleri içerir. C örnek programlarını çalıştırmak için, parametreleri boşluklara göre 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 ' de LOCALQ1 adlı bir kuyrukla çalıştırmak istiyorsanız, PARM deyiminiz şu şekilde görünmelidir:

```
PARM=(CSQ1, LOCALQ1)
```

C dili JCL ' de, PARM deyiminiz şöyle görünmelidir:

```
PARM=(' CSQ1 LOCALQ1 ')
```

Şimdi işleri teslim etmeye hazırsınız.

### **z/OS** [z/OS üzerindeki örnek toplu iş uygulamalarının adları](#)

Örnek toplu iş uygulamaları için sağlanan programların bir özeti.

Toplu iş uygulama programları aşağıdaki tablolarda özetlenir:

- [Çizelge 179 sayfa 1201](#) Put ve Get örnekleri
- [Çizelge 180 sayfa 1201](#) Göz at örneği



- Çizelge 181 sayfa 1202 Yazdırma iletisi örneği
- Çizelge 182 sayfa 1202 Yayınlama/Abone Olma örnekleri
- Çizelge 183 sayfa 1202 Diğer örnekler

<i>Çizelge 179. Toplu Put ve Alma örnekleri</i>				
Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4BCJ1	C	Kaynak programı al	SCSQC37S	SCSQLOAD
CSQ4BCK1	C	Kaynak program koy	SCSQC37S	SCSQLOAD
CSQ4BCJR	C	Sample run JCL for CSQ4BCJ1 and CSQBCK1	SCSQPROC	Yok
CSQ4BVJ1	COBOL	Kaynak programı al	SCSQCOBS	SCSQLOAD
CSQ4BVK1	COBOL	Kaynak program koy	SCSQCOBS	SCSQLOAD
CSQ4BVJR	COBOL	Sample run JCL for CSQBVJ1 and CSQBVK1	SCSQPROC	Yok

<i>Çizelge 180. Toplu Göz At örneği</i>				
Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	Kitaplıkta sağlanan yürütülebilir 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 181. Toplu Yazdırma İletisi örneği (Yalnızca C dili)

Üye adı	Tanım	Kaynak dosya kitaplıkta sağlandı	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4BCG1	Kaynak program	SCSQC37S	SCSQLOAD
CSQ4BCGR	CSQ4BCG1 için örnek çalıştırma JCL	SCSQPROC	Yok
CSQ4BCL1	Kaynak programa göz at	SCSQC37S	SCSQLOAD
CSQ4BCLR	CSQ4BCL1 için örnek çalıştırma JCL	SCSQPROC	Yok

Çizelge 182. Yayınlama/Abone Olma örnekleri

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	SCSQPROC ' de JCL	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4BCP1	C	Konu kaynak programına yayınla	SCSQC37S	CSQ4BCPP	SCSQLOAD
CSQ4BCP2	C	Konuya abone ol ve ileti kaynağı programını al	SCSQC37S	CSQ4BCPS	SCSQLOAD
CSQ4BCP3	C	Kullanıcı tarafından sağlanan bir hedefi kullanarak konuya abone olun ve ileti kaynağı programını alın	SCSQC37S	CSQ4BCPD	SCSQLOAD
CSQ4BCP4	C	Ek seçenekleri kullanarak konuya abone olun ve ileti kaynağı programını alın	SCSQC37S	CSQ4BCPE	SCSQLOAD
CSQ4BVP1	COBOL	Konu kaynak programına yayınla	SCSQC OBS	CSQ4BVPP	SCSQLOAD
CSQ4BVP2	COBOL	Konuya abone ol ve ileti kaynağı programını al	SCSQC OBS	CSQ4BVPS	SCSQLOAD

Çizelge 183. Diğer örnekler

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	SCSQPROC ' de JCL	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4BCS1	C	Zamanuyumsuz tüketim kaynağı programı	SCSQC37S	CSQ4BCSC	SCSQLOAD

Çizelge 183. Diğer örnekler (devamı var)

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	SCSQPROC ' de JCL	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4BCS2	C	Zamanuyumsuz Koma ve Denetim durumu kaynak programı	SCSQC37S	CSQ4BCSP	SCSQLOAD
CSQ4BCM1	C	İleti özellikleri kaynak programı	SCSQC37S	CSQ4BCMP	SCSQLOAD
CSQ4BCM2	C	İleti özellikleri kaynak programını ayarla	SCSQC37S	CSQ4BCMP	SCSQLOAD

### z/OS z/OSüzerinde TSO ortamı için örnek uygulamalar hazırlama

TSO ortamında çalışan bir örnek uygulama hazırlamak için, herhangi bir toplu iş IBM MQ for z/OS uygulamasını oluştururken uyguladığınız adımları gerçekleştirin.

Bu adımlar “z/OS toplu iş uygulamaları oluşturma” sayfa 1060’inde listelenir. Kullanılacak kitaplık üyeleri Çizelge 184 sayfa 1203’te listesinde yer alıyor.

Diğer bir seçenek olarak, bir örnek için yürütülebilir bir form sağladığımız durumlarda, bunu thlqual.SCSQLOAD yükleme kitaplığından çalıştırabilirsiniz.

Posta Yöneticisi örnek uygulaması için, kullandığı kuyrukların sisteminizde var olduğundan emin olun. Bunlar thlqual.SCSQPROC (CSQ4CVD) üyesinde tanımlanmıştır. Bu kuyrukların her zaman kullanılabilir olmasını sağlamak için, bu üyeleri CSQINP2 kullanıma hazırlama giriş veri kümenize ekleyebilir ya da bu kuyruk tanımlamalarını yüklemek için CSQUTIL programını kullanabilirsiniz.

### z/OS z/OSüzerindeki örnek TSO uygulamalarının adları

Örnek TSO uygulamalarının her biri için sağlanan programların ve kaynağın, JCL ' nin ve İleti İşleyici örneğinin yalnızca, yürütülebilir dosyaların bulunduğu kitaplıkların adlarıyla ilgili bilgiler.

TSO uygulama programları aşağıdaki tablolarda özetlenmiştir:

- Çizelge 184 sayfa 1203 Mail Manager örneği
- Çizelge 185 sayfa 1204 Message işleyicisi örneği

Bu örnekler ISPF panolarını kullanır. Bu nedenle, programları bağlarken ISPF sınırlı kod öbeğini, ISPLINK ' ı da eklemelisiniz.

Çizelge 184. TSO Mail Manager örneği

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı
CSQ4CVD	Bağımsız	IBM MQ for z/OS nesne tanımları	SCSQPROC
CSQ40	Bağımsız	ISPF iletileri	SCSQMSGE
CSQ4RVD1	COBOL	CSQ4TVD1başlatmak için CTLIST	SCSQCLST
CSQ4TVD1	COBOL	Menü programı için kaynak program	SCSQCOBS
CSQ4TVD2	COBOL	Posta almak için kaynak program	SCSQCOBS

Çizelge 184. TSO Mail Manager örneği (devamı var)

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı
CSQ4TVD4	COBOL	Posta Gönderme programı için kaynak program	SCSQCOBS
CSQ4TVD5	COBOL	Takma ad programı için kaynak program	SCSQCOBS
CSQ4VDP1-6	COBOL	Pano tanımları	SCSQPNLA
CSQ4VD0	COBOL	Veri tanımı	SCSQCOBC
CSQ4VD1	COBOL	Veri tanımı	SCSQCOBC
CSQ4VD2	COBOL	Veri tanımı	SCSQCOBC
CSQ4VD4	COBOL	Veri tanımı	SCSQCOBC
CSQ4RCD1	C	CSQ4TCD1' i başlatmak için CTLIST	SCSQCLST
CSQ4TCD1	C	Menü programı için kaynak program	SCSQ37S
CSQ4TCD2	C	Posta almak için kaynak program	SCSQ37S
CSQ4TCD4	C	Posta Gönderme programı için kaynak program	SCSQ37S
CSQ4TCD5	C	Takma ad programı için kaynak program	SCSQ37S
CSQ4CDP1-6	C	Pano tanımları	SCSQPNLA
CSQ4TC0	C	İçerme dosyası	SCSQ370

Çizelge 185. TSO İleti İşleyicisi örneği

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4TCH0	C	Veri tanımı	SCSQ370	Yok
CSQ4TCH1	C	Kaynak program	SCSQ37S	SCSQLOAD
CSQ4TCH2	C	Kaynak program	SCSQ37S	SCSQLOAD
CSQ4TCH3	C	Kaynak program	SCSQ37S	SCSQLOAD
CSQ4RCH1	C ve COBOL	CLIST to initiate CSQ4TCH1 or CSQ4TVH1	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

Çizelge 185. TSO İleti İşleyicisi örneği (devamı var)

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4CHP9	C ve COBOL	Pano tanımı	SCSQPNLA	Yok
CSQ4TVH0	COBOL	Veri tanımı	SCSQCOBC	Yok
CSQ4TVH1	COBOL	Kaynak program	SCSQCOBS	SCSQLOAD
CSQ4TVH2	COBOL	Kaynak program	SCSQCOBS	SCSQLOAD
CSQ4TVH3	COBOL	Kaynak program	SCSQCOBS	SCSQLOAD

### z/OS Preparing the sample applications for the CICS environment on z/OS

CICS örnek programlarını çalıştırmadan önce, 32702 LOGMODE komutunu kullanarak CICS ' ta oturum açın. Bunun nedeni, örnek programların bir 3270 kip 2 ekranını kullanmak için yazıldığı içindir.

CICS ortamında çalışan bir örnek uygulama hazırlamak için aşağıdaki adımları gerçekleştirin:

1. BMS ekran tanımlaması kaynağını (kitaplık **thlqual.SCSQMAPS** içinde sağlanan) BMS ekran tanımı kaynağını (burada **thlqual** , kuruluşunuz tarafından kullanılan üst düzey niteleyicidir) bir araya getirerek, simgesel tanımlama eşlemeni ve fiziksel ekran eşlemeni yaratın. Haritalar adını verdiğinizde, BMS ekran tanımlaması kaynağının adını kullanın (örneğin, örnek programları koymak ve almak için kullanılamaz), ancak bu adın son karakterini kaldırın.
2. Herhangi bir CICS IBM MQ for z/OS uygulamasını oluştururken uyguladığınız adımları gerçekleştirin. Bu adımlar “z/OS’inde CICS uygulamaları oluşturma” sayfa 1063’inde listelenir. Kullanılacak kitaplık üyeleri Çizelge 186 sayfa 1206, Çizelge 187 sayfa 1206, Çizelge 188 sayfa 1206ve Çizelge 189 sayfa 1207’ de listelenir.

Diğer bir seçenek olarak, bir örnek için yürütülebilir bir form sağladığımız durumlarda, bunu **thlqual.SCSQCICS** yükleme kitaplığından çalıştırabilirsiniz.

3. Identify the map set, programs, and transaction to CICS by updating the CICS system definition (CSD) data set. Gereksinim duyduğunuz tanımlamalar **thlqual.SCSQPROC** (CSQ4S100) üyesinde yer alıyor. Bunu nasıl yapacağını ilişkin yönergeler için, CICS Transaction Server for z/OS 4.1 ürün belgelerindeki *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, örnek tarafından kullanılan VSAM veri kümesini önceden yaratmadıysanız, bu aşamada bir hata iletisi alabilirsiniz.

4. Credit Check ve Mail Manager örnek uygulamaları için, kullandığınız kuyrukların sisteminizde kullanılabilir olduğundan emin olun. Credit Check örneği için, COBOL için **thlqual.SCSQPROC** (CSQ4CVB) üyesinde ve C için **thlqual.SCSQPROC** (CSQ4CCB) adlı üyelerde tanımlanmışlar. Mail Manager örneği için, bunlar **thlqual.SCSQPROC** (CSQ4CVD) üyesinde tanımlanmıştır. Bu kuyrukların her zaman kullanılabilir olmasını sağlamak için, bu üyeleri CSQINP2 kullanıma hazırlama giriş veri kümenize 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 birkaçını kullanabilirsiniz. Diğer bir seçenek olarak, kendi kuyruklarınızı da kullanabilirsiniz. Ancak, bu örnek yalnızca, adının ilk sekiz baytındaki CSQ4SAMP karakterlerine sahip kuyruklarla birlikte çalışır.

### z/OS z/OS’ündeki örnek CICS uygulamalarının adları

Bu konuda, örnek CICS uygulamaları için sağlanan programların bir özeti yer alır.

CICS uygulama programları aşağıdaki tablolarda özetlenir:

- Çizelge 186 sayfa 1206 Put ve Get örnekleri

- Çizelge 187 sayfa 1206 Kuyruk Öznitelikleri örneği
- Çizelge 188 sayfa 1206 Mail Manager örneği (yalnızca COBOL)
- Çizelge 189 sayfa 1207 Kredi Denetimi örneği
- Çizelge 190 sayfa 1208 Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örnekleri

<i>Çizelge 186. CICS Put ve Get örnekleri</i>				
Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4CCK1	C	Kaynak program koy	SCSQC37S	SCSQCICS
CSQ4CCJ1	C	Kaynak programı al	SCSQC37S	SCSQCICS
CSQ4CVJ1	COBOL	Kaynak programı al	SCSQC OBS	SCSQCICS
CSQ4CVK1	COBOL	Kaynak program koy	SCSQC OBS	SCSQCICS
CSQ4S100	Bağımsız	CICS sistem tanımlaması veri kümesi	SCSQPROC	Yok

<i>Çizelge 187. CICS Kuyruk Öznitelikleri örneği</i>				
Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı	Kitaplıkta sağlanan yürütülebilir dosya
CSQ4CVC1	COBOL	Kaynak program	SCSQC OBS	SCSQCICS
CSQ4VMSG	COBOL	İleti Tanımlaması	SCSQC OBC	Yok
CSQ4VCMS	COBOL	BMS ekran tanımlaması	SCSQMAPS	SCSQCICS (adı CSQ4ACM)
CSQ4CAC1	Çevirici	Kaynak program	SCSQASMS	SCSQCICS
CSQ4AMSG	Çevirici	İleti Tanımlaması	SCSQMACS	Yok
CSQ4ACMS	Çevirici	BMS ekran tanımlaması	SCSQMAPS	SCSQCICS (adı CSQ4ACM)
CSQ4CCC1	C	Kaynak program	SCSQC37S	SCSQCICS
CSQ4CMMSG	C	İleti Tanımlaması	SCSQC370	Yok
CSQ4CCMS	C	BMS ekran tanımlaması	SCSQMAPS	SCSQCICS (adı CSQ4ACM)
CSQ4S100	Bağımsız	CICS sistem tanımlaması veri kümesi	SCSQPROC	Yok

<i>Çizelge 188. CICS Mail Manager örneği (yalnızca COBOL)</i>		
Üye adı	Tanım	Kaynak dosya kitaplıkta sağlandı
CSQ4CVD	IBM MQ for z/OS nesne tanımları	SCSQPROC

Çizelge 188. CICS Mail Manager örneği (yalnızca COBOL) (devamı var)

Üye adı	Tanım	Kaynak dosya kitaplıkta sağlandı
CSQ4CVD1	Menü programı kaynağı	SCSQCOBS
CSQ4CVD2	Posta Alma programı için kaynak	SCSQCOBS
CSQ4CVD3	Görüntü İletisi programı için kaynak	SCSQCOBS
CSQ4CVD4	Posta Gönderme programı için kaynak	SCSQCOBS
CSQ4CVD5	Takma ad programı için kaynak	SCSQCOBS
CSQ4VDMS	BMS ekran tanımı kaynağı	SCSQMAPS
CSQ4S100	CICS sistem tanımlaması veri kümesi	SCSQPROC
CSQ4VD0	Veri tanımı	SCSQCOBC
CSQ4VD3	Veri tanımı	SCSQCOBC
CSQ4VD4	Veri tanımı	SCSQCOBC

Çizelge 189. CICS Kredi Denetimi örneği

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı
CSQ4CVB	Bağımsız	IBM MQ nesne tanımları	SCSQPROC
CSQ4CCB	Bağımsız	IBM MQ nesne tanımları	SCSQPROC
CSQ4CVB1	COBOL	User-interface programına ilişkin kaynak	SCSQCOBS
CSQ4CVB2	COBOL	Kredi uygulama yöneticisi için kaynak	SCSQCOBS
CSQ4CVB3	COBOL	Hesap denetimi için kaynak-hesap programı	SCSQCOBS
CSQ4CVB4	COBOL	Dağıtım programı kaynağı	SCSQCOBS
CSQ4CVB5	COBOL	Ajans-sorgu programı kaynağı	SCSQCOBS
CSQ4CCB1	C	User-interface programına ilişkin kaynak	SCSQ37S
CSQ4CCB2	C	Kredi uygulama yöneticisi için kaynak	SCSQ37S
CSQ4CCB3	C	Hesap denetimi için kaynak-hesap programı	SCSQ37S
CSQ4CCB4	C	Dağıtım programı kaynağı	SCSQ37S
CSQ4CCB5	C	Ajans-sorgu programı kaynağı	SCSQ37S
CSQ4CB0	C	İçerme dosyası	SCSQ370
CSQ4CBMS	C	BMS ekran tanımı kaynağı	SCSQMAPS
CSQ4VBMS	COBOL	BMS ekran tanımı kaynağı	SCSQMAPS
CSQ4VB0	COBOL	Veri tanımı	SCSQCOBC
CSQ4VB1	COBOL	Veri tanımı	SCSQCOBC

Çizelge 189. CICS Kredi Denetimi örneği (devamı var)

Üye adı	Dil için	Tanım	Kaynak dosya kitaplıkta sağlandı
CSQ4VB2	COBOL	Veri tanımı	SCSQCOBC
CSQ4VB3	COBOL	Veri tanımı	SCSQCOBC
CSQ4VB4	COBOL	Veri tanımı	SCSQCOBC
CSQ4VB5	COBOL	Veri tanımı	SCSQCOBC
CSQ4VB6	COBOL	Veri tanımı	SCSQCOBC
CSQ4VB7	COBOL	Veri tanımı	SCSQCOBC
CSQ4VB8	COBOL	Veri tanımı	SCSQCOBC
CSQ4BAQ	Bağımsız	VSAM veri kümesi için kaynak	SCSQPROC
CSQ4FILE	Bağımsız	CSQ4CVB3 tarafından kullanılan VSAM veri kümesi oluşturmak için JCL	SCSQPROC
CSQ4S100	Bağımsız	CICS sistem tanımlaması veri kümesi	SCSQPROC

Çizelge 190. CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örnekleri

Üye adı	Tanım	Kaynak dosya kitaplıkta sağlandı
CSQ4CVCN	Basit İletim Tüketimi programı kaynağı	SCSQCOBS
CSQ4CVCT	Control Message Consumption programı için kaynak	SCSQCOBS
CSQ4CVEV	Olay İşleyici programı için kaynak	SCSQCOBS
CSQ4CVPT	Message put Client programı için kaynak	SCSQCOBS
CSQ4CVRG	Registration Client programı için kaynak	SCSQCOBS
CSQ4S100	CICS System Definition veri kümesi	SCSQPROC

### **z/OS** **Preparing the sample application for the IMS environment on z/OS**

Kredi Denetimi örnek uygulamasının bir bölümü IMS ortamında çalışabilir.

Uygulamanın bu bölümünü CICS örneğiyle çalıştırmak üzere hazırlamak için, önce [“Preparing the sample applications for the CICS environment on z/OS”](#) sayfa 1205’inde açıklanan adımları gerçekleştirin.

Daha sonra aşağıdaki adımları gerçekleştirin:

1. Herhangi bir IMS IBM MQ for z/OS uygulamasını oluştururken uyguladığınız adımları gerçekleştirin. Bu adımlar [“Building IMS \(BMP or MPP\) applications”](#) sayfa 1064’inde listelenir. Kullanılacak kitaplık üyeleri [Çizelge 191](#) sayfa 1209’listesinde yer alıyor.
2. Uygulama programını ve veritabanını IMSolarak tanımlayın. Bu seçeneği etkinleştirmek için PSBGEN, DBDGEN, ACB tanımlaması, MSGEN ve IMSDALOC deyimleriyle örnekler verilir.
3. Bu amaçla sağlanan örnek JCL ' yi (CSQ4ILDB) uyarlayarak ve çalıştırarak CSQ4CA veritabanını yükleyin. Bu JCL, veritabanını CSQ4BAQkütüğünden veri ile yükler. IMS denetim bölgesini CSQ4CAveritabanına ilişkin DD (DD) deyimiyile güncelleyin.



4. Bu amaçla sağlanan örnek JCL ' yi uyarlayarak ve çalıştırarak, check-account programını toplu ileti işleme (BMP) programı olarak başlatın. Bu JCL, toplu iş odaklı BMP programını başlatır. Programı ileti odaklı 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, Credit Check örnek IMS uygulaması için sağlanan kaynakların ve JSP ' lerin listesini içeren bir tablo sağlar.

Üye adı	Tanım	Kitaplıkta sağlanan
CSQ4CVB	IBM MQ nesne tanımları	SCSQPROC
CSQ4ICB3	Hesap denetimi için kaynak-hesap programı	SCSQC37S
CSQ4ICBL	Checking-account veritabanının yüklenmesi için kaynak	SCSQC37S
CSQ4CBI	Veri tanımı	SCSQC370
CSQ4PSBL	Veritabanı yükleme programı için PSBGEN JCL	SCSQPROC
CSQ4PSB3	Hesap denetimi için PSBGEN JCL	SCSQPROC
CSQ4DBDS	CSQ4CA veritabanı için DBDGEN JCL	SCSQPROC
CSQ4GIMS	CSQ4IVB3 ve CSQ4CA için IMSGEN makro tanımlamaları	SCSQPROC
CSQ4ACBG	CSQ4IVB3 için uygulama denetim bloğu (ACB) tanımı	SCSQPROC
CSQ4BAQ	Veritabanı kaynağı	SCSQPROC
CSQ4ILDB	Veritabanı yükleme işi için örnek çalıştırma JCL	SCSQPROC
CSQ4ICBR	Hesap denetimi için örnek çalıştırma JCL	SCSQPROC
CSQ4DYNA	Veritabanı için IMSDALOC makro tanımlamaları	SCSQPROC

#### **z/OS** The Put samples on z/OS

Put örnek programları, MQPUT çağrısını kullanan bir kuyruğa ileti yerleştirdi.

Kaynak programlar, C ve COBOL ' de toplu ve CICS ortamlarında sağlanır (bkz. [Çizelge 179 sayfa 1201](#) ve [Çizelge 186 sayfa 1206](#) ).

### Put örneğinin tasarımı

Program mantığının içinden geçen akış şöyledir:

1. MQCONN çağrısını kullanarak kuyruk yöneticisine bağlanın. Bu çağrı başarısız olursa, tamamlama ve neden kodlarını yazdırın ve işlemeyi durdurun.

**Not:** Örneği bir CICS ortamında çalıştırıyorsanız, bir MQCONN çağrısı yayınlamaya gerek yoktur; varsa, DEF\_HCONN değerini döndürür. İzleyen MQI çağrılarında ilişkin MQHC\_DEF\_HCONN bağlantı tanıtıcısını kullanabilirsiniz.

2. MQOPEN çağrısını MQOO\_OUTPUT seçeneğiyle kullanarak kuyruğu açın. Bu çağrıya ilişkin girişlerde, program “1” sayfa 1211adımında döndürülen bağlantı tanıtıcısını kullanır. Nesne tanımlayıcı yapısı (MQOD) için, program için parametre olarak geçirilen kuyruk adı alanı dışında tüm alanlar için varsayılan değerleri kullanır. MQOPEN çağrısı başarısız olursa, tamamlama ve neden kodlarını yazdırın ve işlemeyi durdurun.
3. MQPUT çağrıları yayınlayan program içinde, kuyruğa gereken sayıda ileti konuncaya kadar bir döngü yaratın. Bir MQPUT çağrısı başarısız olursa, döngü erken terk edilir, başka MQPUT çağrıları denenmez ve tamamlanma ve neden kodları döndürülür.
4. Close the queue using the MQCLOSE call with the object handle returned in step “2” sayfa 1211. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın.
5. “1” sayfa 1211adımında döndürülen bağlantı tanıtıcısı ile MQDISC çağrısını kullanarak kuyruk yöneticisiyle bağlantıyı kesin. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın.

**Not:** Örneği bir CICS ortamında çalıştırıyorsanız, bir MQDISC çağrısı yayınlamaya gerek yoktur.

### z/OS üzerindeki toplu iş ortamına ilişkin koyma örnekleri

Toplu iş ortamı için örnekler koymayı dikkate aldığımızda bu konuyu kullanın.

To run the samples, edit and run the sample JCL, as described in “z/OSüzerinde toplu iş ortamı için örnek uygulamalar hazırlama ve çalıştırma” sayfa 1200.

Programlar, COBOL 'de C ve virgüllerde boşluklarla ayrılmış bir EXEC PARM' de aşağıdaki parametreleri alır:

1. Kuyruk yöneticisinin adı (4 karakter)
2. Hedef kuyruğun adı (48 karakter)
3. İletilerin sayısı (en çok 4 haneli)
4. İletide yazabilmek için doldurma karakteri (1 karakter)
5. İletide yazılacak karakter sayısı (en çok 4 karakter)
6. İletinin sürekliliği (1 karakter: Kalıcı ya da kalıcı olmayan için H değeri)

Bu parametrelerin herhangi birini yanlış bir şekilde 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 CSQ4BVJRörneklerinde gösterilen parametrelerin yerleşim düzenini kullanırsanız bu farklar en aza indirilir. Farkların hiçbiri MQI ile ilişkilendirilmedi.
- CSQ4BCK1 , gönderilen ileti sayısı ve iletilerin uzunluğu için dörtten fazla basamak girmenize olanak tanır.
- İki sayısal alan için, 1-9999 aralığında herhangi bir basamağı 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 eksi değerler girerseniz, bir hata alabilirsiniz. Örneğin, -1 değerini girerseniz, COBOL programı 1 byte 'lık bir ileti gönderir, ancak C programı bir hata alır.
- Her iki program için ( CSQ4BCK1 ve CSQ4BVK1), iletinin kalıcı olmasını istiyorsanız, kalıcı saklama değiştirgesi + + PER + + olarak P girmeniz gerekir. Bunu yapmazsanız, ileti kalıcı olmaz.

### The Put samples for the CICS environment on z/OS

CICS ortamı için örnekler koymayı dikkate aldığımızda bu konuyu kullanın.

Hareketler, virgüllerle ayrılmış olarak aşağıdaki parametreleri alır:

1. İletilerin sayısı (en çok 4 haneli)
2. İletide yazabilmek için doldurma karakteri (1 karakter)
3. İletide yazılacak karakter sayısı (en çok 4 karakter)
4. İletinin sürekliliği (1 karakter: Kalıcı ya da kalıcı olmayan için H değeri)
5. Hedef kuyruğun adı (48 karakter)

Bu parametrelerden herhangi birini yanlış bir şekilde girerseniz, uygun hata iletileri alırsınız.

COBOL örneği için, aşağıdaki bilgileri girerek CICS ortamında put (put) örneğini çağırın:

```
MVPT,9999,*,9999,P,QUEUE.NAME
```

C örneği için, aşağıdaki bilgileri girerek CICS ortamında put (ekle) örneğini 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 ilişkilendirilmedi.
- 48 karakterden daha uzun bir kuyruk adı girerseniz, uzunluğu 48 karakter üst sınırı olarak kesilir, ancak hata iletileri döndürülmez.
- İşlemi 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 eksi değerler girerseniz, bir hata alabilirsiniz. Örneğin, -1 girerseniz, COBOL programı 1 byte 'lık bir ileti gönderir ve C programı malloc () kaynaklı bir hata ile olağandışı biter.
- Her iki program için de CSQ4CCK1 ve CSQ4CVK1 için, iletinin kalıcı olmasını istiyorsanız, kalıcılık parametresine P girin. Kalıcı olmayan iletiler için, kalıcılık parametresine N değerini girin. Başka bir değer girerseniz, bir hata iletileri alırsınız.
- Varsayılan değerler, program çağırma işlemi sırasında ayarlanmaları dışında, tüm değiştirgeler için kullanıldığından, iletiler uyumluluk noktasına yerleştirilir.

## **The Get samples on z/OS**

Alma örnek programları, MQGET çağrısını kullanarak kuyruktan ileti alır.

Kaynak programlar, C ve COBOL ' de toplu ve CICS ortamlarında sağlanır (bkz. [Çizelge 179 sayfa 1201](#) ve [Çizelge 186 sayfa 1206](#) ).

## **Get Sample on z/OS(Alma örneği tasarımı)**

Get (Al) örneğinin tasarımı ve bazı kullanım notlarının dikkate alınması hakkında bilgi edinin.

Program mantığının içinden geçen akış şöyledir:

1. MQCONN çağrısını kullanarak kuyruk yöneticisine bağlanın. Bu çağrı başarısız olursa, tamamlama ve neden kodlarını yazdırın ve işlemeyi durdurun.

**Not:** Örneği bir CICS ortamında çalıştırıyorsanız, bir MQCONN çağrısı yayınlamaya gerek yoktur; varsa, DEF\_HCONN değerini döndürür. İzleyen MQI çağrılarında ilişkin MQHC\_DEF\_HCONN bağlantı tanıtıcısını kullanabilirsiniz.

2. MQOPEN çağrısını MQOO\_INPUT\_SHARED ve MQOO\_BROWSE seçenekleri ile kullanarak kuyruğu açın. Bu çağrıya ilişkin girişlerde, program "1" sayfa 1211 adımıyla döndürülen bağlantı tanıtıcısını kullanır. Nesne tanımlayıcı yapısı (MQOD) için, program için parametre olarak geçirilen kuyruk adı alanı dışında

tüm alanlar için varsayılan değerleri kullanır. MQOPEN çağrısı başarısız olursa, tamamlama ve neden kodlarını yazdırın ve işlemeyi durdurun.

3. Kuyruktan gerekli sayıda ileti alınincaya kadar, MQGET çağrılarını yayınlayarak program içinde bir döngü yaratın. Bir MQGET çağrısının başarısız olması durumunda, döngü erken terk edilir, başka MQGET çağrılarını denemez ve tamamlanma ve neden kodları döndürülür. MQGET çağrısında aşağıdaki seçenekler belirtilir:

- MQGMO\_NO\_BEKLEME
- MQGMO\_ACCEPT\_TRUNCATED\_MESSAGE
- MQGMO\_SYNCPOINT ya da MQGMO\_NO\_SYNCPOINT
- MQGMO\_BROWSE\_FIRST ve MQGMO\_BROWSE\_NEXT

Bu seçeneklere ilişkin açıklamalar için [MQGET](#) başlıklı konuya bakın. Her ileti için, ileti numarası ve ileti verileri, iletinin uzunluğunun ardından yazılır.

4. Close the queue using the MQCLOSE call with the object handle returned in step “2” sayfa 1211. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın.
5. “1” sayfa 1211 adımıyla döndürülen bağlantı tanıtıcısı ile MQDISC çağrısını kullanarak kuyruk yöneticisiyle bağlantıyı kesin. Bu çağrı başarısız olursa, tamamlanma ve neden kodlarını yazdırın.

**Not:** Örneği bir CICS ortamında çalıştırıyorsanız, bir MQDISC çağrısını yayınlamaya gerek yoktur.

## Kullanım notları

- Örnekleri basit tutmak için, dil sürümleri arasında bazı küçük işlevsel farklılıklar vardır. Ancak, örnek çalıştırma JCL, CSQ4BCJR ve CSQ4BVJR örneklerinde gösterilen parametrelerin yerleşim düzenini kullanıyorsanız bu farklılıklar en aza indirilir. Farkların hiçbiri MQI ile ilişkilendirilmedi.
- CSQ4BCJ1 , alınan ileti sayısı için dörtten fazla basamak girmenize olanak tanır.
- 64 KB ' den uzun iletiler kısaltılır.
- CSQ4BCJ1 , yalnızca ilk NULL (\0) karakteri görüntüleninceye kadar görüntülendiğinden, karakter iletilerini doğru olarak görüntüleyebilir.
- Sayısal ileti sayısı alanı için, 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 değerini girebilirsiniz. Sayısal olmayan ya da eksi değerler girerseniz, bir hata alabilirsiniz. Örneğin, -1 girerseniz, COBOL programı bir ileti alır, ancak C programı herhangi bir ileti almaz.
- For both programs, CSQ4BCJ1 and CSQ4BVJ1, enter B in the get parameter, ++GET++, if you want to browse the messages.
- Her iki program için, CSQ4BCJ1 ve CSQ4BVJ1 için, syncpoint 'te alınacak iletiler için, + + SYNC + + eşitleme noktası parametresine S girin.

## z/OS üzerindeki toplu iş ortamına ilişkin alma örnekleri

To run the samples, edit and run the sample JCL, as described in [“z/OS üzerinde toplu iş ortamı için örnek uygulamalar hazırlama ve çalıştırma” sayfa 1200](#).

Programlar, COBOL 'de C ve virgüllerde boşluklarla ayrılmış bir EXEC PARM' de aşağıdaki parametreleri alır:

1. Kuyruk yöneticisinin adı (4 karakter)
2. Hedef kuyruğun adı (48 karakter)
3. Elde edilen ileti sayısı (en çok 4 haneli)
4. Göz atma/alma iletisi seçeneği (1 karakter: Göz atmak için B ya da iletileri yok etmek için D ya da D)
5. Syncpoint denetimi (1 karakter: syncpoint için S ya da syncpoint yok için N)

Bu parametrelerden herhangi birini yanlış girerseniz, uygun hata iletileri alırsınız.

Örneklerden çıkış SYSPRINT veri kümesine yazılır:

```
=====
PARAMETERS PASSED :
QMGR      - VC9
QNAME     - A.Q
NUMMSGs   - 000000002
GET       - D
SYNCPPOINT - N
=====
MQCONN SUCCESSFUL
MQOPEN SUCCESSFUL
000000000 : 000000010 : *****
000000001 : 000000010 : *****
000000002 MESSAGES GOT FROM QUEUE
MQCLOSE SUCCESSFUL
MQDISC SUCCESSFUL
```

**z/OS** z/OS ortamında CICS ortamına ilişkin alma örnekleri  
Special considerations for the Get samples for the CICS environment.

İşlemler, EXEC PARM ' de virgülle ayrılmış olarak aşağıdaki parametreleri alır:

1. Elde edilen ileti sayısı (en çok dört basamağa kadar)
2. Göz atma/alma iletisi seçeneği (bir karakter: Göz atmak için B ya da iletileri ortadan yok etmek için D ya da D)
3. Syncpoint denetimi (bir karakter: syncpoint için S ya da syncpoint yok için N)
4. Hedef kuyruğun adı (48 karakter)

Bu parametrelerden herhangi birini yanlış girerseniz, uygun hata iletileri alırsınız.

COBOL örneği için, aşağıdaki bilgileri girerek CICS ortamında get (Al) örneğini çağırın:

```
MVGT,9999,B,S,QUEUE.NAME
```

C örneği için, aşağıdaki bilgileri girerek CICS ortamında get (Al) örneğini çağırın:

```
MCGT,9999,B,S,QUEUE.NAME
```

İletiler kuyruktan alındığında, bunlar CICS işlemi ile aynı adı taşıyan bir CICS geçici depolama kuyruğuna (örneğin, C örneği için MCGT) konacaklar.

Alma örneklerinin örnek çıkışı şöyledir:

```
***** 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 ilişkilendirilmedi.
- 48 karakterden daha uzun bir kuyruk adı girerseniz, uzunluğu 48 karakter üst sınırı olarak kesilir, ancak hata iletisi döndürülmez.
- İşlemi girmeden önce CLEAR tuşuna basın.
- CSQ4CCJ1 , yalnızca ilk NULL (\0) karakteri görüntüleninceye kadar görüntülendiğinden, karakter iletilerini doğru olarak görüntüleyebiliyor.
- 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.

- CBOL ' de 24 526 bayttan daha uzun iletiler ve COBOL içindeki 9 950 bayt kısaltılır. Bunun nedeni, CICS geçici saklama alanı kuyruklarının kullanılmasıdır.
- Her iki program için de CSQ4CCK1 ve CSQ4CVK1 programları için, iletilere göz atmak istiyorsanız alma (get) parametresi için B girin, tersi durumda D girin. Bu, yıkıcı MQGET çağrılarını gerçekleştirir. Başka bir değer girerseniz, bir hata iletileri alırsınız.
- Her iki program için, CSQ4CCJ1 ve CSQ4CVJ1 için, syncpoint 'teki iletileri almak için syncpoint parametresine S girin. Syncpoint parametresine N değerini girerseniz, MQGET çağrıları syncpoint dışında yayınlanır. Başka bir değer girerseniz, bir hata iletileri alırsınız.

## **The Browse sample on z/OS**

Göz At (Browse) örneği, MQGET çağrısını kullanarak kuyruklardaki iletilere nasıl göz atacağını gösteren bir toplu uygulamadır.

Uygulama, kuyrukta bulunan tüm iletileri aşar ve her birinin ilk 80 baytını yazdırır. Bu uygulamayı, kuyruklardaki iletilere değiştirmeden, kuyrukta arama yapmak için kullanabilirsiniz.

Kaynak programlar ve örnek çalıştırma JCL, COBOL, çevirici, PL/I ve C dillerinde sağlanır (bkz. [Çizelge 180 sayfa 1201](#)).

To start the application, edit and run the sample run JCL, as described in [“z/OS üzerinde toplu iş ortamı için örnek uygulamalar hazırlama ve çalıştırma” sayfa 1200](#). JCL ' nin çalıştırılması sırasında kuyruğun adını belirterek, kendi kuyruklarınızdan birine ilişkin iletilere bakabilirsiniz.

Uygulamayı çalıştırdığınızda (ve kuyrukta bazı iletiler varsa), çıkış veri kümesi şuna bakar:

```

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 iletilerini içerir. MQI çağrılarında herhangi biriyle ilgili bir hata oluşursa, tamamlanma ve neden kodları çıkış veri kümesine eklenir.

## **Design of the Browse sample on z/OS**

Göz At örnek uygulaması tek bir program modülü kullanır; biri desteklenen programlama dillerinin her birinde sağlanır.

Program mantığının içinden geçen akış şöyledir:

1. Bir yazdırma verileri kümesini açın ve raporun başlık satırını yazdırın. Kuyruk yöneticisi ve kuyruk adlarının JCL çalıştırımından geçirilip geçirildiğini denetleyin. Her iki ad da iletiliyse, adları içeren raporun satırlarını yazdırın. Yazmadıysanız, bir hata iletileri yazdırın, yazdırma veri kümesini kapatın ve işlemeyi durdurun.

Programın, JCL ' den geçireceği parametreleri test ettiği şekilde, programın yazıldığı dile bağlıdır; daha fazla bilgi için bkz. [“z/OS' ta dile bağımlı tasarım konuları” sayfa 1215](#).

2. MQCONN çağrısını kullanarak kuyruk yöneticisine bağlanın. Bu çağrı başarılı değilse, tamamlanma ve neden kodlarını yazdırın, yazdırma verileri kümesini kapatın ve işlemeyi durdurun.
3. MQOPEN çağrısını MQOO\_BROWSE seçeneğiyle kullanarak kuyruğu açın. Bu çağrıya ilişkin girişlerde, program [“2” sayfa 1214](#) adımıyla döndürülen bağlantı tanıtıcısını kullanır. Nesne tanımlayıcı yapısı (MQOD) için, kuyruk adı dışında tüm alanlar için varsayılan değerleri kullanır ( [“1” sayfa 1214](#) adımıyla

aktarıldı). Bu çağrı başarılı değilse, tamamlanma ve neden kodlarını yazdırın, yazdırma verileri kümesini kapatın ve işlemeyi durdurun.

4. MQGET çağrısını kullanarak kuyruğun ilk iletisine göz atın. Bu çağrıya giriş sırasında, program şunları belirtir:

- The connection and queue handles from steps “2” sayfa 1214 and “3” sayfa 1214
- Tüm alanları ilk değerlerine ayarlı olan bir MQMD yapısı
- İki seçenek:
  - MQGMO\_BROWSE\_FIRST
  - MQGMO\_ACCEPT\_TRUNCATED\_MSG
- İletiden kopyalanan verileri tutmak için 80 baytlık bir arabellek

MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneği, çağrıda belirtilen 80 baytlık arabelleğinden daha uzun bir ileti olsa bile çağrıyı tamamlamasına olanak sağlar. İleti arabelleğinden uzunsa, ileti, arabelleğe sığması için kesilir ve tamamlanma ve neden kodları bu iletiyi gösterecek şekilde ayarlanır. Bu örnek, iletilerin okunmasını kolaylaştırmak için iletilerin 80 karaktere kısaltılması amacıyla tasarlanmıştır. Arabellek boyutu bir DEFINE deyimiyle ayarlanır; bu nedenle, istediğiniz zaman kolayca değiştirebilirsiniz.

5. MQGET çağrısına başarısız oluncaya kadar aşağıdaki döngüye bakın:

a. Rapor gösteren bir satır yazdır:

- İletinin sıra numarası (bu, göz atma işlemlerine ilişkin bir sayıdır).
- İletinin gerçek uzunluğu (kısaltılmış uzunluğa değil). Bu değer MQGET çağrısının DataLength alanında döndürülür.
- İleti verilerinin ilk 80 baytı.

b. MQMD yapısındaki MsqId ve CorrelId alanlarını boş değere sıfırlayın

c. Bu iki seçenekle MQGET çağrısını kullanarak sonraki iletiye göz atın:


- MQGMO\_BROWSE\_NEXT
- MQGMO\_ACCEPT\_TRUNCATED\_MSG

6. MQGET çağrısı başarısız olursa, göz atma imlecinin kuyruğun sonuna gelmesinden dolayı çağrıyı başarısız olup olmadığını görmek için neden kodunu sınavın. In this case, print the Raporun sonu message and go to step “7” sayfa 1215 ; otherwise, print the completion and reason codes, close the print data set, and stop processing.

7. Close the queue using the MQCLOSE call with the object handle returned in step “3” sayfa 1214.

8. “2” sayfa 1214 adımıyla döndürülen bağlantı tanıtıcısı ile MQDISC çağrısını kullanarak kuyruk yöneticisiyle bağlantıyı kesin.

9. Yazdırma verileri kümesini kapatın ve işlemeyi durdurun.

 z/OS' ta dile bağımlı tasarım konuları

Kaynak modüller, dört programlama dilindeki Göz At örneği için sağlanır.

Kaynak modüller arasında iki temel fark vardır:

- JCL çalıştırımından geçirilen parametreleri test ederken, virgül karakteri (,) için COBOL, PL/I ve çevirici dili modülleri aramanızı sağlar. JCL, PARM=( , LOCALQ1) değerini geçerse, uygulama varsayılan kuyruk yöneticisinde LOCALQ1 kuyruğunu açmayı dener. Virgülden (ya da virgülden) sonra ad yoksa, uygulama bir hata döndürür. C birimi, virgül karakterini aramaz. JCL tek bir parametreyi geçerse (örneğin, PARM=( ' LOCALQ1 ' )), C modülü bunu varsayılan kuyruk yöneticisinde kuyruk adı olarak kullanır.
- Çevirici dil modülünü basit tutmak için yazdırma raporunu oluşturduğunda yy/ddd tarih biçimini kullanır (örneğin, 05/116). Diğer modüller takvim tarihini aa/gg/yy biçiminde kullanır.

## **z/OS** **z/OSüzerindeki Print Message (Yazdırma İletisi) örneği**

İleti Yazdır örneği, MQGET çağrısını kullanarak kuyruktan tüm iletilerin nasıl kaldırılacağını gösteren bir toplu uygulamadır.

Print Message Sample, üç parametre kullanır:

1. Kuyruk yöneticisinin adı
2. Kaynak kuyruğun adı
3. Özellikler için isteğe bağlı bir parametre

Ayrıca, her ileti için ileti tanımlayıcısının alanları ve ardından ileti verileri yazdırılıyor. Bu program, verileri hem onaltılı, hem de karakter olarak (yazdırılabilir ise) yazdırır. Karakter yazdırılmaz ise, program bu karakteri bir nokta karakteri (.) ile değiştirir. Bir kuyruğa ileti yerleştiren bir uygulamayla ilgili sorunları tanımlarken bu programı kullanabilirsiniz.

Özellik parametresine ilişkin izin verilen değerler şunlardır:

Değer	Davranış
0	Varsayılan davranış, IBM WebSphere MQ 6 için olduğu gibi. Uygulamaya teslim edilen özellikler, iletinin alındığı <b>PropertyControl</b> kuyruk özneliğine bağlıdır.
1	Bir ileti tanıtıcısı yaratılır ve MQGET ile kullanılır. İleti tanımlayıcısında (ya da uzantıda) yer alan durumlar dışında, iletinin özellikleri ileti tanımlayıcısına benzer bir şekilde görüntülenir. Örneğin: <pre>****Message properties**** property name: property value</pre> Ya da kullanılabilir özellik yoksa: <pre>****Message properties**** None</pre> Sayısal değerler printfkullanılarak görüntülenir, dizgi değerleri tek tırnak işaretleriyle çevrilir ve bayt dizgileri, ileti tanımlayıcısına göre X ve tek tırnak işaretleriyle çevrelenir.
2	MQGMO_NO_XX_ENCODE_CASE_ONE properties belirtildi, bu nedenle yalnızca ileti tanımlayıcısı özellikleri döndürülecek.
3	İleti verilerinde tüm özelliklerin döndürülmesi içinMQGMO_PROPERTIES_FORCE_MQRFH2 belirtildi.
4	MQGMO_PROPERTIES_COMPATIBILITY belirtildi; böylece, bir IBM WebSphere MQ 6 özelliğinin dahil edilip edilmediğine bağlı olarak tüm özellikler döndürülebilecek, tersi durumda özellikler atılır.

İletileri, kuyruktan kaldırmak yerine, iletileri göz atması için değiştirebilirsiniz. Bunu yapmak için, BROWSE makrosunu tanımlamak için -DABROWSE seçeneğiyle, “z/OSüzerinde Yazdırma İletisi örneğini tasarlama” sayfa 1217’ da belirtildiği şekilde derleyin. SCSQLOAD kitaplığında yürütülebilir kod sağlanır. CSQ4BCG0 modülü -DBROWSE; ile oluşturulmuş; CSQ4BCG1 birimi yok edici olarak kuyruğu okuyor.

Uygulamanın, C dilinde yazılmış tek bir kaynak programı vardır. Örnek çalıştırma JCL kodu da sağlanır (bkz. Çizelge 181 sayfa 1202 ).

To start the application, edit and run the sample run JCL, as described in “z/OSüzerinde toplu iş ortamı için örnek uygulamalar hazırlama ve çalıştırma” sayfa 1200. Uygulamayı çalıştırdığınızda (ve kuyruқта bazı iletiler varsa), çıkış veri kümesi Şekil 144 sayfa 1217’ ta 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'C3E2D840D4D8F1C5404040404040C1EA537F03167D88'
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'000000000000000000000000000000000000000000000000000000000'
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 144. İleti Yazdırma örnek uygulamasından bir rapor örneği

 z/OS üzerinde Yazdırma İletisi örneğini tasarlama  
Print Message Sample uygulaması, C dilinde yazılmış tek bir programı kullanır.

Program mantığının içinden geçen akış şöyledir:

1. Kuyruk yöneticisi ve kuyruk adlarının JCL çalıştırmından geçirilip geçirildiğini denetleyin. Yoksa, bir hata iletisi yazdırın ve işlemeyi durdurabilirler.
2. MQCONN çağrısını kullanarak kuyruk yöneticisine bağlanın. Bu çağrı başarılı değilse, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun; tersi durumda kuyruk yöneticisinin adını yazdırın.
3. MQOPEN çağrısını MQOO\_INPUT\_SHARED seçeneği ile kullanarak kuyruğu açın.

**Not:** Uygulamanın kuyruktan kaldırmak yerine iletilere göz atmasını istiyorsanız, örneği -DABROWSE ile derleyin ya da kaynağın başına #define BROWSE ögesini ekleyin. Bunu yaptığınız zaman, makro ön işlemcisi, derlemede MQOO\_BROWSE seçeneğini seçen programa satır ekler.

Bu çağrıya ilişkin girişlerde, program “2” sayfa 1217 adımında döndürülen bağlantı tanıtıcısını kullanır. Nesne tanımlayıcı yapısı (MQOD) için, kuyruk adı dışında tüm alanlar için varsayılan değerleri kullanır

- (“1” sayfa 1217 adımımda aktarıldı). Bu çağrı başarılı değilse, tamamlanma ve neden kodlarını yazdırın ve işlemeyi durdurun; tersi durumda, kuyruğun adını yazdırın.
4. İleti özelliklerini elde etmek için bir ileti tanıtıcısı kullanırsanız, izleyen MQGET çağrılarınıyla kullanmak üzere bu tür bir tanıtıcı yaratmak üzere MQCRTMH kullanılmasını kullanın. Bu çağrı başarılı değilse, tamamlama ve neden kodlarını yazdırın ve işlemeyi durdurun.
  5. İleti alma seçeneklerini, ileti özellikleri için istek işlemi yansıtabilecek şekilde ayarlayın.
  6. MQGET çağrısına başarısız oluncaya kadar aşağıdaki döngüye bakın:
    - a. Arabelleğin arabelleğindeki veriler tarafından bozulmaması için, arabelleğin arabelleğini boşluklara ilk kullanıma hazırlayın.
    - b. MQGET çağrısının, kuyruktan ilk iletiyi seçmesi için, MQMD yapısındaki MsgId ve CorrelId alanlarını boş değere ayarlayın.
    - c. MQGET çağrısını kullanarak kuyruktan bir ileti alın. Bu çağrıya giriş sırasında, program şunları belirtir:
      - The connection and object handles from steps “2” sayfa 1217 and “3” sayfa 1217.
      - Tüm alanları ilk değerlerine ayarlı olan bir MQMD yapısı. (MsgId ve CorrelId , her MQGET çağrısı için boş değere sıfırlanır.)
      - MQGMO\_NO\_WAIT seçeneğini belirleyin.

**Not:** Uygulamanın kuyruktan kaldırmak yerine iletilere göz atmasını istiyorsanız, örneği -DABROWSE ile derleyin ya da kaynağın başına #define BROWSE ögesini ekleyin. Bunu yaparken, makro ön işlemcisi, derleme sırasında MQGMO\_BROWSE\_NEXT seçeneğini seçen programa satır ekler. Bu seçenek, imlecin yürürlükteki nesne tanıtıcısı ile daha önce kullanılmadığı bir kuyrukla ilgili olarak çağrıldığında, imleç, ilk iletiden önce mantıksal olarak konumlandırılır.
      - İletiden kopyalanan verileri tutmak için 64KB büyüklüğünün arabelleği.
    - d. printMD alt yordamını çağırın. Bu, ileti tanımlayıcısındaki her alanın adını ve ardından içeriğini yazdırır.
    - e. “4” sayfa 1218 adımımda bir ileti tanıtıcısı yarattıysa, herhangi bir ileti özelliğini görüntülemek için printProperties alt yordamını çağırın.
    - f. İletinin uzunluğunu, ardından ileti verilerini yazdırın. İleti verilerinin her bir satırı bu biçimdeki biçimlerde yer alıyor:
      - Verilerin bu bölümünün görelisi konumu (onaltılı)
      - 16 bayt 'lık onaltılı veri
      - Aynı 16 baytlık veri biçimi, yazdırılabilir ise (yazdırılmayan karakterler dönemlere göre değiştirilir).
  7. MQGET çağrısı başarısız olursa, kuyrukta başka ileti olmadığından arama başarısız olup olmadığını görmek için neden kodunu sınavın. Bu durumda, iletiyi yazdırın: Başka ileti yok; tersi durumda, tamamlanma ve neden kodlarını yazdırın. Her iki durumda da “9” sayfa 1218adımına gidin.

**Not:** MQGET çağrısı, 64KB ' den daha fazla veri içeren bir ileti bulursa başarısız olur. Daha büyük iletileri işlemek üzere programı değiştirmek için, aşağıdakilerden birini yapabilirsiniz:

    - Arama, ilk 64KB veri alır ve kalanı atar; MQGET çağrısına MQGMO\_ACCEPT\_TRUNCATED\_MSG seçeneğini ekleyin.
    - Programı, bu miktarda veri içeren bir ileti bulunduğu kuyruğun kuyruğunda bırakması
    - Arabelleğin büyüklüğünü artırın
  8. “4” sayfa 1218 adımımda bir ileti tanıtıcısı yarattıysa, bu iletiyi silmek için MQDLTMH çağrısını çağırın.
  9. Close the queue using the MQCLOSE call with the object handle returned in step “3” sayfa 1217.
  10. “2” sayfa 1217adımımda döndürülen bağlantı tanıtıcısı ile MQDISC çağrısını kullanarak kuyruk yöneticisiyle bağlantıyı kesin.

## z/OS z/OSüzerinde Kuyruk Öznitelikleri örneği

Kuyruk Öznitelikleri örneği, MQINQ ve MQSET çağrılarının kullanılmasını gösteren bir conversasyonel kiptir CICS uygulamasıdır.

Bu program, kuyrukların **InhibitPut** ve **InhibitGet** özniteliklerinin değerlerini ve bu değerlerin nasıl değiştirileceğini, böylece programların iletileri koyamayacağı ya da kuyruktan ileti alamayacağı şekilde nasıl değiştirileceğini gösterir. Bir programı sınavken, bu şekilde bir kuyruğu *kilitlemek* isteyebilirsiniz.

Kendi kuyruklarınıza yanlışlıkla müdahale etmeyi önlemek için, bu örnek yalnızca adının ilk sekiz baytındaki CSQ4SAMP karakterlerine sahip bir kuyruk nesnesiyle çalışır. Ancak, kaynak kod, bu kısıtlamanın nasıl kaldırılacağını göstermek için açıklamaları içerir.

Kaynak programlar COBOL, çevirici ve C dillerinde sağlanır (bkz. [Çizelge 187 sayfa 1206](#)).

Örneğin, çevirici dil sürümü yeniden girilebilen kod kullanımları kullanır. Bunu yapmak için, örneğin o sürümünde her bir MQI çağrısı kodunun MF anahtar sözcüğünü içerdiğini fark etmeniz; örneğin:

```
CALL MQCONN, (NAME, HCONN, COMPCODE, REASON), MF=(E, PARMAREA), VL
```

(The VL keyword means that you can use the CICS Execution Diagnostic Facility (CEDF) supplied transaction for debugging the program.) Yeniden eğlendirilebilir programlar yazma hakkında daha fazla bilgi için bkz. [System/390 çevirici dilinde kodlama](#).

Uygulamayı başlatmak için, CICS sisteminizi başlatın ve aşağıdaki CICS işlemlerini kullanın:

- COBOL için, MVC1
- Çevirici dili için MAC1
- C için, MCC1

Adım 3' te belirtilen CSD veri kümesini değiştirerek, bu işlemlerin herhangi birinin adını değiştirebilirsiniz.

### Örneğin tasarımı

Örneği başlattığınızda, aşağıdakilerin alanları olan bir ekran eylemi görüntüler:

- Kuyruğun adı
- Kullanıcı isteği (geçerli işlemler şunlardır: sorgulamak, izin vermek ya da engellemedir)
- Kuyruğa ilişkin koyma işlemlerinin yürürlükteki durumu
- Kuyruğa ilişkin alma işlemlerinin yürürlükteki durumu

İlk iki alan kullanıcı girişi içindir. The last two fields are filled by the application: they show the word INHIBITED or the word ALLOWED.

Uygulama, ilk iki alana girdiğiniz değerleri doğrular. Kuyruk adının CSQ4SAMP karakterleriyle başladığını ve işlem alanındaki geçerli üç istekten birine girdiğinizi denetler. Uygulama tüm girdinizi büyük harfe dönüştürür; bu nedenle, küçük harf karakterleri içeren adlara sahip kuyruklar kullanamazsınız.

**Eylem** alanına inquire yazarsanız, program mantığının içinden aşağıdaki akış olur:

1. MQOO\_SORGULAMA seçeneği ile MQOPEN çağrısını kullanarak kuyruğu açın.
2. MQIA\_INHIBIT\_GET ve MQIA\_INSTRIBIT\_PUT seçicileri kullanarak MQINQ ' u çağırın
3. MQCLOSE çağrısını kullanarak kuyruğu kapat
4. MQINQ çağrısının **IntAttr**s parametresindeki döndürülen öznitelikleri çözümlemeniz ve ilgili ekran alanlarına uygun olduğu şekilde, ENGELLEYICI ya da İZINE IZIN VERILEN Sözcükleri Uygun şekilde hareket ettirin

**Eylem** alanına inhibit yazarsanız, program mantığının içinden aşağıdaki akış olur:

1. MQOO\_SET seçeneğiyle MQOPEN çağrısını kullanarak kuyruğu açın.
2. MQIA\_INHIBIT\_GET ve MQIA\_INHIBIT\_PUT ve **IntAttr**s parametresindeki MQQA\_GET\_INHIBITED ve MQQA\_PUT\_INHIBITED değerleri kullanılarak MQSET ' i çağırın.

3. MQCLOSE çağrısını kullanarak kuyruğu kapat
4. Engellenmiş sözcüğü ilgili ekran alanlarına taşı

**Eylem** alanına allow yazarsanız, uygulama, bir inhibit isteği için benzer işleme gerçekleştirir. Tek farklar, özniteliklerin ve ekranda görüntülenen sözcüklerin ayarlarıdır.

When the application opens the queue, it uses the default connection handle to the queue manager. ( CICS establishes a connection to the queue manager when you start your CICS system.) Uygulama şu aşamada aşağıdaki hataları kaplayabilir:

- Uygulama kuyruk yöneticisine bağlı değil
- Kuyruk yok
- Kullanıcının kuyruğa erişme yetkisi yok
- Uygulamanın kuyruğun açılması için yetkisi yok

Diğer MQI hataları için, uygulama tamamlanma ve neden kodlarını görüntüler.

### **The Mail Manager sample on z/OS**

Mail Manager örnek uygulaması, hem tek bir ortam içinde, hem de farklı ortamlar arasında ileti göndermeyi ve almayı gösteren bir program grubudur. Uygulama, kullanıcıların farklı kuyruk yöneticileri kullansa bile ileti alışverişi için izin veren basit bir elektronik postalama sistemidir.

Uygulama, MQOPER çağrısını kullanarak ve sistem komutu giriş kuyruğuna IBM MQ for z/OS komutlarını koyarak kuyrukların nasıl yaratılacağını gösterir.

Uygulamanın üç sürümü sağlanır:

- COBOL ' da yazılmış bir CICS uygulaması
- COBOL ' da yazılmış bir TSO uygulaması
- C içinde yazılmış bir TSO uygulaması

### **Preparing the Mail Manager sample on z/OS**

Posta Yöneticisi, iki ortamda çalışan sürümlerde sağlanır. Uygulamayı çalıştırmadan önce gerçekleştirmeniz gereken hazırlık, kullanmak istediğiniz ortama bağlıdır.

Kullanıcılar, her sistemde oturum açma kullanıcı kimlikleri aynı olduğu sürece, hem TSO hem de CICS ' deki posta kuyruklarına ve takma ad kuyruklarına erişebilirler.

Başka bir kuyruk yöneticisine ileti göndermeden önce, o kuyruk yöneticisine bir ileti kanalı ayarlamalısınız. To do this, use the channel control function of IBM MQ, described in [Kanal denetimi işlevi](#).

## **TSO ortamı için örnek hazırlanıyor**

Aşağıdaki adımları izleyin:

1. Örneği, "[z/OSüzerinde TSO ortamı için örnek uygulamalar hazırlama](#)" sayfa 1203 içinde açıklandığı gibi hazırlayın.
2. Örneğin, aşağıda belirtilecek örnek için CLIST ' i uyarlayın:

- Panellerin konumu
- İleti dosyasının yeri
- Yükleme modüllerinin yeri
- Uygulamayla kullanmak istediğiniz kuyruk yöneticisinin adı

Örneğin her bir dil sürümü için ayrı bir CLIST sağlanır:

- COBOL sürümü için: CSQ4RVD1
- C sürümü için: CSQ4RCD1

3. Uygulama tarafından kullanılan kuyrukların kuyruk yöneticisinde kullanılabilir olduğundan emin olun. (Kuyruklar CSQ4CVD' de tanımlanır.)

**Not:** VS COBOL II, ISPF ile çoklu görevi desteklemeyi desteklemez. Bu, bölünmüş bir ekranın her iki tarafındaki Posta Yöneticisi örnek uygulamasını kullanamayamazsınız. Bunu yapmazsanız, sonuçlar tahmin edilemez.

### *Running the Mail Manager sample on z/OS*

To start the sample in the CICS Transaction Server for z/OS environment, run transaction MAIL. CICS' ta önceden oturum açmadıysanız, uygulama, postanızı gönderebileceği bir kullanıcı kimliği girmenizi ister.

Uygulamayı başlattığınızda, bu, posta kuyruğunuzu açar. Bu kuyruk yoksa, uygulama sizin için bir kuyruk yaratır. Posta kuyruklarının adları CSQ4SAMP.MAILMGR. *userid*; burada *klnckimliği* , ortama bağlıdır:

#### **TSO ' da**

Kullanıcının TSO Tanıtıcısı

#### **İçindeCICS**

Mail Manager başlatıldığında kullanıcının CICS oturum açma kullanıcı kimliği ya da kullanıcı tarafından girilen kullanıcı kimliği

Posta Yöneticisi 'nin kullandığı kuyruk adlarının tüm bölümleri büyük harfli olmalıdır.

Daha sonra uygulama, aşağıdakiler için seçenekleri olan bir menü panosunu gösterir:

- Gelen postayı oku
- Posta Gönder
- Takma ad yarat

Menü panosu ayrıca, posta kuyruğunuzda kaç ileti beklediğini de gösterir. Menü seçeneklerinin her biri daha ayrıntılı bir pano görüntüler:

#### **Gelen postayı oku**

Posta Yöneticisi, posta kuyruğunuzda bulunan iletilerin bir listesini görüntüler. (Kuyruktaki yalnızca ilk 99 ileti görüntülenir.) Bu panoya ilişkin bir örnek için bkz. [Şekil 147 sayfa 1225](#). Bu listeden bir ileti seçtiğinizde, iletinin içeriği görüntülenir (bkz. [Şekil 148 sayfa 1226](#)).

#### **Posta Gönder**

Bir pano girmenizi ister:

- İleti göndermek istediğiniz kullanıcının adı.
- Posta kuyruklarının sahibi olan kuyruk yöneticisinin adı
- İletinizin metni

Kullanıcı adı alanında, Posta Yöneticisi 'ni kullanarak yarattığınız bir kullanıcı kimliği ya da takma ad girebilirsiniz. Kullanıcının posta kuyruğu kullandığınız kuyruk yöneticisine aitse ve kullanıcı adı alanına bir takma ad girdiyseniz, kuyruk yöneticisi adı alanını boş bırakabilirsiniz ve bu alanı boş bırakmanız gerekir.

- Yalnızca bir kullanıcı adı belirtirseniz, program önce adın bir takma ad olduğunu varsayar ve iletiyi bu adı kullanarak tanımlanan nesneye gönderir. Böyle bir takma ad yoksa, program iletiyi o adı içeren yerel bir kuyruğa göndermeyi dener.
- Hem bir kullanıcı adı, hem de kuyruk yöneticisi adı belirtirseniz, program iletiyi, bu iki adla tanımlanan posta kuyruğuna gönderir.

Örneğin, uzak kuyruk yöneticisinde ( QM12) kullanıcı JONESM ' ye bir ileti göndermek istiyorsanız, aşağıdaki iki yoldan birine bir ileti gönderebilirsiniz:

- Use both fields to specify user JONESM at queue manager QM12.
- Bu kullanıcıya ilişkin bir takma ad (örneğin, MARY) tanımlayın ve bunları, kullanıcı adı alanına MARY koyarak ve kuyruk yöneticisi adı alanında hiçbir şey yazmayarak bir ileti gönderin.

## Takma ad yarat

Sık olarak iletişim kurduğunuz başka bir kullanıcıya ileti gönderirken kullanabileceğiniz, kolay anımsanan bir ad tanımlayabilirsiniz. Diğer kullanıcının kullanıcı kimliğini ve posta kuyruklarının sahibi olan kuyruk yöneticisinin adını girmeniz istenir.

Takma adlar, CSQ4SAMP.MAILMGR. *userid.nickname*; burada *klncikimliği* , kendi kullanıcı kimliğiniz ve *takma ad* kullanmak istediğiniz takma addır. Bu şekilde yapılandırılmış adlar varsa, kullanıcıların her biri kendi takma adlarına sahip olabilir.

Programın yarattığı kuyruk tipi, Nickname (Takma ad yarat) panelindeki alanların nasıl tamamlandığına bağlıdır:

- Yalnızca bir kullanıcı adı belirtirseniz ya da kuyruk yöneticisi adı, Posta Yöneticisi 'nin bağlı olduğu kuyruk yöneticisiyle aynıysa, program bir diğer ad kuyruğu oluşturur.
- Hem bir kullanıcı adı, hem de kuyruk yöneticisi adı belirtirseniz (ve kuyruk yöneticisi, Posta Yöneticisi 'nin bağlı olduğu kuyruk yöneticisi değilse), program uzak bir kuyruğun yerel tanımlamasını yaratır. Program, bu tanımın çözdüğü kuyruğun varlığını denetmez ya da uzak kuyruk yöneticisinin var olduğunu da denetmez.

For example, if your own user ID is SMITHK and you create a nickname called MARY for user JONESM (who uses the remote queue manager QM12), the nickname program creates a local definition of a remote queue named CSQ4SAMP.MAILMGR.SMITHK.MARY. This definition resolves to Mary's mail queue, which is CSQ4SAMP.MAILMGR.JONESM at queue manager QM12. Kuyruk yöneticisi QM12 ' yi kendiniz kullanıyorsanız, program aynı adı içeren bir diğer ad kuyruğu yaratır (CSQ4SAMP.MAILMGR.SMITHK.MARY).

TSO uygulamasının C sürümü, ISPF ' nin ileti işleme yeteneklerinin COBOL sürümünü kullandığından daha çok kullanılabilirlik sağlar. C ve COBOL sürümlerine göre farklı hata iletilerinin görüntülendiğini fark edebilirsiniz.

### *Design of the Mail Manager sample on z/OS*

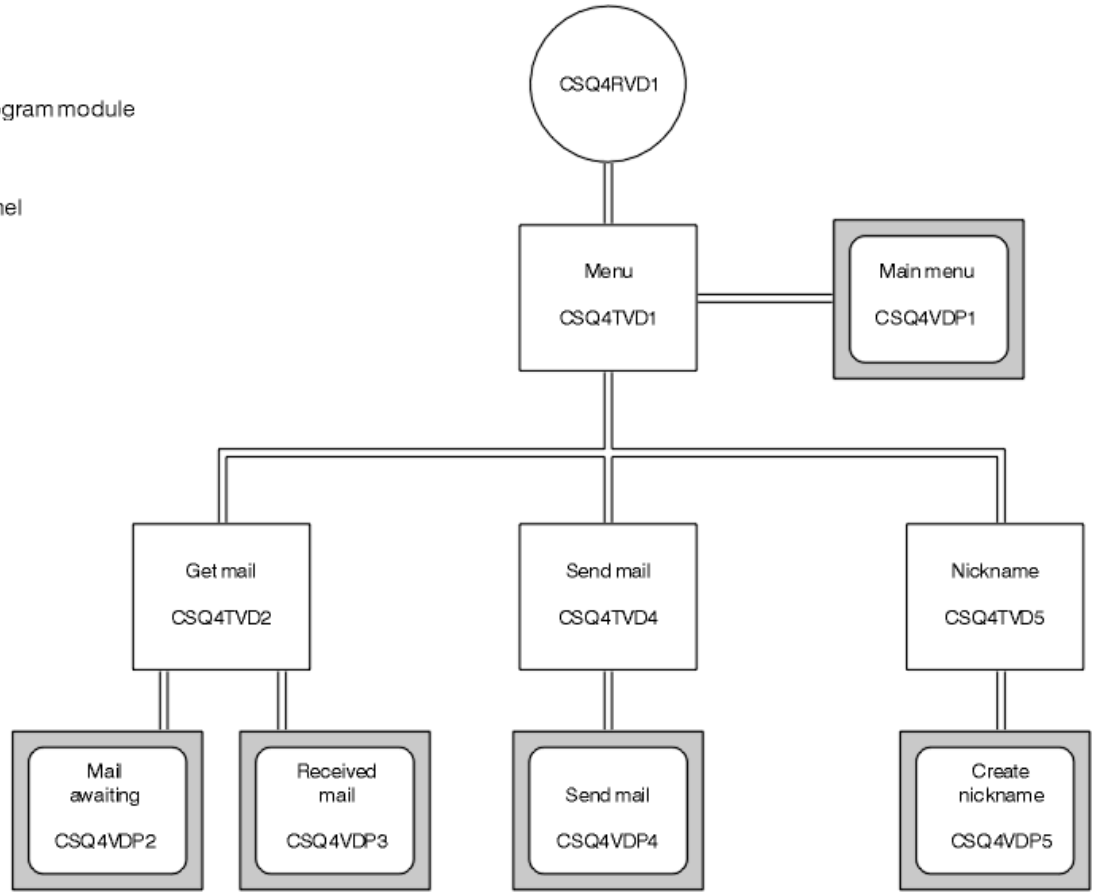
Aşağıdaki kısımlarda, Mail Manager örnek uygulamasını oluşturan programların her biri açıklanmıştır.

Uygulamaların kullandığı programlar ve panolar arasındaki ilişkiler, TSO sürümü için [Şekil 145 sayfa 1223](#) içinde ve CICS Transaction Server for z/OS sürümü için [Şekil 146 sayfa 1224](#) içinde gösterilir.

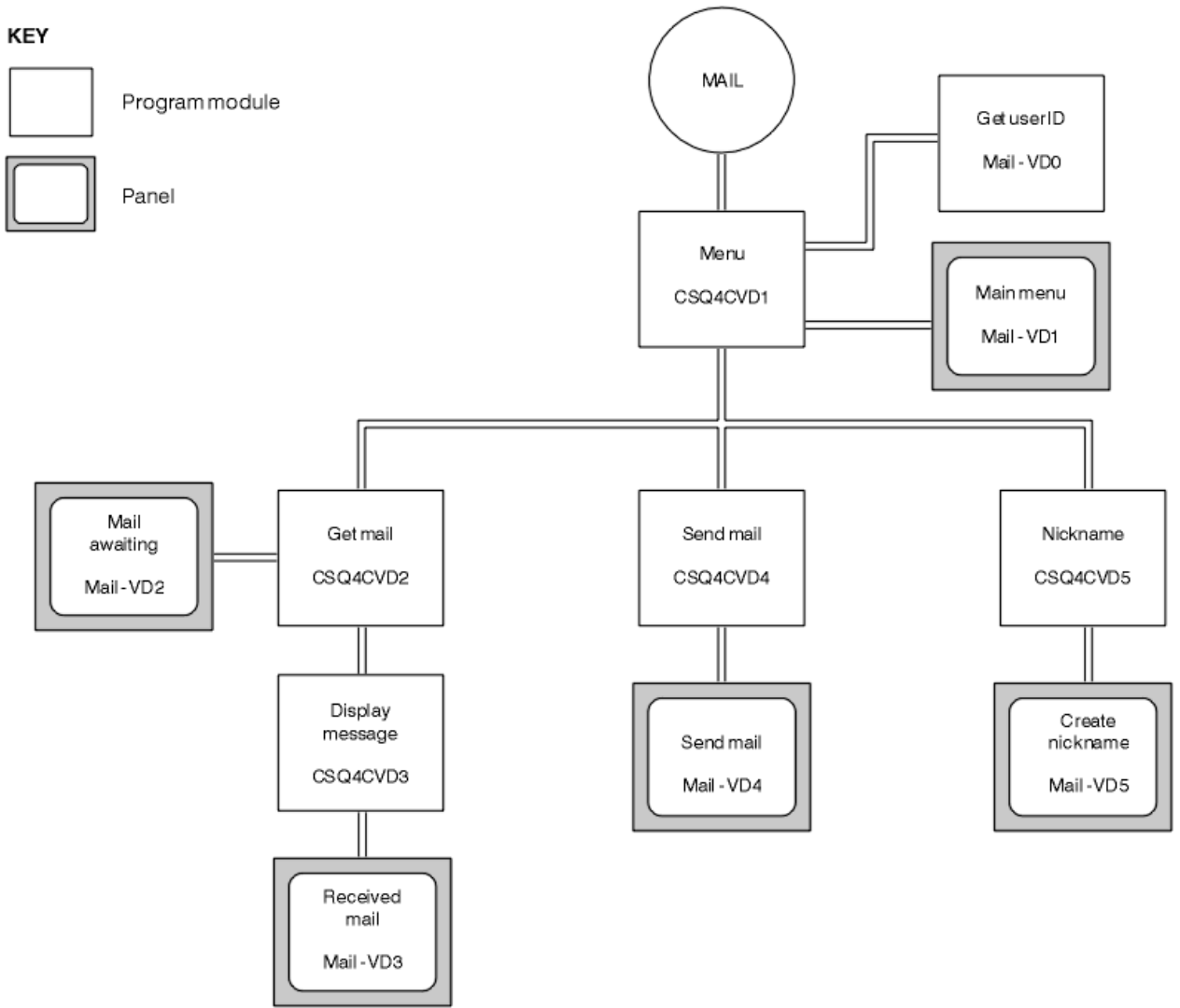
**KEY**

 Program module

 Panel



Şekil 145. Posta Yöneticisinin TSO sürümlerine ilişkin programlar ve panolar



Şekil 146. Posta Yöneticisi 'nin CICS sürümüne ilişkin programlar ve panolar

#### z/OS z/OS üzerinde menü programı

TSO ortamında, menü programı CLIST tarafından çağrılır. In the CICS environment, the program is invoked by transaction MAIL.

Menü programı (TSO için CSQ4TVD1 , CICS için CSQ4CVD1 ), takımdaki ilk programdır. Menüyü görüntüler (TSO için CSQ4VDP1 , CICS için VD1 ) ve menüden seçtiklerinde diğer programları çağırır.

Program, kullanıcının kimliğini ilk olarak alır:

- Programın CICS sürümünde, kullanıcı CICS' ta oturum açmışsa, kullanıcı kimliği CICS komutu ASSIGN USERID kullanılarak elde edilir. Kullanıcı oturum açmazsa, program oturum açma panosunu görüntüler (CSQ4VD0), kullanıcıdan bir kullanıcı kimliği girmesini ister. Bu program içinde herhangi bir güvenlik işlemi yoktur; kullanıcı herhangi bir kullanıcı kimliği verebilir.
- TSO sürümünde, kullanıcının kimliği CLIST içinde TSO ' dan elde edilir. Menü programına ISPF paylaşılan havuzunda bir değişken olarak geçirilir.

Program kullanıcı kimliğini edindikten sonra, kullanıcının bir posta kuyruğuna (CSQ4SAMP.MAILMGR. kullanıcı kimliği ). Bir posta kuyruğu yoksa, program sistem komut giriş kuyruğuna bir ileti koyarak bir ileti yaratır. Bu ileti, IBM MQ for z/OS komutunun DEFINE QLOCAL komutunu içerir. Bu komutun kullandığı nesne tanımı, kuyruğun maksimum derinliğini 9999 iletiyle ayarlar.

Program, sistem komutu giriş kuyruğundan gelen yanıtları işlemek için de geçici bir dinamik kuyruk yaratır. Bunu yapmak için, program, dinamik kuyruk için şablon olarak SYSTEM.DEFAULT.MODEL.QUEUE



belirterek, MQOPEN çağrısını kullanır. Kuyruk yöneticisi, CSQ4SAMP; öneğine sahip bir adla geçici dinamik kuyruk yaratır; adın geri kalan kısmı kuyruk yöneticisi tarafından üretilir.

Daha sonra program, kullanıcının posta kuyruğunu açar ve kuyruğun yürürlükteki derinliğini sorarak, kuyruklardaki ileti sayısını bulur. Bunu yapmak için, program MQINA\_current\_q\_depth selector belirtilerek MQINQ çağrısını kullanır.

Daha sonra program, menüyü görüntüleyen bir döngü gerçekleştirir ve kullanıcının yaptığı seçimi işler. Kullanıcı PF3 tuşuna bastığında döngü durdurulur. Geçerli bir seçim yapıldığında, uygun program başlatılır; tersi durumda bir hata iletisi görüntülenir.

#### **z/OS** z/OS' ta posta ve görüntüleme-ileti programları

Uygulamanın TSO sürümlerinde, alma-posta ve görüntüleme iletisi işlevleri aynı program tarafından (CSQ4TVD2) gerçekleştirilir. Uygulamanın CICS sürümünde, bu işlevler ayrı programlar (CSQ4CVD2 ve CSQ4CVD3) tarafından gerçekleştirilir.

Posta Bekleniyor panosu (TSO için CSQ4VDP2), CICS için VD2; Örneğin, [Şekil 147 sayfa 1225](#) ' a bakın), kullanıcının posta kuyruğunda bulunan tüm iletileri gösterir. Bu listeyi yaratmak için, program kuyruktaki tüm iletilere göz atmak, her birine ilişkin bilgileri saklamak için MQGET çağrısını kullanır. Program, görüntülenen bilgilere ek olarak, her iletinin MsgId ve CorrelId ' lerini 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 147. Bekleme iletileri listesini gösteren bir pano örneği*

Mail Abekleme panosundan, kullanıcı bir ileti seçebilir ve iletinin içeriğini görüntüleyebilir (örnek için [Şekil 148 sayfa 1226](#) ' a bakın). The program uses the MQGET call to remove this message from the queue, using the MsgId and CorrelId that the program noted when it browsed all the messages. Bu MQGET çağrısı, MQGMO\_SYNCPOINT seçeneği kullanılarak gerçekleştirilir. Bu program iletinin içeriğini görüntüler, sonra bir eşitleme noktası bildirir: Bu, MQGET çağrısını kesinleştirir, bu nedenle artık ileti artık yok.



Program, bu tanımın çözdüğü kuyruğun varlığını denetmez ya da uzak kuyruk yöneticisinin var olduğunu da denetmez.

Program, sistem komutu giriş kuyruğundan gelen yanıtları işlemek için de geçici bir dinamik kuyruk yaratır.

Kuyruk yöneticisi, programın beklediği bir nedenden dolayı takma ad kuyruğunu yaratamazsa (örneğin, kuyruk zaten var), program kendi hata iletisini görüntüler. Kuyruk yöneticisi, programın beklemediği bir nedenle kuyruğu yaratamazsa, program, komut sunucusu tarafından programa döndürülen hata iletilerinden en fazla iki tanesi görüntüler.

**Not:** Her takma ad için, takma ad programı yalnızca bir diğer ad kuyruğu ya da uzak bir kuyruğun yerel tanımlaması yaratır. Bu kuyruk adlarının çözemediği yerel kuyruklar, yalnızca takma ada sahip olan kullanıcı kimliği Posta Yöneticisi uygulamasını başlatmak için kullanıldığında yaratılır.

### **The Credit Check sample on z/OS**

Credit Check örnek uygulaması, IBM MQ for z/OS tarafından sağlanan özelliklerin çoğunu nasıl kullanacağını gösteren bir dizi program grubudur. Bir uygulamanın birçok bileşen programının ileti kuyruklama tekniklerini kullanarak, iletileri birbirine nasıl geçirebileceğini gösterir.

Örnek, bağımsız bir CICS uygulaması olarak çalıştırılabilir. Ancak, hem CICS hem de IMS ortamlarında sağlanan olanakları kullanan bir ileti kuyruklama uygulamasının nasıl tasarlanması olduğunu göstermek için, bir modül de IMS toplu ileti işleme programı olarak sağlanır. Örneğe ilişkin bu uzantı "[z/OS üzerindeki Credit Check örneğine IMS uzantısı](#)" sayfa 1236 içinde açıklanmıştır.

Ayrıca, örneği birden çok kuyruk yöneticisinden çalıştırabilir ve uygulamanın her eşgörünümü arasında ileti gönderebilirsiniz. Bunu yapmak için bkz. "[The Credit Check sample with multiple queue managers on z/OS](#)" sayfa 1236.

CICS programları C ve COBOL ' de teslim edilir. Tek IMS programı yalnızca C ' de teslim edilir. Sağlanan veri kümeleri [Çizelge 189 sayfa 1207](#) ve [Çizelge 191 sayfa 1209](#) ' te gösterilir.

Uygulama, banka müşterileri kredi istediklerinde riski değerlendirmenin bir yöntemini sergiliyor. Uygulama, bir bankanın kredi taleplerini işleme almak için iki şekilde nasıl çalışabileceğini gösterir:

- Banka personeli doğrudan bir müşteriyle doğrudan ilgilenirken, hesap ve kredi riski bilgilerine anında erişim izni isterler.
- Banka personeli, yazılı uygulamalarla çalışırken, hesap ve kredi riski bilgileri için bir dizi talep gönderebilir ve yanıtlarla daha sonra ilgilenebilir.

Uygulamadaki mali ve güvenlik ayrıntıları, ileti kuyruklama tekniklerinin açık olması için basit olarak tutulurlar.

### **z/OS' ta Kredi Denetimi örneğini hazırlama ve çalıştırma**

Credit Check örneğini hazırlamak ve çalıştırmak için aşağıdaki adımları gerçekleştirin:

1. Bazı örnek hesaplarla ilgili bilgileri bulunduran VSAM veri kümesini yaratın. Bu işlemi, CSQ4FILE veri kümesinde sağlanan JCL ' yi düzenleyerek ve çalıştırarak gerçekleştirin.
2. Perform the steps in "[Preparing the sample applications for the CICS environment on z/OS](#)" sayfa 1205. (The additional steps that you must perform if you want to use the IMS extension to the sample are described in "[z/OS üzerindeki Credit Check örneğine IMS uzantısı](#)" sayfa 1236.)
3. CKTI tetikleme izleyicisini başlatın ( IBM MQ for z/OS ile birlikte sağlanır) CSQ4SAMP.INITIATION.QUEUE, CICS hareket CKQC ' yi kullanarak.
4. To start the application, start your CICS system and use the transaction MVB1.
5. İlk panodan **Hemen** ya da **Toplu İş** sorgularını seçin.

Anında ve toplu sorgu panoları benzerdir; [Şekil 149 sayfa 1228](#) , Anında Sorgu panosunu 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 149. Kredi Denetimi örnek uygulaması için Anında Sorgu panosu

6. Uygun alanlara bir hesap numarası ve kredi tutarı girin. Bu alanlara hangi bilgilerin girileceğini görmek için bkz. [“Sorgu panolarına bilgi girilmesi” sayfa 1228](#) .

## Sorgu panolarına bilgi girilmesi

Kredi Denetimi örnek uygulaması, sorgu panolarının **İstenen tutar** alanında girdiğiniz verilerin tamsayı biçiminde olduğunu denetler.

If you enter one of the following account numbers, the application finds the appropriate account name, average account balance, and credit worthiness index in the VSAM data set CSQ4BAQ:

- 2222222222
- 3111234329
- 3256478962
- 3333333333
- 3501676212
- 3696879656
- 4444444444
- 5555555555
- 6666666666
- 7777777777

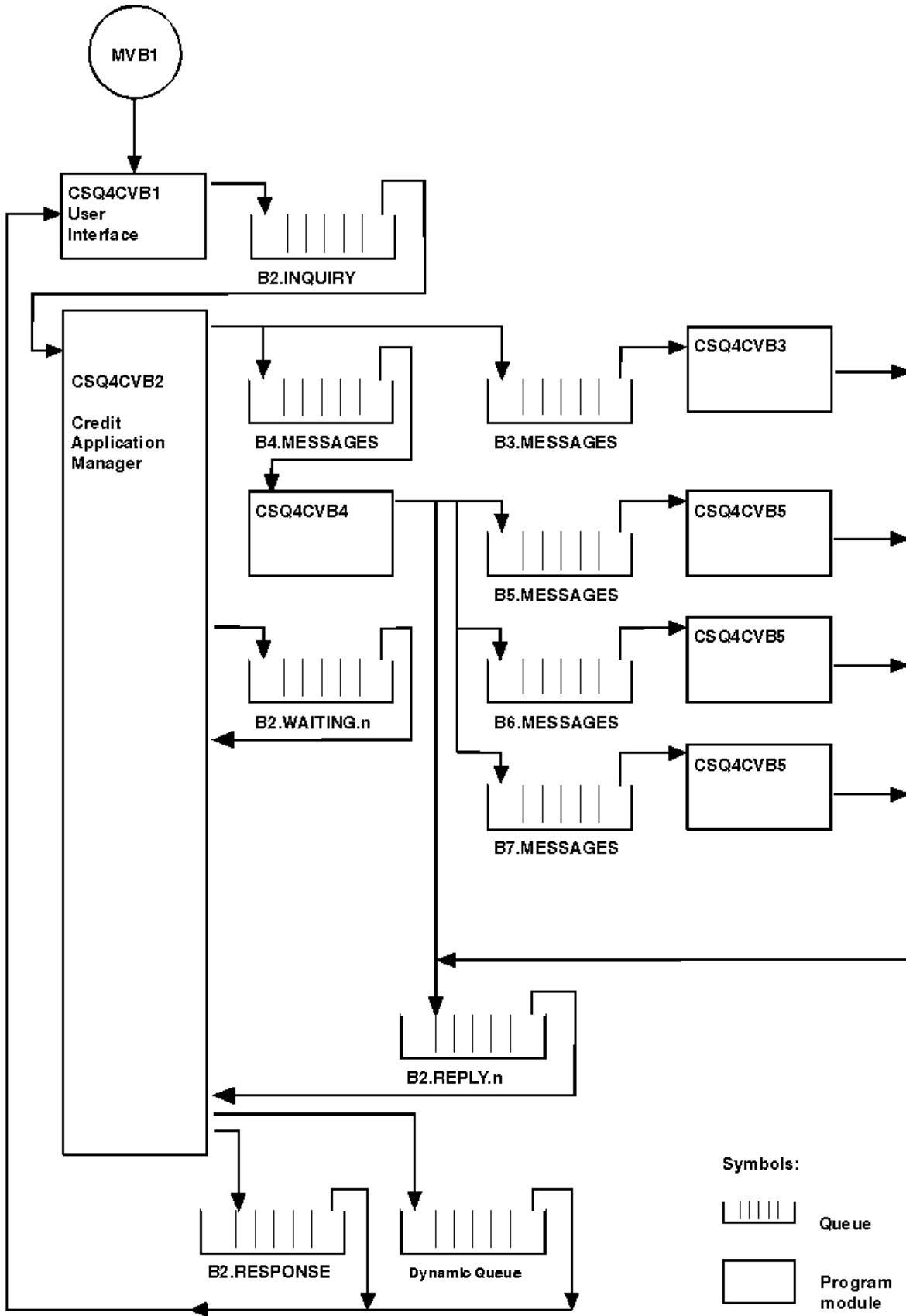
Diğer alanlara bilgi girebilir ya da başka alanlara bilgi girilebilir. Uygulama, girdiğiniz tüm bilgileri saklar ve oluşturduğu raporlarda aynı bilgileri döndürür.

### ► z/OS Design of the Credit Check sample on z/OS

Bu bölümde, Credit Check örnek uygulamasını oluşturan programların her birinin tasarımı açıklanmaktadır.

Uygulamanın tasarımı sırasında dikkate alınan bazı tekniklerle ilgili daha fazla bilgi için bkz. [“Design considerations for the Credit check sample on z/OS” sayfa 1234](#).

[Şekil 150 sayfa 1229](#) , uygulamayı oluşturan programları ve ayrıca bu programların hizmet veren kuyuklarını gösterir. Bu şekilde, CSQ4SAMP öneki, daha kolay anlaşılmasını sağlamak için tüm kuyruk adlarından çıkarılmıştır.



Şekil 150. Credit Check örnek uygulamasına ilişkin programlar ve kuyruklar (yalnızca COBOL programları)

## **z/OS** User interface program (CSQ4CVB1) on z/OS

Etkileşimli kip CICS hareketi MVB1' i başlattığınızda, uygulama için kullanıcı arabirimi programı başlatılır.

Bu program, CSQ4SAMP.B2.INQUIRY kuyruğuna sorgu iletileri koyar ve bu sorguları, sorguyu yaparken belirlediği bir yanıtlama kuyruğundan yanıtlar alır. Kullanıcı arabiriminden anında ya da toplu sorguları sunabilirsiniz:

- Anında sorgular için, program, yanıt kuyruğu olarak kullandığı geçici bir dinamik kuyruk yaratır. Bu, her sorgunun kendi yanıtlama kuyruğuna sahip olduğu anlamına gelir.
- For batch inquiries, the user-interface program gets replies from the queue CSQ4SAMP.B2.RESPONSE. Basitlik için, program bu yanıt kuyruğundan tüm sorgularına yanıt alır. Bir bankanın her bir MVB1kullanıcısı için ayrı bir yanıt kuyruğu kullanmak isteyebileceğini görmek kolaydır, böylece her biri yalnızca başlattıkları sorgulara yanıt verebilirler.

Toplu iş ve anında kipteki uygulamada kullanılan iletilerin özellikleri arasındaki önemli farklar şunlardır:

- Toplu iş için, iletiler düşük önceliğe sahiptir; bu nedenle, anında kipe girilen kredi taleplerinden sonra işlenir. Ayrıca, iletiler kalıcıdır, bu nedenle uygulama ya da kuyruk yöneticisinin yeniden başlatılması gerekiyorsa, bunlar kurtarılır.
- Anında çalışma için, iletiler yüksek önceliğe sahiptir, bu nedenle toplu kipte girilen herhangi bir kredi isteğinden önce işlenir. Ayrıca, iletiler kalıcı değildir; uygulama ya da kuyruk yöneticisinin yeniden başlatılması gerekiyorsa bu iletiler atılır.

Ancak, tüm durumlarda, kredi isteği iletilerinin özellikleri uygulama boyunca yayılır. Örneğin, yüksek öncelikli bir isteğin sonucu olan tüm iletiler de yüksek önceliğe sahip olur.

## **z/OS** Credit application manager (CSQ4CVB2) on z/OS

Credit Application Manager (CAM) programı, Kredi Denetimi uygulaması için işlemlerin çoğunu gerçekleştirir.

The CAM is started by the CKTI trigger monitor (supplied with IBM MQ for z/OS) when a trigger event occurs on either queue CSQ4SAMP.B2.INQUIRY or queue CSQ4SAMP.B2.REPLY. *n*, burada *n*, bir yanıt kuyrukları kümesinin birini tanımlayan bir tamsayıdır. Tetikleme iletileri, tetikleme olayının ortaya çıktığı kuyruğun adını içeren verileri içerir.

CAM, işlediği sorgulara ilişkin bilgileri saklamak için CSQ4SAMP.B2.WAITING.*n* formunun adlarıyla kuyrukları kullanır. Kuyruklar, her biri bir yanıtlama kuyruğuyla eşlenmiş olacak şekilde adlandırılır; örneğin, CSQ4SAMP.B2.WAITING.3 kuyruğu belirli bir sorgu için giriş verilerini, CSQ4SAMP.B2.REPLY.3 kuyruğu da aynı sorguyla ilgili olan bir yanıt iletileri kümesini (bu sorgu veritabanlarından oluşan programlar) içerir. Bu tasarımın ardındaki nedenleri anlamak için bkz. ["CAM ' da ayrı sorgu ve yanıt kuyrukları" sayfa 1234.](#)

## **Başlatma mantığı**

Tetikleme olayı CSQ4SAMP.B2.INQUIRY, CAM paylaşılan erişim için kuyruğu açar. Daha sonra, ücretsiz bir yanıt kuyruğu bulununcaya kadar her bir yanıt kuyruğunu açmaya çalışır. Boş bir yanıt kuyruğu bulamazsa, CAM olguları günlüğe kaydeder ve olağan şekilde sonlandırır.

Tetikleme olayı CSQ4SAMP.B2.REPLY.*n*, CAM dışlayıcı erişim için kuyruğu açar. Dönüş kodu, nesnenin önceden kullanımda olduğunu bildiriyorsa, CAM normal şekilde sonlandırılır. Başka bir hata oluşursa, CAM hatayı günlüğe kaydeder ve sonlandırır. CAM karşılık gelen bekleme kuyruğunu ve sorgu kuyruğunu açar, ardından iletileri almaya ve işlemeye başlar. Bekleme kuyruğundan, CAM kısmen tamamlanmış sorguların ayrıntılarını kurtarır.

Bu örnekteki basitlik için, kullanılan kuyrukların adları programda tutulur. İş ortamında, kuyruk adları büyük olasılıkla program tarafından erişilen bir dosyada tutulacaktır.

## Sorgu kuyruğundan ileti alma

CAM, MQGET çağrısını kullanarak sorgu kuyruğundan bir ileti almak için ilk kez MQGMO\_SET\_SIGNAL seçeneğini kullanarak girişimde bulunur. Bir ileti hemen kullanılabiliriyorsa, ileti işlenir; ileti yoksa, bir sinyal ayarlanır.

Daha sonra CAM, aynı seçenekle MQGET çağrısını kullanarak yanıt kuyruğundan bir ileti alma girişiminde bulunur. Bir ileti hemen varsa, ileti işlenir; tersi durumda bir işaret 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 sonlandırılırsa, program sonlandırılır.

## CAM tarafından alınan iletinin işlenmesi

CAM tarafından alınan bir ileti dört tipten biri olabilir:

- Sorgu iletisi
- Yanıt iletisi
- Bir yayma iletisi
- Beklenmeyen ya da istenmeyen bir ileti

CAM, bu iletileri [“Processing the message retrieved by the CAM on z/OS” sayfa 1231](#) içinde açıkladığı gibi işler.

## Yanıt gönderme

CAM bir sorgu beklediği tüm yanıtları aldıysa, yanıtları işler ve tek bir yanıt iletisi oluşturur. Bu, aynı `CorrelId`' e sahip tüm yanıt iletilerinden tüm verileri tek bir iletiye birleştirir. Bu yanıt, özgün kredi isteğinde belirtilen yanıtlama kuyruğuna konmaktadır. Yanıt iletisi, son yanıtlama iletisinin alınmasını içeren aynı iş birimi içinde yer alır. Bu, `CSQ4SAMP.B2.WAITING.n`.

## Kısmen tamamlanmış sorguların kurtarılması

CAM, `CSQ4SAMP.B2.WAITING.n` kuyruğuna, aldığı tüm iletileri kuyruğa alır. İleti açıklayıcısının alanlarını şu şekilde ayarlar:

- *Priority*, ileti tipine göre saptanır:
  - İstek iletileri için öncelik = 3
  - Veri paketleri için öncelik = 2
  - Yanıt iletileri için öncelik = 1
- *CorrelId* is set to the *MsgId* of the loan request message
- Diğer MQMD alanları, alınan iletinin kopyalarından kopyalanır.

Bir sorgu tamamlandığında, belirli bir sorguya ilişkin iletiler yanıt işleme sırasında bekleme kuyruğundan kaldırılır. Bu nedenle, bekleme kuyruğunda devam eden sorgularla ilgili tüm iletiler bekleme kuyruğunda yer alır. Bu iletiler, programın yeniden başlatılması gerekiyorsa, devam etmekte olan sorguların ayrıntılarını kurtarmak için kullanılır. Farklı öncelikler, önzilemeler ya da yanıt iletilerinden önce sorgu iletilerinin kurtarılabilmesi için ayarlanır.

## Processing the message retrieved by the CAM on z/OS

Credit Application Manager (CAM) tarafından alınan bir ileti dört tipten biri olabilir. CAM ' ın bir iletiyi işlemesine ilişkin yöntem, bu iletinin tipine bağlıdır.

CAM tarafından alınan bir ileti dört tipten biri olabilir:

- Sorgu iletisi
- Yanıt iletisi

- Bir yayma iletisi
- Beklenmeyen ya da istenmeyen bir ileti

CAM, bu iletileri aşağıdaki gibi işler:

### **Sorgu iletisi**

Sorgu iletileri, kullanıcı arabirimi programından gelir. Bu, her bir kredi isteği için bir sorgu iletisi oluşturur.

Tüm kredi taleplerinde CAM, müşterinin kontrol hesabının ortalama bakiyecisini istiyor. Bunu, CSQ4SAMP.B2.OUTPUT.ALIAS Bu kuyruk adı, check-account programı CSQ4CVB3 tarafından işlenen CSQ4SAMP.B3.MESSAGES kuyruğuna çözüyor. CAM, bu diğer ad kuyruğuna bir ileti yerleştirdiğinde, uygun CSQ4SAMP.B2.REPLY.n kuyruğu oluştu. Burada bir diğer ad kuyruğu kullanılmıştır; böylece, CSQ4CVB3 programı farklı bir adla temel kuyruğu işleyen başka bir program tarafından değiştirilebilir. Bunu yapmak için, diğer ad kuyruğunu yeniden tanımlıyor ve adı yeni kuyruğa çözüyor. Ayrıca, diğer ad kuyruğuna ve temel kuyruğa farklı erişim yetkileri atayabilirsiniz.

Bir kullanıcı, 10000 biriminden daha büyük bir kredi isteğinde bulunursa, CAM diğer veritabanlarını da denetler. Bu, CSQ4SAMP.B4.MESSAGES, dağıtım programı CSQ4CVB4 tarafından işlenir. Bu kuyruğa hizmet veren süreç, kredi kartı geçmişi, tasarruf hesapları ve konut kredisi ödemeleri gibi diğer kayıtlara erişimi olan programlar tarafından hizmet verilen kuyruklara ileti yayılır. Bu programlardaki veriler, koyma işleminde belirtilen yanıtla kuyruğuna döndürülür. Buna ek olarak, bu program tarafından yanıtla kuyruğuna, kaç tane yayma iletisi gönderildiğini belirtmek için bir yayma iletisi gönderilir.

Bir iş ortamında, dağıtım programı, diğer banka hesabı tiplerinin her birinin gerektirdiği biçimde eşleşmesi için sağlanan verileri büyük olasılıkla yeniden biçimlendirir.

Gönderme yapılan kuyruklardan herhangi biri uzak bir sistemde olabilir.

Her sorgu iletisi için CAM, bellek yerleşik Sorgu Kaydı Çizelgesindeki (IRT) bir girişi başlatır. Bu kayıt şunları içerir:

- Sorgu iletisinin MsgId
- ReplyExp alanında, beklenen yanıt sayısı (gönderilen ileti sayısına eşit)
- ReplyRec alanına, alınan yanıt sayısı (bu aşamada sıfır)
- PropsOut alanında, bir yayma iletisinin beklenip beklenmediğini gösteren bir gösterge

CAM sorgu iletisini bekleme kuyruğuna kopyalıyor:

- Priority , 3 olarak ayarlanır
- CorrelId set to the MsgId of the inquiry message
- Diğer ileti tanımlayıcı alanları, sorgu iletisine ayarlanır.

### **Yayma iletisi**

Bir yayma iletisi, dağıtım programının sorguyu iletildiği kuyrukların sayısını içerir. İleti şu şekilde işlenir:

1. Gönderilen ileti sayısı, IRT ' deki uygun kaydın ReplyExp (ReplyExp) alanına ekleyin. Bu bilgiler iletide yer alıyor.
2. IRT ' deki kaydın ReplyRec alanına 1 değerini artırın.
3. IRT ' deki kaydın PropsOut (PropsOut) alanına göre azalsın.
4. İletiyi bekleme kuyruğuna kopyalayın. CAM, Priority ile 2 arasındaki ve ileti tanımlayıcısının diğer alanlarını yayma iletisininlere ayarlar.

### **Yanıt iletisi**

Yanıt iletisi, denetleme hesabı programına ya da aracı sorgu programlarından birine verilen isteklerden birine yanıt içerir. Yanıt iletileri aşağıdaki gibi işlenir:

1. IRT ' deki kaydın ReplyRec alanına 1 değerini artırın.
2. İletiyi, Priority ile 1 arasındaki bekleme kuyruğuna kopyalayın ve ileti tanımlayıcısının diğer alanları, yanıt iletisine ayarlı olarak ayarlanır.



3. ReplyRec = ReplyExpve PropsOut = 0 ise, MsgComplete işaretinin adını belirleyin.

### Diğer iletiler

Uygulama başka iletiler beklemiyor. Ancak, uygulama sistem tarafından yayınlanan iletileri ya da bilinmeyen bir CorrelIdsile yanıt iletileri alabilir.

CAM, bu iletileri CSQ4SAMP.DEAD.QUEUE, bunlar incelenebilir. Bu put işlemi başarısız olursa, ileti kaybedilir ve program devam eder. Programın bu bölümünün tasarımıyla ilgili daha fazla bilgi için bkz. [“Örnek, beklenmeyen iletileri nasıl işleyeceğini” sayfa 1235.](#)

### z/OS z/OSüzerinde denetleme programı (CSQ4CVB3) denetimi

Denetim hesabı programı, CSQ4SAMP.B3.MESSAGES. Kuyruk açıldıktan sonra, bu program, bekleme seçeneğiyle MQGET çağrısını kullanarak kuyruktan bir ileti alır ve bekleme aralığı 30 saniyeye ayarlanır.

Program, kredi isteği iletilerinde hesap numarası için VSAM veri kümesinde CSQ4BAQ arama kümesini arar. Bu ürün, karşılık gelen hesap adını, ortalama bakiyeyi ve kredi değerliliği dizinini alır ya da hesap numarasının veri kümesinde olmadığı notlarını alır.

Daha sonra program, kredi isteği iletilerinde belirtilen yanıt kuyruğunda (MQPUT1 çağrısını kullanarak) bir yanıt iletileri yerleştirir. Bu yanıt iletileri için program:

- Kredi isteği iletilerinin CorrelId kopyasını kopyalar.
- MQPMO\_PASS\_IDENTITY\_CONTEXT seçeneğini kullanır.

Program, bekleme süresi sona erinceye kadar kuyruktan ileti almaya devam eder.

### z/OS Distribution program (CSQ4CVB4) on z/OS

Dağıtım programı, CSQ4SAMP.B4.MESSAGES.

To simulate the distribution of the loan request to other agencies that have access to records such as credit card history, savings accounts, and mortgage payments, the program puts a copy of the same message on all the queues in the namelist CSQ4SAMP.B4.NAMELIST. There are three of these queues, with names of the form CSQ4SAMP.B n.MESSAGES, where n is 5, 6, or 7. Bir iş uygulamasında, acenteler ayrı konumlarda olabilir, bu nedenle bu kuyruklar uzak kuyruklar olabilir. Örnek uygulamayı bunu gösterecek şekilde değiştirmek istiyorsanız bkz. [“The Credit Check sample with multiple queue managers on z/OS” sayfa 1236.](#)

Dağıtım programı aşağıdaki adımları gerçekleştirir:

1. Ad listesinden, programın kullanmak üzere olduğu kuyrukların adlarını alır. Program, adlist nesnesinin özniteliklerini sorgulamak için MQINQ çağrısını kullanarak bunu yapar.
2. Bu kuyrukları açar ve CSQ4SAMP.B4.MESSAGES.
3. Performs the following loop until there are no more messages on queue CSQ4SAMP.B4.MESSAGES:
  - a. Bekleme seçeneği ile MQGET çağrısını kullanarak ve bekleme süresi 30 saniye olarak ayarlanmış bir ileti alın.
  - b. İlgili CSQ4SAMP.B2.REPLY.n kuyruğu oluştu. Bu program, kredi isteği iletilerinin CorrelId 'unu bu kopyalama iletilerine kopyalar ve MQPUT çağrısında MQPMO\_PASS\_IDENTITY\_CONTEXT seçeneğini kullanır.
  - c. CSQ4SAMP.B2.REPLY.n (REPLAY.n), kaç iletilerin başarıyla yerleştirdiğini gösterir.
  - d. Bir uyumluluk noktası bildirin.

### z/OS Agency-query program (CSQ4CVB5/CSQ4CCB5) on z/OS

Ajans-sorgu programı hem COBOL programı hem de C programı olarak sağlanır. Her iki program da aynı tasarıma sahiptir. Bu, farklı tiplerdeki programların bir IBM MQ uygulaması içinde kolaylıkla birlikte bulunabileceğini ve böyle bir uygulamayı oluşturan program modüllerinin kolayca değiştirilebileceğini gösterir.

Programın bir eşgörünümü, bu 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:** If you want to use the C program, you must alter the definition of the namelist CSQ4SAMP.B4.NAMELIST to replace the queue CSQ4SAMP.B7.MESSAGES with CSQ4SAMP.B8.MESSAGES. Bunu yapmak için aşağıdakilerden birini kullanabilirsiniz:

- IBM MQ for z/OS işlemleri ve denetim panoları
- [ALTER NAMELIST](#) komutu
- [CSQUTIL](#) yardımcı programı

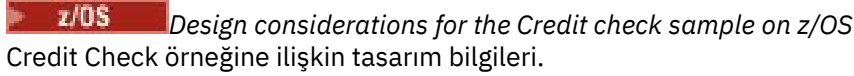
Bu program uygun kuyruğu açtıktan sonra, bekleme seçeneği ile MQGET çağrısını kullanarak kuyruktan bir ileti alır ve bekleme aralığı 30 saniyeye ayarlanır.

Program, kredi isteği iletisinde geçirilen hesap numarası için VSAM veri kümesinde CSQ4BAQ adlı VSAM veri kümesinde arama yaparak, bir dairenin veritabanının aranmasını simüle eder. Daha sonra, hizmet verdiği kuyruğun adını ve bir kredi değeri endeksini içeren bir yanıt oluşturur. İşleme basitleştirmek için, alacaklı elverişlilik endeksi rasgele seçilir.

Yanıt iletisi yerleştirilirken, program MQPUT1 çağrısını kullanır ve:

- Kredi isteği iletisinin `CorrelId` kopyasını kopyalar.
- `MQPMO_PASS_IDENTITY_CONTEXT` seçeneğini kullanır.

Program, yanıt iletisini, kredi isteği iletisinde belirtilen yanıtlanacak kuyruğa gönderir. (Kredi isteği iletisinde, yanıtlama kuyruğuna sahip olan kuyruk yöneticisinin adı da belirtilir.)



Bu konuda aşağıdakiyle ilgili bilgiler yer alır:

- [“CAM ' da ayrı sorgu ve yanıt kuyrukları” sayfa 1234](#)
- [“Örnek tutamaçlarının hataları nasıl” sayfa 1235](#)
- [“Örnek, beklenmeyen iletileri nasıl işleyeceğini” sayfa 1235](#)
- [“Numunenin syncpoints nasıl” sayfa 1235](#)
- [“Örnek, ileti bağlamı bilgilerini nasıl kullanıyor?” sayfa 1236](#)
- [“CAM ' da ileti ve ilinti tanıtıcılarının kullanılması” sayfa 1236](#)

## **CAM ' da ayrı sorgu ve yanıt kuyrukları**

Uygulama hem sorgu hem de yanıtlar için tek bir kuyruk kullanabilir, ancak aşağıdaki nedenlerden dolayı ayrı kuyruklar kullanmak üzere tasarlanmıştır:

- Program, sorgu sayısı üst sınırını işlediğinde, kuyruktan başka sorgular bırakılabilir. Tek bir kuyruk kullanılıyorsa, kuyruğun kapatılması ve başka bir yerde saklanabilmesi gerekir.
- İleti trafiği izin vermek için yeterince yüksekse, aynı sorgu kuyruğuna hizmet vermek için CAM ' nin diğer örnekleri de otomatik olarak başlatılabilir. Ancak program devam etmekte olan sorguları izlemeli ve bunu yapmak için tüm yanıtları başlattığı sorgulara geri dönmelidir. Yalnızca bir kuyruk kullanılırsa, program bu program için mi, yoksa başka bir program mı için iletilere göz atmak zorunda olur. Bu, operasyonu daha az verimli hale getirecektir.

Uygulama birden çok CAM' yi destekleyebilir ve eşleştirilmiş yanıtlama ve bekleme kuyrukları kullanılarak devam etmekte olan sorguları etkili bir şekilde kurtarabilir.

- Program, sinyalizasyonu kullanarak birden çok kuyrukta etkili bir şekilde bekleyebilir.

## Örnek tutamaçlarının hataları nasıl

Kullanıcı arabirimi programı, hataları doğrudan kullanıcıya bildirerek hataları işler.

Diğer programlarda kullanıcı arabirimleri yoktur, bu nedenle hataları başka şekillerde ele almak zorunda kalmazlar. Ayrıca, birçok durumda (örneğin, bir MQGET çağrısı başarısız olursa) bu diğer programlar, uygulama kullanıcısının kimliğini bilmiyor.

The other programs put error messages on a CICS temporary storage queue called CSQ4SAMP. You can browse this queue using the CICS-supplied transaction CEBR. Programlar, CICS CSML günlüğüne hata iletileri de yazar.

## Örnek, beklenmeyen iletileri nasıl işleyeceğini

Bir ileti kuyruklama uygulaması tasarladığınızda, beklenmedik bir şekilde kuyruğa gelen iletilerin nasıl işleneceğine karar vermelisiniz.

İki temel seçenek vardır:

- Uygulama, beklenmeyen iletiyi işleinceye kadar daha fazla iş yapmaz. Bu, uygulamanın bir operatöre bildirdiği, kendisini sonlandırdığı ve otomatik olarak yeniden başlatılmamasını sağladığı anlamına gelir (bu işlemi tetiklemeyi ayarlayarak yapabilir). Bu seçenek, uygulamaya ilişkin tüm işlemlerin tek bir beklenmeyen ileti tarafından durdurulabileceği ve uygulamanın yeniden başlatılması için bir işletmenin müdahalesini gerektireceği anlamına gelir.
- Uygulama, iletiyi hizmet ettiği kuyruktan kaldırır, iletiyi başka bir yere koyar ve işlemeye devam eder. Bu iletiyi koymak için en iyi yer sistem ölüm mektubu kuyruğunda yer alıyor.

İkinci seçeneği belirlerseniz:

- İletilerin nereden geldiğini bulmak için, bir işletmen ya da başka bir program, ölü-mektup kuyruğuna yerleştirilecek iletileri incelemelidir.
- Beklenmeyen bir ileti, ölü-mektup kuyruğuna konulamazsa kaybedilir.
- Uzun süredir beklenmeyen bir ileti kesildi. Bu ileti, programdaki arabellek büyüklüğünden daha uzun ya da daha uzun kuyruklardaki iletiler için sınırdan uzunsa kesilir.

Uygulamanın dış etkinliklerden en az etkiye sahip tüm sorguları sorunsuz bir şekilde işlediğinden emin olmak için, Credit Check örnek uygulaması ikinci seçeneği kullanır. Aynı kuyruk yöneticisini kullanan diğer uygulamalardan örneği ayrı tutmanıza izin vermek için, Credit Check (Kredi Denetimi) numunesi sistem ölü harf kuyruğunu kullanmaz; bunun yerine, kendi ölü harf kuyruğunu kullanır. Bu kuyruk CSQ4SAMP.DEAD.QUEUE. Örnek, örnek programlar için sağlanan arabellek alanından daha uzun olan iletileri kısaltma. Bu kuyruktaki iletilere göz atmak için Göz At örnek uygulamasını kullanabilir ya da iletileri ileti açıklayıcılarıyla birlikte yazdırmak için İleti Yazdırma örneği uygulamasını kullanabilirsiniz.

Ancak, örneği birden çok kuyruk yöneticisi, beklenmeyen iletiler ya da teslim edilemeyen iletiler arasında çalışacak şekilde genişletiyorsanız, kuyruk yöneticisi tarafından sistem ölümüne ilişkin kuyruğu yerleştirilebilir.

## Numunenin syncpoinces nasıl

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 örneğe ait ölü harf kuyruğuna konmaz.
- CAM, bekleme kuyruğundan kalıcı iletiler olarak, kısmen tamamlanmış tüm sorguların durumunu kurtarabilir.

Bunu başarmak için, tek bir iş birimi, bir iletinin alınması, bu iletinin işlenmesi ve izleyen herhangi bir sonraki işlemler için kullanılır.

## Örnek, ileti bağlamı bilgilerini nasıl kullanıyor?

Kullanıcı arabirimi programı (CSQ4CVB1) ileti gönderdiğinde, MQPMO\_DEFAULT\_CONTEXT seçeneğini kullanır. Bu, kuyruk yöneticisinin hem kimlik, hem de kaynak bağlamı bilgilerini oluşturduğu anlamına gelir. Kuyruk yöneticisi, programı başlatan işlemden (MVB1) ve hareketi başlatan kullanıcı kimliğinden bu bilgileri alır.

CAM sorgu iletileri gönderdiğinde, bu ileti MQPMO\_PASS\_IDENTITY\_CONTEXT seçeneğini kullanır. Bu, yerleştirmekte olan iletinin kimlik bağlamı bilgilerinin, özgün sorgu iletisinin kimlik bağlamından kopyalandığı anlamına gelir. Bu seçenikle, başlangıç noktası bağlamı kuyruk yöneticisi tarafından oluşturulur.

CAM yanıt iletileri gönderdiğinde, bu ileti MQPMO\_ALTERNATE\_USER\_AUTHORITY seçeneğini kullanır. Bu, kuyruk yöneticisinin CAM bir yanıt kuyruğu açtığında güvenlik denetimi için başka bir kullanıcı kimliği kullanmasına neden olur. CAM, özgün sorgu iletisini göndericinin kullanıcı kimliğini kullanır. Bu, kullanıcıların yalnızca kaynaklandığı sorgularla ilgili yanıtları görmelerine izin verileceği anlamına gelir. Diğer kullanıcı kimliği, özgün sorgu iletisinin ileti tanımlayıcısındaki kimlik bağlamı bilgilerinden elde edilir.

Sorgu programları (CSQ4CVB3/4/5) yanıt iletileri gönderdiğinde, bunlar MQPMO\_PASS\_IDENTITY\_CONTEXT seçeneğini kullanır. Bu, yerleştirmekte olan iletinin kimlik bağlamı bilgilerinin, özgün sorgu iletisinin kimlik bağlamından kopyalandığı anlamına gelir. Bu seçenikle, başlangıç noktası bağlamı kuyruk yöneticisi tarafından oluşturulur.

**Not:** MVB3/4/5 işlemleriyle ilişkili kullanıcı kimliği, B2.REPLY.n kuyruklar. Bu kullanıcı kimlikleri, işlenmekte olan istekle ilişkili işlemlerle aynı olmayabilir. Bu olası güvenlik açıklarını almak için, sorgu programları yanıtlarını koyarken MQPMO\_ALTERNATE\_USER\_AUTHORITY seçeneğini kullanabilir. Bu, MVB1 ' un her bir kullanıcılarında B2.REPLY.n kuyruklarını açmak için gereken yetkiye sahip olması anlamına gelir.

## CAM ' da ileti ve ilinti tanıtıcılarının kullanılması

Uygulama, herhangi bir zamanda işlenmekte olduğu tüm canlı sorguların ilerleyişini izlemek zorundadır. Bunu yapmak için her bir kredi isteği iletisinin benzersiz ileti tanıtıcısını kullanır. Bu ileti, her bir sorgu hakkında sahip olduğu tüm bilgileri ilişkilendirir.

CAM, sorgu iletisinin MsgId 'unu, ilgili sorgu için gönderdiği tüm istek iletilerinin CorrelId ' ine kopyalar. Örnekteki diğer programlar (CSQ4CVB3 -5),receive'un yanıt iletisinin CorrelId ' e aldıkları her iletinin CorrelId dosyasını kopyaladıklarını sağlar.

### *The Credit Check sample with multiple queue managers on z/OS*

Credit Check örnek uygulamasını kullanarak, iki kuyruk yöneticisine ve CICS sistemine (farklı bir CICS sistemine bağlı her kuyruk yöneticisi ile) örnek kurarak dağıtılmış kuyruklama örneği gösterebilirsiniz.

Örnek program kurulduğunda ve tetikleme izleme programı (CKTI) her bir sistemde çalışıyorsa, şunları yapmak gerekir:

1. İki kuyruk yöneticisi arasındaki iletişim bağlantısını ayarlayın. Bunun nasıl yapacağına ilişkin bilgi için [Dağıtılmış kuyruklama yapılandırılması](#) başlıklı konuya bakın.
2. Bir kuyruk yöneticisinde, kullanmak istediğiniz her uzak kuyruğun (diğer kuyruk yöneticisinde) her biri için yerel bir tanımlama yaratın. Bu kuyruklar CSQ4SAMP.B n' den herhangi biri olabilir. ILETILER; burada n 3, 5, 6 ya da 7 'dir. (bunlar, check-account programı ve ajans-sorgu programı tarafından hizmet verilen kuyruklar.) Bunun nasıl yapacağına ilişkin bilgi için bkz. [DEFINE QREMOTE](#) ve [DEFINE kuyrukları](#).
3. Ad listesinin tanımını değiştirin (CSQ4SAMP.B4.NAMELIST) işlevi, kullanmak istediğiniz uzak kuyrukların adlarını içerir. Bunun nasıl yapacağına ilişkin bilgi için bkz. [DEFINE NAMELIST](#).

### *z/OS üzerindeki Credit Check örneğine IMS uzantısı*

A version of the checking-account program is supplied as an IMS batch message processing (BMP) program. C dilinde yazılmış.

The program performs the same function as the CICS version, except that to obtain the account information, the program reads an IMS database instead of a VSAM file. Check-account programının CICS sürümünü IMS sürümüyle değiştirirseniz, uygulamayı kullanma yönteminde bir fark yoktur.

IMS sürümünü hazırlamak ve çalıştırmak için aşağıdakileri yapmak gerekir:

1. Follow the steps in “z/OS' ta Kredi Denetimi örneğini hazırlama ve çalıştırma” sayfa 1227.
2. Follow the steps in “Preparing the sample application for the IMS environment on z/OS” sayfa 1208.
3. CSQ4SAMP.B3.IMSkuyruğuna kadar çözümlmek için CSQ4SAMP.B2.OUTPUT.ALIAS diğer ad kuyruğunun tanımlamasını değiştirin.İLETİLER ( CSQ4SAMP.B3.MESSAGESyerine). Bunu yapmak için aşağıdakilerden birini kullanabilirsiniz:
  - IBM MQ for z/OS işlemleri ve denetim panoları
  - ALTER QALIAS komutu.

IMS check-account programını kullanmanın başka bir yolu da, dağıtım programından ileti alan kuyruklardan birine hizmet etmek. Credit Check örnek uygulamasının teslim edilen formunda, bu kuyruklardan üç tanesi bulunur (B5/6/7.MESSAGES), tüm kurum-sorgu programı tarafından hizmet verdi. Bu program bir VSAM veri kümesinde arama yapar. VSAM veri kümesinin ve IMS veritabanının kullanımını karşılaştırmak için, IMS check-account programını bu kuyruklardan birine hizmet verebilirsiniz. To do this, you must alter the definition of the namelist CSQ4SAMP.B4.NAMELIST to replace one of the CSQ4SAMP.B n.MESSAGES queues with the CSQ4SAMP.B3.IMS.MESSAGES queue. Aşağıdakilerden birini kullanabilirsiniz:

- IBM MQ for z/OS işlemleri ve denetim panoları
- ALTER NAMELIST komutu.

Daha sonra, örneği CICS hareketinden MVB1' den çalıştırabilirsiniz. Kullanıcı, işlem ya da yanıtta hiçbir fark görmez. IMS BMP, bir durdurma iletisi aldıktan sonra ya da beş dakika devre dışı kaldıktan sonra durur.

## IMS denetleme-hesap programının tasarımı (CSQ4ICB3)

Bu program bir BMP olarak çalıştırılır. Programı, herhangi bir IBM MQ iletisi gönderilmeden önce JCL ' yi kullanarak başlatın.

Program, kredi isteği iletilerindeki hesap numarası için bir IMS veritabanında arama yapar. Bu ürün, karşılık gelen hesap adını, ortalama bakiyeyi ve kredi değerliliği endeksini alır.

Program, işlenmekte olan IBM MQ iletisinde belirtilen yanıtlama kuyruğuna veritabanı araması sonuçlarını gönderir. Döndürülen ileti, hesap tipini ve alınan iletiye yapılan aramanın sonuçlarını, yanıtı doğru sorgunun işlenmekte olduğunu doğrulayabilecek şekilde iletir. Bu ileti, aşağıdaki gibi olmak üzere üç adet 79 karakterlik grup biçiminde yer alıyor:

```
'Response from CHECKING ACCOUNT for name : JONES J B'  
'  Opened 870530, 3-month average balance = 000012.57'  
'  Credit worthiness index - BBB'
```

When running as a message-oriented BMP, the program drains the IMS message queue, then reads messages from the IBM MQ for z/OS queue and processes them. IMS ileti kuyruğundan herhangi bir bilgi alınmıyor. Çekme noktaları kapatıldığı için, program her denetim noktasından sonra kuyruk yöneticisine yeniden bağlanır.

Toplu iş odaklı bir BMP ' de çalışırken, tutamaçlar kapatılmadığından, program her denetim noktasından sonra kuyruk yöneticisine bağlanmaya devam eder.

## The Message Handler sample on z/OS

İleti İşleyici örneği TSO uygulaması, bir kuyruktaki iletileri göz atarak, iletmenize ve silmenize olanak sağlar. Örnek, C ve COBOL ' de kullanılabilir.

## Örneği hazırlama ve çalıştırma


Aşağıdaki adımları izleyin:

1. Örneği, "[z/OS üzerinde TSO ortamı için örnek uygulamalar hazırlama](#)" sayfa 1203 içinde açıklandığı gibi hazırlayın.
2. Örneğin, panoların yerini, ileti dosyasının yerini ve yükleme modüllerinin yerini tanımlamak için CLIST ' i (CSQ4RCH1) uyarlayın.

Örneğin, hem C hem de COBOL sürümünü çalıştırmak için CLIST CSQ4RCH1 ' i kullanabilirsiniz. The supplied version of CSQ4RCH1 runs the C version, and contains instructions on the tailoring necessary for the COBOL version.

### Not:

1. Örnekle birlikte sağlanan bir örnek kuyruk tanımlaması yok.
2. VS COBOL II, ISPF ile çoklu görevi desteklemez; bu nedenle, İleti İşleyici örnek uygulamasını, bölünmüş bir ekranın her iki yanında kullanmayın. Bunu yapmazsanız, sonuçlar tahmin edilemez.

 *Using the Message Handler sample on z/OS*

Örneği taktıktan ve uyarlanan CLIST CSQ4RCH1' den çağırılmış olması, [Şekil 151 sayfa 1238](#) içinde gösterilen ekranın görüntülendiğini göstermektedir.

```
----- 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 151. İleti İşleyici örneği için ilk ekran*

Görüntülenecek kuyruk yöneticisini ve kuyruk adını girin (büyük ve küçük harfe duyarlı) ve ileti listesi ekranı görüntülenir (bkz. [Şekil 152 sayfa 1239](#)).

```
----- 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 152. İleti İşleyici örneği için ileti listesi ekranı

Bu ekranda kuyruğun ilk 99 iletisi gösterilir ve her biri için aşağıdaki alanlar gösterilir:

**İlt No**

İleti numarası

**Bitiş Tarihi AA/GG/YYYY**

İletinin kuyruğa konacağı tarih (GMT)

**Put Time HH:MM: SS**

İletinin kuyruğa konması için saat (GMT)

**Biçim Adı**

MQMD.Format alanı

**Kullanıcı Kimliği**

MQMD.UserIdentifier alanı

**Koyma Uygulaması Tipi**

MQMD.PutApplType alanı

**Koyma Uygulaması Adı**

MQMD.PutApplName alanı

Kuyruktaki toplam ileti sayısı da görüntülenir.

Bu ekrandan bir ileti, imleç konumuna göre değil, sayı olarak seçilebilir ve görüntülenebilir. Bir örnek için bkz. [Şekil 153](#) sayfa 1240.

```

----- 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 : `000000000000000000000000000000000000000000000000`X
BackoutCount : 000000000
ReplyToQ : `QL.TEST.ISCRES1
ReplyToQMgr : `VM03
UserIdentifier : `NTSFV02
AccountingToken :
`06F2F5F5F3F0F1000000000000000000000000000000000000000000000000`X
AppIdentityData :
PutApplType : 000000002
PutApplName : `NTSFV02A
PutDate : `19971016`
PutTime : `13511903`
AppLOriginData :

Message Buffer : 108 byte(s)
00000000 : C9C9 C840 0000 0001 0000 0054 0000 0311 `IIH .....`
00000010 : 0000 0000 4040 4040 4040 4040 0000 0000 `.....`
00000020 : 4040 4040 4040 4040 4040 4040 4040 4040 `.....`
00000030 : 4040 4040 4040 4040 4040 4040 4040 4040 `.....`
00000040 : 0000 0000 0000 0000 0000 0000 0000 0000 `.....`
00000050 : 40F1 C300 0018 0000 C9C1 D7D4 C4C9 F2F8 `1C....IAPMDI28`
00000060 : 40C8 C5D3 D3D6 40E6 D6D9 D3C4 `HELLO WORLD`
***** Bottom of data *****

```

### Şekil 153. Seçilen ileti görüntülenir

İleti görüntüledikten sonra, silinebilir, kuyruktan bırakılabilir ya da başka bir kuyruğa iletilebilir. Forward to Q Mgr ve Forward to Queue alanları, MQMD 'deki değerlerle ilk kullanıma hazırlanır. Bunlar, iletiyi iletmeden önce değiştirilebilir.

Bu ileti, anahtar olarak MsgId ve CorrelId kullanılarak ileti alındığından, yalnızca benzersiz MsgId / CorrelId birleşimleri olan iletilerin seçilmesine ve görüntülenmesine olanak sağlar. Anahtar benzersiz değilse, örnek seçilen iletiyi kesinlik ile alamıyor.

**Not:** İletilere göz atmak için SCSQCLST (CSQ4RCH1) örneğini kullandığınızda, her çağırma, iletinin geriletme sayısının artmasına neden olur. Bu örneğe ilişkin davranışı değiştirmek istiyorsanız, örneği kopyalayın ve içeriği gerektiği gibi değiştirin. Bu geri alma sayısına güvenen diğer uygulamaların bu artan sayıdan etkilenebileceğinin farkında olmalısınız.

### z/OS z/OS üzerinde örnek Message Handler örneğinin tasarımı

Bu konuda, Message Handler örnek uygulamasını oluşturan programların her birinin tasarımı açıklanmaktadır.



## Nesne geçerlilik denetimi programı

Bu, geçerli bir kuyruk ve kuyruk yöneticisi adı ister.

Kuyruk yöneticisi adı belirtmezseniz, varsayılan kuyruk yöneticisi kullanılır (varsa). Yalnızca yerel kuyruklar kullanılabilir; kuyruk tipinin denetlenmesi için bir MQINQ yayınlandı ve kuyruk yerel değilse bir hata raporlanır. Kuyruk başarıyla açılmamışsa ya da kuyruğa ilişkin MQGET çağrısını engelliyorsa, CompCode ve neden dönüş kodunu gösteren hata iletileri döndürülür.

## İleti listesi programı

Bu, bir kuyruktaki iletilerin listesini, putdate, puttime ve ileti biçimi gibi kendileriyle ilgili bilgileri içeren bir kuyruktaki görüntüler.

Listede saklanan ileti sayısı üst sınırı 99 'tır. Kuyrukda bundan daha fazla ileti varsa, yürürlükteki kuyruk derinliği de görüntülenir. Görüntülenecek bir iletiyi seçmek için, ileti numarasını giriş alanına yazın (varsayılan değer 01 'dir). Giriniz geçerli değilse, uygun bir hata iletisi alırsınız.

## İleti içerik programı

Bu ileti, ileti içeriğini görüntüler.

İçerik biçimlendirilir ve iki kısma bölünür:

1. İleti tanımlayıcısı
2. İleti arabelleği

İleti tanımlayıcısı, her alanın içeriğini ayrı bir çizgide gösterir.

İleti arabelleği, içinekinine bağlı olarak biçimlendirilir. Arabellek bir ölün harf üstbilgisi (MQDLH) ya da bir iletim kuyruğu üstbilgisi (MQXQH) bulunduruyorsa, bunlar arabelleğin kendisinden önce biçimlendirilir ve görüntülenir.

Arabellek verileri biçimlendirilmeden önce, bir başlık satırı, iletinin bayt cinsinden arabellek uzunluğunu gösterir. Arabellek büyüklüğü üst sınırı 32768 bayttır ve bu iletiden daha uzun bir ileti kesilir. Arabelleğin tam büyüklüğü, iletinin yalnızca ilk 32768 baytının görüntülendiğini gösteren bir iletiyle birlikte görüntülenir.

Arabellek verileri iki şekilde biçimlendirilir:

1. Arabellek değeri yazdırıldıktan sonra, arabellek verileri onaltılı olarak görüntülenir.
2. Arabellek verileri EBCDIC değerleri olarak yeniden görüntülenir. Herhangi bir EBCDIC değeri yazdırılmazsa, bunun yerine bir nokta (.) yazdırılır.

Silme işlemi için D ya da işlem alanına F için F girebilirsiniz. İletiyi iletmeyi seçerseniz, `forward-to queue` ve `queue manager` name doğru ayarlanmış olmalıdır. The defaults for these fields are read from the message descriptor `ReplyToQ` and `ReplyToQMgr` fields.

Bir iletiyi iletteyorsa, arabelleğde saklanan herhangi bir üstbilgi öbeği sökülür. İleti başarıyla iletirse, özgün kuyruktan kaldırılır. Geçersiz işlemler giriyorsanız, hata iletileri görüntülenir.

CSQ4CHP9 adlı bir yardım panosu örneği de kullanılabilir.

## z/OSüzerinde Zamanuyumsuz Koyma örneği

Zamanuyumsuz put örnek programı, zamanuyumsuz MQPUT çağrısını kullanarak bir kuyruğa ileti koyar. Örnek, MQSTAT çağrısını kullanarak durum bilgilerini de alır.

Zamanuyumsuz put uygulamaları bu MQI çağrılarını kullanır:

- MQCONN
- MQOPEN
- MQPUT
- MQSTAT

- MQCLOSE
- MQDISC

Örnek programlar C programlama dilinde teslim edilir.

Zamanuyumsuz put uygulamaları toplu iş ortamında çalışır. Toplu iş uygulamaları için [Diğer örnekler](#) konusuna bakın.

Bu konuda, Asynchronous Consumption programının tasarımı hakkında bilgi ve CSQ4BCS2 örneği çalıştırılmasıyla ilgili bilgiler de sağlanır.

- [“CSQ4BCS2 örneğinin çalıştırılması” sayfa 1242](#)
- [“Zamanuyumsuz Put örnek programının tasarımı” sayfa 1242](#)

## CSQ4BCS2 örneğinin çalıştırılması

Bu örnek program altı parametrelere kadar sürer:

1. Hedef kuyruğun adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. Seçenekleri açın (isteğe bağlı).
4. Seçenekleri kapat (isteğe bağlı).
5. Hedef kuyruk yöneticisinin adı (isteğe bağlı).
6. Dinamik kuyruğun adı (isteğe bağlı).

Kuyruk yöneticisi belirtilmediyse, CSQ4BCS2 varsayılan kuyruk yöneticisine bağlanır. İleti içeriği standart giriş yoluyla sağlanır ( **SYSD** ).

Programı çalıştırmak için bir JCL örneği vardır; bu örnek CSQ4BCSP' de bulunur.

## Zamanuyumsuz Put örnek programının tasarımı

Program, iletileri koymak üzere hedef kuyruğu açmak için, sağlanan çıkış seçenekleri ya da MQOO\_OUTPUT ve MQOO\_FAIL\_IF\_QUIESCING seçenekleri ile MQOPER çağrısını kullanır.

Program kuyruğu açamazsa, program MQOPEN çağrısının döndürdüğü neden kodunu içeren bir hata ileti görüntüler. Programı bu konuda basit tutmak ve sonraki MQI çağrılarını yapmak için, seçeneklerin çoğu için varsayılan değerler kullanılır.

Her giriş satırı için, program metni bir arabelleğe okur ve MQPUT çağrısını kullanarak, o satırın metnini içeren bir veri paketi ileti yaratmak için MQPUT çağrısını kullanır ve iletiyi hedef kuyruğa zamanuyumsuz olarak yerleştirir. Program, girişin sonuna ulaşıncaya kadar ya da MQPUT çağrısının başarısız oluncaya kadar devam eder. Program girişin sonuna ulaşırsa, MQCLOSE çağrısını kullanarak kuyruğu kapatır.

Daha sonra, program bir MQSTS yapısını döndüren MQSTAT çağrısını yayınlar ve iletilerin sayısını başarıyla içeren iletileri, uyarı içeren ileti sayısını ve başarısızlıkların sayısını görüntüler.

**Not:** MQSTAT çağrısı tarafından bir MQPUT hatası saptandığında ne olduğunu gözlemlemek için, hedef kuyrukdaki MAXDEPTH değerini düşük bir değere ayarlayın.

## **The Batch Asynchronous Consumption sample on z/OS**

CSQ4BCS1 örnek programı C 'de teslim edilir; MQCB ve MQCTL' nin birden çok kuyruktan gelen iletileri zamanuyumsuz olarak tüketmesini gösterir.

Zamanuyumsuz tüketim örnekleri toplu iş ortamında çalışır. Toplu iş uygulamaları için [Diğer örnekler](#) konusuna bakın.

Ayrıca, CICS ortamında çalışan bir COBOL örneği de vardır, bkz. [“z/OSüzerinde CICS Zamanuyumsuz Tüketimi ve Yayınlama/Abone Olma örneği” sayfa 1244](#).

Uygulamalar bu MQI çağrılarını kullanır:

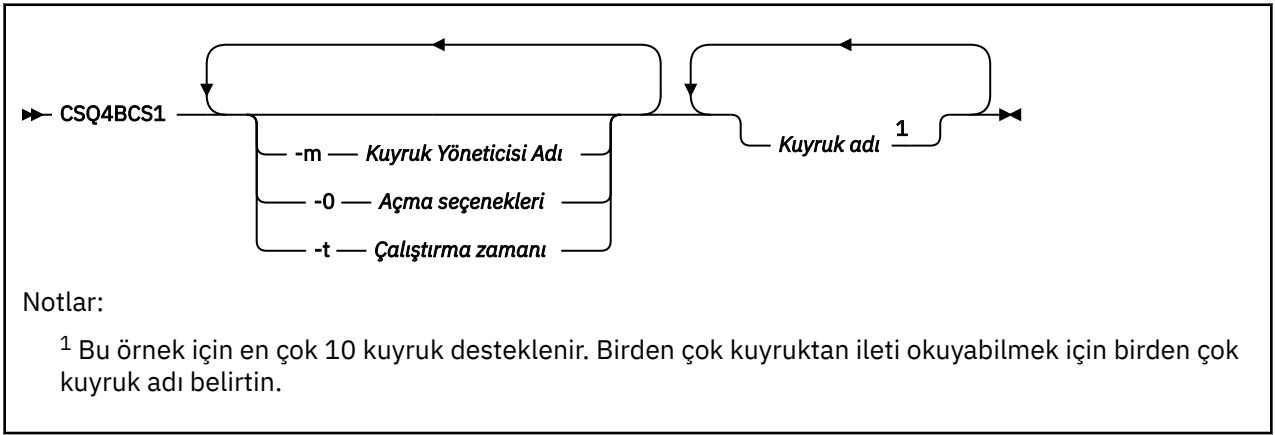
- MQCONN
- MQOPEN
- MQCLOSE
- MQDISC
- MQCB
- MQCTL

Bu konu ayrıca aşağıdaki başlıklarla ilgili bilgiler de sağlar:

- [“CSQ4BCS1 örneğinin çalıştırılması” sayfa 1243](#)
- [“Toplu Zamanuyumsuz Tüketim örnek programının tasarımı” sayfa 1243](#)

## CSQ4BCS1 örneğinin çalıştırılması

Bu örnek program aşağıdaki sözdizimini izler:



Bu programı çalıştırmak için bir JCL örneği vardır; CSQ4BCSC' de bulunur.

## Toplu Zamanuyumsuz Tüketim örnek programının tasarımı

Bu örnek, geliş sırasına göre birden çok kuyruktan iletilerin nasıl okunacağını gösterir. Bu, zamanuyumlu MQGET kullanarak daha fazla kod gerektirecektir. Zamanuyumsuz tüketimle birlikte yoklama gerekmez, iş parçacığı ve depolama yönetimi IBM MQ tarafından gerçekleştirilir. Örnek programda, konsolda hatalar yazılır.

Örnek kod aşağıdaki adımları içerir:

1. Tek ileti tüketimi geri bildirme 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 çağrı işleviyle ilişkilendirin.

```
MQOPEN(Hcon, &od, 0_options, &Hobj, &OpenCode, &Reason);
cbd.CallbackFunction = MessageConsumer;
MQCB(Hcon, MQOP_REGISTER, &cbd, Hobj, &md, &gmo, &CompCode, &Reason);
```

cbd.CallbackFunction , her kuyruk için ayarlanması gerekmez; bu yalnızca giriş alanıdır. Farklı bir geri bildirme işlevini her kuyrukla ilişkilendirebilirsiniz.

#### 4. İletilerin tüketimine başla.

```
MQCTL(Hcon,MQOP_START,&ctlo,&CompCode,&Reason);
```

#### 5. Kullanıcının Enter tuşuna basmasını bekleyin ve sonra ileti tüketimini durdurun.

```
MQCTL(Hcon,MQOP_STOP,&ctlo,&CompCode,&Reason);
```

#### 6. Son olarak, kuyruk yöneticisinden bağlantıyı kesin.

```
MQDISC(&Hcon,&CompCode,&Reason);
```

## z/OSüzerinde CICS Zamanuyumsuz Tüketimi ve Yayınlama/Abone Olma örneği

The Asynchronous Consumption and Publish/Subscribe sample programs demonstrate the use of asynchronous consumption, and publish and subscribe features within CICS.

Bir *Kayıt istemcisi* programı üç Callback işleyiciyi (olay işleyicisi ve iki ileti tüketicisi) kaydeder ve Asynchronous Consumption (Zamanuyumsuz Tüketimi başlatır) başlatır. *İleti alışverişi istemcisi* programı, iletileri bir kuyruğa koyar ya da iki Message Consumers (CSQ4CVCN ve CSQ4CVCT) tarafından tüketime ilişkin bir CICS konsolundan uygun iletileri yayınlar.

Örnek davranış üzerine çalıştırma zamanı denetimini sağlamak için, ileti tüketicilerinden biri, Callback işleyicilerinden herhangi birine, aldığı iletiler, RASPEND, RESUME ya da DEREGISTER iletileri kullanılarak yönerge verilecektir. Bu, denetim altında Zamanuyumsuz Tüketimi sona erdirmek için bir MQCTL STOP yayınlamak için de kullanılabilir. Diğer ileti tüketicisi bir konuya abone olmak için kayıtlıdır.

Her program, örnekteki davranışını görüntülemek için uygun noktalarda COBOL DISPLAY deyimlerini yayınlar.

Uygulamalar bu MQI çağrılarını kullanır:

- MQOPEN
- MQPUT
- MQSUB
- MQGet
- MQCLOSE
- MQCB
- MQCTL

Programlar, COBOL dilinde teslim edilir. CICS uygulamaları için [CICS Zamanuyumsuz Tüketim ve Yayınlama/Abone Olma örnekleri](#) başlıklı konuya bakın.

Bu konu ayrıca aşağıdaki bilgileri de sağlar:

- [“Ayarla” sayfa 1245](#)
- [“Kayıt İstemcisi CSQ4CVRG” sayfa 1245](#)
- [“Olay işleyici CSQ4CVEV” sayfa 1245](#)
- [“Basit İleti Tüketicisi CSQ4CVCN” sayfa 1245](#)
- [“Denetim İletisi Tüketicisi CSQ4CVCT” sayfa 1245](#)
- [“İleti Sistemi İstemcisi CSQ4CVPT” sayfa 1245](#)

## Ayarla

İleti Tüketicileri tarafından kullanılan Kuyruk ve Konu adları, Kayıt ve İleti Sistemi İstemcisi programlarında değişmez olarak kodlanmıştır.

Kuyruk, **SAMPLE.CONTROL.QUEUE**, örneği çalıştırmadan önce CICS bölgesiyle ilişkilendirilmiş Kuyruk Yöneticisi olarak tanımlanmalıdır. The Topic, **Haberler/Medya/Movies**, can be defined if required, or it is created at runtime under the default Administrative Object if it does not exist.

Bir grup kurularakCICS programları ve işlem tanımlamaları kurulabilir: CSQ4SAMP.

## Kayıt İstemcisi CSQ4CVRG

Kayıt İstemcisi programı, CICS hareketi MVRG altında başlatılmalıdır. Giriş yok.

Kayıt İstemcisi başlatıldığında, aşağıdaki Callback işleyicilerini MQCB kullanarak kaydettirir:

- CSQ4CVEV as an Event Handler.
- CSQ4VCVN as a Message Consumer on a topic, **Haberler/Medya/Movies**.
- Kuyruksa İleti Tüketicisi olarakCSQ4CVCT , **SAMPLE.CONTROL.QUEUE**.

Kayıt İstemcisi, kayıtlı tüm üç Callback işleyicilerinin adlarını içeren bir veri yapısını, iki ileti tüketicisiyle ilişkili nesne tanıtıcıları ile birlikte CSQ4CVCT' e aktarır.

Callback işleyicilerini kaydettirmek için, Kayıt İstemcisi, Zamanuyumsuz Tüketimi başlatmak için MQCTL START\_WAN ile bir MQCTL START\_SATT, denetim ona geri verilinceye kadar askıya alır (örneğin, bir MQCTL STOP veren Callback işleyicilerinden biri tarafından).

## Olay İşleyici CSQ4CVEV

Event Handler (Olay İşleyici) çağrıldığında, çağrı tipini (örneğin, START) gösteren bir ileti görüntüler. IBM MQ neden kodu CONNECTION\_QUIESCING için yönlendirildiğinde, Olay İşleyici, Zamanuyumsuz Tüketimi sona erdirmek için bir MQCTL STOP ve Kayıt İstemcisi 'ye geri dönüş denetimini verir.

## Basit İleti Tüketicisi CSQ4VCVN

Bu İleti Tüketicisi çalıştırıldığında, çağrı tipini gösteren bir ileti görüntüler (örneğin, REGISTER). When driven for the MSG\_REMOVED call type, the Message Consumer retrieves the inbound message and outputs it to the CICS job log.

## Denetim İletisi Tüketicisi CSQ4CVCT

Bu İleti Tüketicisi çalıştırıldığında, çağrı türünü (örneğin, START) belirten bir ileti görüntüler. MSG\_KESNE çağrı tipi için yönlendirildiğinde, Message Consumer gelen iletiyi ve Kayıt İstemcisi tarafından geçirilen veri yapısını alır. İleti içeriğine dayalı olarak, aşağıdaki komutlardan birine uygun MQCB ya da MQCTL komutları yayınlar:

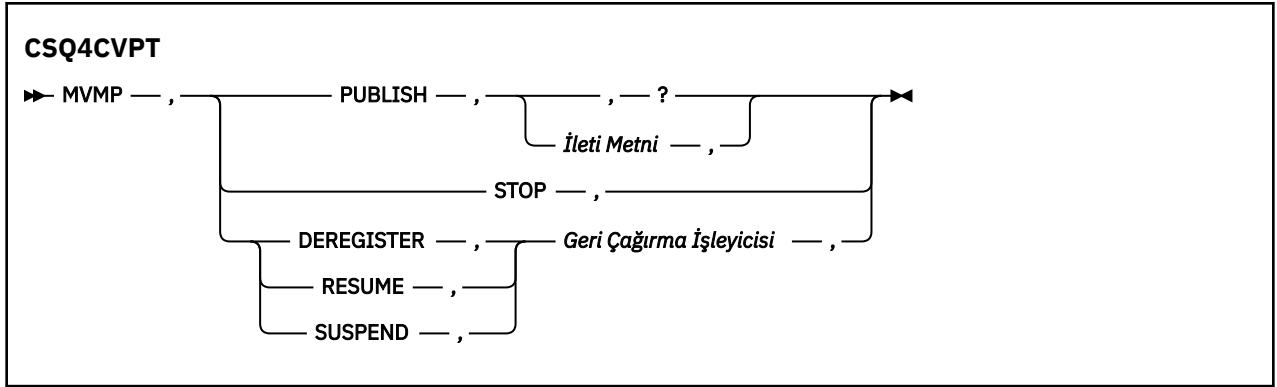
- STOP Zamanuyumsuz Tüketimi (Kayıt İstemcisi 'ne geri döndürme).
- SUSPEND, RESUME ya da Deregister adlı bir Callback işleyicisi (kendisi de içinde olmak üzere)

## İleti Sistemi İstemcisi CSQ4CVPT

Messaging Client 'ın iki işlevi vardır:

- İleti Tüketicisi CSQ4VCVNtarafından tüketime ilişkin bir konuya ileti yayınlar.
- Denetim Message Consumer CSQ4CVCTtarafından tüketime ilişkin bir kuyruğa bir denetim iletisi koyar ve bu, örneğin davranışında olası bir değişiklik ortaya çıkar.

Messaging Client programı, CICS işlemi altındaki bir CICS konsolundan başlatılmalı ve aşağıdaki sözdizimiyle komut satırı girişini alır:



### PUBLISH

İleti metnini (ya da varsayılan bir iletiyi) Yalın İleti Tüketicisi tarafından kullanım için Eğilenmiş İleti olarak yayınlayın.

### STOP

Zamanuyumsuz Tüketimi Durdurun.

### DEREGISTER

Adlandırılmış Callback işleyicisini kayıttan çıkar.

### Süzdür

Adlandırılmış Callback işleyicisini süzdürün.

### Askıya al

Adlandırılmış Callback işleyicisini askıya alın.

Giriş alanları konumlu ve virgülle ayrılmış olarak bulunur. Anahtar sözcükler ve Callback Handler adları büyük/küçük harfe duyarlı değildir.

Örnekler:

Çizelge 193. Giriş örnekleri	
Örnek	Tanım
MVMP, YAYINLANIYOR,	Varsayılan bir ileti yayınla
MVMP,publish, A short message,	Verili metni yayınla
MVMP, DUR,	Zamanuyumsuz Tüketimi Durdur
MVMP,DEREGISTER,CSQ4CVEV,	Olay İşleyicisini kayıttan çıkar
MVMP,resume,csq4cvcn,	Basit İleti Tüketicisini Süzdür
MVMP,SUSPEND,CSQ4CVEV,	Olay İşleyicisini Askıya Al

Burada MVMP, Messaging Client programı CSQ4CVPTile ilişkili CICS işlemidir.

### Not:

- Tüm Callback işleyicilerin askıya alınması ya da kayıttan kaldırılması, Kayıt İstemcisi 'nin yayınladığı START\_WAWN işlemi, denetimi geri döndürerek ve görevi sonlandırarak sona erdirir.
- Control Callback Handler 'ın kasıtlı olarak askıya alınması ya da kayıttan kaldırılması, kasıtlı olarak önlenmedi; ancak, örneğe ilişkin davranışı daha fazla denetleyebilme yeteneğini ortadan kaldırır.

### z/OS z/OSüzerindeki Yayınlama/Abone Olma örneği

Yayınlama/Abone Olma örnek programları, IBM MQ' ta yayınlama ve abone olma özelliklerinin kullanımını gösterir.

IBM MQ Yayınlama/Abone Olma arabirimine nasıl programların göstermeyi gösteren dört adet C ve iki COBOL programlama dili örnek programı vardır. Programlar C ve COBOL dillerinde teslim edilir.

Uygulamalar toplu iş ortamında çalışır; toplu iş uygulamaları için [Örnekle/Abone Olma örnekleri](#) başlıklı konuya bakın.

Ayrıca, CICS ortamında çalışan COBOL örnekleri de vardır; bkz. [“z/OSüzerinde CICS Zamanuyumsuz Tüketimi ve Yayınlama/Abone Olma örneği”](#) sayfa 1244.

Bu konu ayrıca Publish/Subscribe örnek programlarının nasıl çalıştırılacağı hakkında bilgi de sağlar. Bu örnek programlar arasında şunlar yer alır:

- [“CSQ4BCP1 örneğinin çalıştırılması”](#) sayfa 1247
- [“CSQ4BCP2 örneğinin çalıştırılması”](#) sayfa 1247
- [“CSQ4BCP3 örneğinin çalıştırılması”](#) sayfa 1247
- [“CSQ4BCP4 örneğinin çalıştırılması”](#) sayfa 1248
- [“CSQ4BVP1 örneği çalıştırılıyor”](#) sayfa 1248
- [“CSQ4BVP2 örneğinin çalıştırılması”](#) sayfa 1248

### **CSQ4BCP1 örneğinin çalıştırılması**

Bu program C ' de yazılıdır; bir konuya ileti yayınlar. Bu programı çalıştırmadan önce abone örneklerinden birini başlatın.

Bu program en çok dört parametre içerir:

1. Hedef konu dizgisinin adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. Seçenekleri açın (isteğe bağlı).
4. Seçenekleri kapat (isteğe bağlı).

Kuyruk yöneticisi belirtilmediyse, CSQ4BCP1 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için bir JCL örneği vardır; CSQ4BCPP' de bulunur.

İleti içeriği standart giriş yoluyla sağlanır (**SYSD**).

### **CSQ4BCP2 örneğinin çalıştırılması**

Bu program C ' de yazılıdır; bir konuya abone olur ve alınan iletileri yazdırır.

Bu program üç parametrelere kadar sürer:

1. Hedef konu dizgisinin adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. MQSD abonelik seçenekleri (isteğe bağlı).

Kuyruk yöneticisi belirtilmediyse, CSQ4BCP2 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için bir JCL örneği vardır; CSQ4BCPS' de bulunur.

### **CSQ4BCP3 örneğinin çalıştırılması**

Bu program C ' de yazılıdır; kullanıcı tarafından belirlenen hedef kuyruğu kullanılarak bir konuya abone olur ve alınan iletileri yazdırır.

Bu program en çok dört parametre içerir:

1. Hedef konu dizgisinin adı (gerekli).
2. Hedefin adı (gerekli).
3. Kuyruk yöneticisinin adı (isteğe bağlı).
4. MQSD abonelik seçenekleri (isteğe bağlı).

Kuyruk yöneticisi belirtilmediyse, CSQ4BCP3 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için bir JCL örneği vardır; CSQ4BCPD' de bulunur.

## CSQ4BCP4 örneğinin çalıştırılması

Bu program C içinde yazılıdır; bu program, MQSUB çağrısında genişletilmiş seçeneklerin kullanılmasına izin veren bir konudan ileti alır ve daha basit MQSUB örneğinde kullanılabilir olanları uzatır: CSQ4BCP2. İleti bilgi yükünün yanı sıra, her ileti için ileti özellikleri alınır ve görüntülenir.

Bu program değişken bir değiştirge kümesi alır:

- **-t** *Topic string* (gerekli).
- **-o** *Topic object name* (gerekli).
- **-m** *Queue manager name* (isteğe bağlı).
- **-q** *Destination queue name* (isteğe bağlı).
- **-w** *Wait interval on MQGET in seconds* (isteğe bağlı); burada *seconds* , aşağıdaki değerlerden herhangi birine sahip olabilir:
  - unlimited: MQWI\_UNSNMA
  - none: Bekleme yok
  - *n*: Bekleme aralığı (saniye)
  - Değer belirtilmedi: Değer belirlenmezse, varsayılan değer 30 saniyedir.
- **-d** *Subscription name* (isteğe bağlı). Kalıcı abonelik adını yaratır ya da sürdürür.
- **-k** (isteğe bağlı). MQCLOSE ' ye sürekli abonelik alıyolar.

Kuyruk yöneticisi belirtilmediyse, CSQ4BCP4 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için bir JCL örneği vardır; CSQ4BCPE' de bulunur.

## CSQ4BVP1 örneği çalıştırılıyor

Bu program COBOL ' de yazılıdır, bir konuya ileti yayınlar. Bu programı çalıştırmadan önce abone örneklerinden birini başlatın.

Bu program parametre almaz. **SYSIN DD** , giriş konu adını, kuyruk yöneticisi adını ve ileti içeriğini sağlar.

Kuyruk yöneticisi belirtilmediyse, CSQ4BVP1 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için bir JCL örneği vardır, CSQ4BVPPiçinde bulunur.

## CSQ4BVP2 örneğinin çalıştırılması

Bu program COBOL ' de yazılıdır, bir konuya abone olur ve alınan iletileri yazdırır.

Bu program parametre almaz. **SYSIN DD** , konu adı ve kuyruk yöneticisi adı için giriş sağlar.

Kuyruk yöneticisi belirtilmediyse, CSQ4BVP1 varsayılan kuyruk yöneticisine bağlanır. Programı çalıştırmak için bir JCL örneği vardır, CSQ4BVPPiçinde bulunur.

## z/OSüzerinde Set ve Sorgula ileti özelliği örneği

İleti özelliği örnek programları, kullanıcı tanımlı özelliklerin bir ileti tanıtıcısı olarak eklenmesini ve bu iletiyle ilişkili özelliklerin sorgularını gösterir.

Uygulamalar bu 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, bir ileti kuyruğunun özelliklerini bir ileti kuyruğundan sorgulamak için kullanılır ve bu, MQINQMP API çağrısının kullanımına bir örnektir. Örnek bir kuyruktan bir ileti alır ve tüm ileti tanıtıcısı özelliklerini yazdırır.

CSQ4BCM2 programı, ileti kuyruğunda bir ileti tanıtıcısı özelliklerini belirlemek için kullanılır ve bu, MQSETMP API çağrısının kullanılmasına bir örnektir. Bu örnek, bir ileti tanıtıcısı yaratır ve bunu MQGMO yapısının MsgHandle alanına yerleştirir. Daha sonra, iletiyi bir kuyruğa yerleştirir.

Diğer araştırmacı ve yazdırma iletileri özellikleri örnekleri, CSQ4BCG1 ve CSQ4BCP4 örnek programlarında yer alır.

Bu konu ayrıca, aşağıdaki başlıklar altında Küme ve Sorgula ileti özelliği örneklerinin çalıştırılıp çalıştırılabildiğine ilişkin bilgiler de sağlar:

- [“CSQ4BCM1 örneğinin çalıştırılması” sayfa 1249](#)
- [“CSQ4BCM2 örneğinin çalıştırılması” sayfa 1249](#)

### **CSQ4BCM1 örneğinin çalıştırılması**

Bu program en çok dört parametre içerir:

1. Hedef kuyruğun adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. Seçenekleri açın (isteğe bağlı).
4. Seçenekleri kapat (isteğe bağlı).

### **CSQ4BCM2 örneğinin çalıştırılması**

Bu program altı parametrelere kadar sürer:

1. Hedef kuyruğun adı (gerekli).
2. Kuyruk yöneticisinin adı (isteğe bağlı).
3. Seçenekleri açın (isteğe bağlı).
4. Seçenekleri kapat (isteğe bağlı).
5. Hedef kuyruk yöneticisinin adı (isteğe bağlı).
6. Dinamik kuyruğun adı (isteğe bağlı).

Özellik adları, değerleri ve ileti içeriği, standart giriş ( **SYSD** ) aracılığıyla sağlanır. Programı çalıştırmak için bir JCL örneği vardır; CSQ4BCMPiçinde bulunur.

## **Managed File Transfer için uygulama geliştirilmesi**

Managed File Transfer ile çalıştırılacak programları belirtin, Apache Ant komutunu kullanarak Managed File Transfer komutunu kullanın, kullanıcı çıkışlarıyla Managed File Transfer uyarlayın ve aracı komut kuyruğuna ileti koyarak Managed File Transfer denetimini kullanın.

### **MFT ile çalışmak üzere programların belirtilmesi**

Programları Managed File Transfer Agent ' un çalıştığı bir sistemde çalıştırabilirsiniz. Bir dosya aktarma isteğinin bir parçası olarak, aktarma başlamadan önce ya da sona erdikten sonra çalıştırılacak bir program belirleyebilirsiniz. Ayrıca, yönetilen bir çağrı isteğini göndererek, dosya aktarma isteğinin bir parçası olmayan bir program başlatabilirsiniz.

Çalıştırılacak programı belirleyebileceğiniz beş senaryo vardır:

- Aktarma isteğinin bir parçası olarak, aktarma işlemi başlamadan önce kaynak aracıda
- Aktarma isteğinin bir parçası olarak, aktarma başlatılmadan önce hedef aracıda
- Aktarma isteğinin bir parçası olarak, aktarma tamamlandıktan sonra kaynak aracıda
- Aktarma isteğinin bir parçası olarak, aktarma tamamlanınca hedef aracıda
- Aktarma isteğinin bir parçası olarak değil. Bir programı çalıştırmak için bir aracıya istek gönderebilirsiniz. Bu senaryoya göre, bazen yönetilen çağrı olarak da adlandırılır.

Kullanıcı çıkışları ve program çağrıları aşağıdaki sırada 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** , aktarma tamamlanınca başarıyla ya da kısmen başarıyla çalıştırılır.
2. **postDestinationCall** , aktarma tamamlanınca başarıyla ya da kısmen başarıyla çalıştırılır.
3. **SourceTransferEndExits** , başarılı, kısmen başarılı ya da başarısız aktarımlar için çalıştırılır.
4. **postSourceCall** yalnızca aşağıdaki durumlarda çağrılır:
  - Aktarma iptal edilemedi.
  - Başarılı ya da kısmen başarılı bir sonuç var.
  - Hedef sonrası aktarma programları başarıyla çalıştırıldı.

Çalıştırmak istediğiniz bir programı belirtmenin birkaç yolu vardır. Bu seçenekler şunlardır:

#### Apache Ant görevi kullan

Bir program başlatmak için `fte:filecopy`, `fte:filemove` ve `fte:call` Ant görevlerinden birini kullanın. Using an Ant task, you can specify a program in any of the five scenarios, using the `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrc`, and `fte:command` nested elements. Ek bilgi için [Program çağırma içiçe yerleştirilmiş öğeler](#) başlıklı konuya bakın.

#### Dosya aktarma isteği iletisini düzenle

Bir aktarma isteği tarafından oluşturulan XML ' i düzenleyebilirsiniz. Using this method, you can run a program in any of the five scenarios, by adding **preSourceCall**, **postSourceCall**, **preDestinationCall**, **postDestinationCall**, and **managedCall** elements to the XML file. Daha sonra, bu değiştirilen XML dosyasını yeni bir dosya aktarma isteği (örneğin, **fteCreateTransfer -td** parametresiyle) için aktarma tanımlaması olarak kullanın. Daha fazla bilgi için bkz. [MFT aracı çağrı isteği iletisi örnekleri](#).

#### fteCreateTransfer komutunu kullanın.

Başlatılacak programları belirlemek için **fteCreateTransfer** komutunu kullanabilirsiniz. Bir aktarma isteğinin bir parçası olarak, ilk dört senaryoda çalıştırılacak programları belirlemek için bu komutu kullanabilirsiniz, ancak yönetilen bir aramayı başlatamazsınız. Kullanılacak parametrelerle ilgili bilgi için bkz. **fteCreateTransfer**: yeni bir dosya aktarımı başlatma. Bu komutu kullanmanın örnekleri için bkz. [fteCreateAkarımı programını başlatmak için kullanma örnekleri](#).

#### İlgili başvurular

[commandPath MFT özelliği](#)

## Yönetilen çağrılar

Managed File Transfer (MFT) araçları genellikle dosyaları ya da iletileri aktarmak için kullanılır. Bunlar *Yönetilen Aktarmalar* olarak bilinir. Araçlar, 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 görevi kullanılarak.
- 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.

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 çağrı 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 çağırarak 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 çağrılar diğer tüm tiplerinde, yönetilen çağrının 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 limit](#) başlıklı 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 aldığı anda 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 aktarımın tamamlanması uzun sürerse ya da kurtarma işlemine başlarsa, yönetilen aktarım tamamlanmaya, 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 çağrılarının sayısını sınırlayabilir.

Bu nedenle, kaynak aktarım yuvalarını uzun süre meşgul etmediğinden emin olmak için yönetilen bir çağrı tasarımını göz önünde bulundurmanız gerekir.

## Apache Ant ile MFTkomutunu kullanma

Managed File Transfer , dosya aktarma işlevini Apache Ant aracına bütünleştirmek için kullanabileceğiniz görevleri sağlar.

You can use the **fteAnt** command to run Ant tasks in a Managed File Transfer environment that you have already configured. Karmaşık dosya aktarma işlemlerini, yorumlanan bir komut dosyası oluşturma dilinden koordine etmek için Ant komut dosyalarınızdan dosya aktarma Ant görevlerini kullanabilirsiniz.

Apache Antile ilgili daha fazla bilgi için Apache Ant projesi web sayfasına bakın: <https://ant.apache.org/>

### İlgili kavramlar

“Getting started using Ant scripts with MFT” sayfa 1252

Using Ant scripts with Managed File Transfer allows you to coordinate complex file transfer operations from an interpreted scripting language.

### İlgili başvurular

**fteAnt**: run Ant tasks in MFT

“MFTiçin örnek Ant görevleri” sayfa 1253

Managed File Transferkurulumunuzla birlikte verilen bir dizi örnek Ant komut dosyası vardır. Bu örnekler *MQ\_INSTALLATION\_PATH/mqft/samples/fteant* dizininde bulunur. Her bir örnek komut dosyası bir `init` hedefi içerir, bu komut dosyalarını yapılandırmanızı kullanarak çalıştırmak için `init` hedefinde ayarlanan özellikleri düzenleyin.

## Getting started using Ant scripts with MFT

Using Ant scripts with Managed File Transfer allows you to coordinate complex file transfer operations from an interpreted scripting language.

### 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 Ant' e bütünleştirmek için kullanabileceğiniz görevler sağlar. Ant komut dosyaları hakkında bilgi edinmek için Apache Ant projesi web sayfasına bakın: <https://ant.apache.org/>

Examples of Ant scripts that use Managed File Transfer tasks are provided with your product installation in the directory *MQ\_INSTALLATION\_PATH/mqft/samples/fteant*

On protocol bridge agents, Ant scripts are run on the protocol bridge agent system. Bu Ant komut dosyalarında, FTP ya da SFTP sunucusundaki dosyalara doğrudan erişim yetkisi yoktur.

### Ad alanı

Bir ad alanı, dosya aktarma Ant görevlerini diğer Ant görevlerinden ayırt etmek için kullanılır. Bu görev, aynı adı paylaşabilecek olan diğer görevlerden de olabilir. You define the namespace in the project tag of your Ant script.

```
<?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>
```

The attribute `xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs"` tells Ant to look for the definitions of tasks prefixed by `fte` in the library `com.ibm.wmqfte.ant.taskdefs`.

Ad alanı örneğiniz olarak `fte` 'yi kullanmanıza gerek yoktur; herhangi bir değeri kullanabilirsiniz. The namespace prefix `fte` is used in all examples and sample Ant scripts.

## Ant komut dosyalarının çalıştırılması

Dosya aktarımı 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: run Ant tasks in MFT](#).

## Dönüş kodları

Dosya aktarımı Ant görevleri, Managed File Transfer komutlarıyla aynı dönüş kodlarını döndürür. Daha fazla bilgi için bakınız: [Return codes for MFT](#).

## İlgili başvurular

**fteAnt: run Ant tasks in MFT**

“MFT için örnek Ant görevleri” sayfa 1253

Managed File Transfer kurulumunuzla birlikte verilen bir dizi örnek Ant komut dosyası vardır. Bu örnekler `MQ_INSTALLATION_PATH/mqft/samples/fteantdizininde` bulunur. Her bir örnek komut dosyası bir `init` hedefi içerir, bu komut dosyalarını yapılandırmanızı kullanarak çalıştırmak için `init` hedefinde ayarlanan özellikleri düzenleyin.

## MFT için örnek Ant görevleri

Managed File Transfer kurulumunuzla birlikte verilen bir dizi örnek Ant komut dosyası vardır. Bu örnekler `MQ_INSTALLATION_PATH/mqft/samples/fteantdizininde` bulunur. Her bir örnek komut dosyası bir `init` hedefi içerir, bu komut dosyalarını yapılandırmanızı kullanarak çalıştırmak için `init` hedefinde ayarlanan özellikleri düzenleyin.

## e-posta

e-posta örneği, bir dosyayı aktarmak için Ant görevlerinin nasıl kullanılacağını gösterir ve aktarma başarısız olursa, belirli bir e-posta adresine e-posta gönderir. Komut dosyası, kaynak ve hedef araçların etkin olduğunu ve aktarımları Managed File Transfer ping görevini kullanarak gerçekleştirebildiğini denetler. Her iki görevli de etkinse, komut dosyası, özgün dosyayı silmeden kaynak ve hedef araçlar arasında bir dosya aktarmak için Managed File Transfer `fte: filecopy` görevini kullanır. Aktarım başarısız olursa komut dosyası, standart Ant e-posta görevini kullanarak hatayla ilgili bilgileri içeren bir e-posta gönderir.

## göbek

The göbek sample is made up of two scripts: `hubcopy.xml` and `hubprocess.xml`. The `hubcopy.xml` script shows how you can use Ant scripting to build 'hub and spoke' style topologies. Bu örnekte iki dosya, konuşma makinelerinde çalışan araçlardan, göbek makinesinde çalışan bir aracıya aktarılır. Both files are transferred at the same time, and when the transfers are complete the `hubprocess.xml` Ant script is run on the hub machine to process the files. Her iki dosya da doğru aktarılırsa, Ant komut dosyası, dosyaların içeriğini birleştirir. Dosyalar doğru şekilde aktarılamazsa, Ant komut dosyası aktarılan tüm dosya verilerini silerek temizler. Bu örneğin doğru çalışması için, Hub aracısının komut yoluna `hubprocess.xml` komut dosyasını yerleştirmeniz gerekir. Bir aracının komut yolunu ayarlama hakkında daha fazla bilgi için bkz. [commandPath MFT property](#).

## librarytransfer (yalnızca IBM i altyapısı)



**IBM i** Kitaplık aktarımı örneği, bir IBM i sisteminde bir IBM i kitaplığını ikinci bir IBM i sistemine aktarmak için Ant görevlerinin nasıl kullanılacağını gösterir.

**IBM i** IBM i üzerinde IBM WebSphere MQ File Transfer Edition 7.0.2 , yerel IBM i kitaplık nesnelere aktarımları için doğrudan destek içermez. Librarytransfer örneği, yerel kitaplık nesnelere iki IBM i sistemi arasında aktarmak için Managed File Transfer içinde kullanılabilir olan önceden tanımlanmış Ant görevleri olan IBM i üzerinde yerel saklama dosyası desteğini kullanır. Örnek, kaynak aracı sisteminde istenen kitaplığı geçici bir saklama dosyasına saklayan librarysave . sh yürütülebilir komut dosyasını çağırmak için bir Managed File Transfer filecopy görevinde bulunan < presrc > içiçe yerleşimli öğeyi kullanır. The save file is moved by the filecopy ant task to the destination agent system where a < postdst > nested element is used to invoke the executable script libraryrestore . sh to restore the library saved in the save file to the destination system.

**IBM i** Bu örneği çalıştırmadan önce, bazı yapılandırmayı librarytransfer . xml dosyasında açıklandığı gibi tamamlamaya gereksinim duyarsınız. Ayrıca, iki IBM i makinesinde çalışan bir Managed File Transfer ortamınız da olmalıdır. Kuruluş, ilk IBM i makinesinde çalışan bir kaynak araçından ve ikinci IBM i makinesinde çalışan bir hedef araçından oluşmalıdır. İki ajan birbiriyle iletişim kurabilmelidir.

**IBM i** Librarytransfer Sample, aşağıdaki üç dosyadan oluşur:

- librarytransfer . xml
- librarysave . sh (< önr > yürütülebilir komut dosyası)
- libraryrestore . sh (< postdst > yürütülebilir komut dosyası)

Örnek dosyalar şu dizinde yer alır: /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/librarytransfer

**IBM i** Bu örneği çalıştırmak için kullanıcının aşağıdaki adımları tamamlaması gerekir:

1. Qshell oturumu başlatın. Bir IBM i komut penceresi tipinde: STRQSH
2. Dizini bin dizinine aşağıdaki şekilde değiştirin:

```
cd /QIBM/ProdData/WMQFTE/V7/bin
```

3. Gereken yapılanışı tamamladıktan sonra, aşağıdaki komutu kullanarak örneği çalıştırın:

```
fteant -f /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/librarytransfer/librarytransfer.xml
```

## physicalfiletransfer (yalnızcaIBM i altyapısı)

**IBM i** The physicalfiletransfer sample demonstrates how to use Ant tasks to transfer a Source Physical or Database file from a library on one IBM i system to a library on a second IBM i system.

**IBM i** IBM WebSphere MQ File Transfer Edition 7.0.2 on IBM i does not include direct support for transfers of native Source Physical or Database files on IBM i . physicalfiletransfer örneği, complete'ta bulunan önceden tanımlanmış Ant Görevleri ile IBM i ' ta yerel saklama dosyası desteğini kullanır. Bu destek, tam Kaynak Fiziksel ve Veri Tabanı dosyalarını iki IBM i sistemi arasında aktarmak için Managed File Transfer ' ta kullanılabilir. Örnek, istenen Kaynak Fiziksel ya da Veritabanı dosyasını kaynak aracı sistemindeki bir kitaplıktan geçici bir saklama dosyasına kaydetmek üzere physicalfilesave . sh yürütülebilir bir komut dosyasını çağırmak için bir Managed File Transfer filecopy görevi içinde bir < presrc > içiçe yerleşimli öğeyi kullanır. Saklama dosyası, filecopy ant görevi tarafından hedef aracı sistemine taşınır. Burada, yürütülebilir komut dosyasını çağırmak için < postdst > içiçe yerleştirilmiş öğe kullanılır. physicalfilerestore . sh daha sonra, saklama dosyasının içindeki dosya nesnesini, hedef sistemdeki belirli bir kitaplığa geri yükler.

**IBM i** Bu örneği çalıştırmadan önce, bazı yapılandırmayı `physicalfiletransfer.xml` dosyasında açıklandığı gibi tamamlamanız gerekir. Ayrıca, iki IBM i sisteminde çalışan bir Managed File Transfer ortamınız da olmalıdır. Kurulum, ilk IBM i sisteminde çalışan bir kaynak aracından ve ikinci IBM i sisteminde çalışan bir hedef aracından oluşmalıdır. İki ajan birbiriyle iletişim kurabilmelidir.

**IBM i** `physicalfiletransfer Sample`, aşağıdaki üç dosyadan oluşur:

- `physicalfiletransfer.xml`
- `physicalfilesave.sh` (< önrnc> yürütülebilir komut dosyası)
- `physicalfilerestore.sh` (< postdst> yürütülebilir komut dosyası)

Örnek dosyalar şu dizinde yer alır: `/QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/physicalfiletransfer`

**IBM i** Bu örneği çalıştırmak için kullanıcının aşağıdaki adımları tamamlaması gerekir:

1. Qshell oturumu başlatın. Bir IBM i komut penceresi tipinde: STRQSH
2. Dizini bin dizinine aşağıdaki şekilde değiştirin:

```
cd /QIBM/ProdData/WMQFTE/V7/bin
```

3. Gereken yapılanışı tamamladıktan sonra, aşağıdaki komutu kullanarak örneği çalıştırın:

```
fteant -f /QIBM/ProdData/WMQFTE/V7/samples/fteant/ibmi/physicalfiletransfer/  
physicalfiletransfer.xml
```

## Zaman aşımı

Zamanaşımı örneği, dosya aktarımını denemek ve belirlenen bir zamanaşımı değerinden uzun sürerse, aktarma işlemi iptal etmek için Ant görevlerini nasıl kullanacağını gösterir. Komut dosyası, Managed File Transfer `fte: filecopy` görevini kullanarak bir dosya aktarımı başlatır. Bu aktarımın sonucu ertelendi. Komut dosyası, aktarımın tamamlanmak üzere belirli bir saniye sayısını beklemek için Managed File Transfer `fte: awaitrite Ant task` 'ı kullanır. Aktarma işlemi belirtilen zamanda tamamlanmazsa, dosya aktarımı iptal etmek için Managed File Transfer `fte: Ant görevini iptal et` kullanılır.

## vsamptansfer

**z/OS**

**z/OS** vsamptansfer örneği, bir VSAM veri kümesinden Managed File Transfer komutunu 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 desteklemiyor. The sample script unloads the VSAM data records to a sequential data set by using the `presrc Program` çağırma içiçe yerleştirilmiş öğeler to call the executable file `datasetcopy.sh`. Komut dosyası, kaynak aracından hedef aracıya sıralı veri kümesini aktarmak için Managed File Transfer `fte: filemove` görevini kullanır. Daha sonra komut dosyası, `loadvsam.jcl` komut dosyasını çağırma için `postdst Program` çağırma içiçe öğeleri kullanır. Bu JCL komut dosyası, aktarılan veri kümesi kayıtlarını bir hedef VSAM veri kümesine yükler. Bu örnek, bu dil seçeneğini göstermek için hedef çağrıya ilişkin JCL 'yi kullanır. Aynı sonuç, bunun yerine ikinci bir kabuk komut dosyası kullanılarak da elde edilebilir.

**z/OS** Bu örnek, kaynak ve hedef veri kümelerinin VSAM olarak kullanılmasını gerektirmez. Kaynak ve hedef veri kümeleri aynı türse, bu örnek her veri kümesi için çalışır.

**z/OS** Bu örneğin doğru çalışması için, kaynak aracının komut yoluna `datasetcopy.sh` komut dosyasını ve hedef aracının komut yoluna `loadvsam.jcl` komut dosyasını koymanız gerekir. Bir aracının komut yolunu ayarlama hakkında daha fazla bilgi için bkz. [commandPath MFT property](#).

## PostaKodu

zip örneği iki komut dosyasından oluşur: zip.xml ve zipfiles.xml. Örnek, bir dosya aktarma taşıma işlemini gerçekleştirmeden önce Ant komut dosyasını çalıştırmak için Managed File Transfer [fte: filemove](#) görevinin içinde presrc içiçe yerleştirilmiş öge ' in nasıl kullanılacağını gösterir. zip.xml komut dosyasında presrc adlı içiçe yerleştirilmiş öge 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 var olmasını gerektirir. Bunun nedeni, zipfiles.xml Ant komut dosyasının kaynak araçındaki dizinin içeriğini sıkıştırabilme için kullanılan hedefi içermesi olabilir. Bir aracının komut yolunu ayarlama hakkında daha fazla bilgi için bkz. [commandPath MFT property](#).

## 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 IBM tarafından 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 arabirimlerinin her birini özetler.

<i>Çizelge 194. Kaynak tarafı ve hedef tarafı çıkış noktalarının ve Java arabirimlerinin özeti</i>	
<b>Çıkış noktası</b>	<b>Gerçekleştirilecek Java arabirimi</b>
<b>Kaynak tarafı çıkış noktaları:</b>	
Tüm dosya aktarımı başlamadan önce	<a href="#">SourceTransferStartExit.java arabirimi</a>
Tüm dosya aktarımı tamamlandıktan sonra	<a href="#">SourceTransferEndExit.java arabirimi</a>
<b>Hedef tarafı çıkış noktaları:</b>	
Tüm dosya aktarımı başlamadan önce	<a href="#">DestinationTransferStartExit.java arabirimi</a>
Tüm dosya aktarımı tamamlandıktan sonra	<a href="#">DestinationTransferEndExit.java arabirimi</a>

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 çağruları aşağıdaki sırada 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** , aktarma tamamlanınca başarıyla ya da kısmen başarıyla çalıştırılır.
2. **postDestinationCall** , aktarma tamamlanınca başarıyla ya da kısmen başarıyla çalıştırılır.
3. **SourceTransferEndExits** , başarılı, kısmen başarılı ya da başarısız aktarımlar için çalıştırılır.
4. **postSourceCall** yalnızca aşağıdaki durumlarda çağrılır:
  - Aktarma iptal edilemedi.
  - Başarılı ya da kısmen başarılı bir sonuç var.
  - Hedef sonrası aktarma programları başarıyla çalıştırıldı.

## Kullanıcı çıkışınızı oluşturma

Kullanıcı çıkışı oluşturmak için arabirimler *MQ\_INSTALL\_DIRECTORY/mqft/lib/com.ibm.wmqfte.exitroutines.api.jar* içinde 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 `exitClassYolu`yla 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`, `destinationTransferStartExitClasses` ve `destinationTransferEndExitClasses`. Bu özelliklerin nasıl kullanılacağına ilişkin bilgi için bkz. [MFT Kullanıcı çıkışları için aracı özellikleri](#).

## Protokol köprüsü araçlarında kullanıcı çıkışlarının çalıştırılması

Kaynak aracı çıkışı çağırdığında, aktarım için kaynak öğelerin listesini iletir. Normal araçlar için bu, tam olarak nitelenmiş dosya adlarının bir listesidir. Dosyaların yerel olması (ya da bağlama yoluyla erişilebilir olması) gerektiğinden, çıkış dosyaya erişilebilir 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 araçlarında kullanıcı çıkışlarının çalıştırılması

Connect:Direct köprü araçlarında kullanıcı çıkışlarını çalıştıramazsınız.

## MFT kaynak ve hedef kullanıcı çıkışları

### Dizin ayırıcıları

Directory separators in source file specifications are always represented using forward slash (/) characters, regardless of how you have specified directory separators in the **fteCreateTransfer** command or in the IBM MQ Explorer. Bir çıkış yazdığınızda bunu dikkate almalısınız. Örneğin, şu kaynak dosyasının var olduğunu denetlemek istiyorsanız: c:\a\b.txt ve **fteCreateTransfer** komutunu ya da IBM MQ Explorer komutunu kullanarak bu kaynak dosyayı belirtmiş değilseniz, dosya adını şu şekilde depolanmış olarak not edin: c:/a/b.txt Bu nedenle, özgün c:\a\b.txt dizesini ararsanız bir eşleşme bulamayacaksınız.

### Kaynak tarafı çıkış noktaları

#### Tüm dosya aktarma işlemi başlamadan önce

Bu çıkış, aktarma isteği bekleyen aktarımlar listesinde bir sonraki aktarma isteği olduğunda ve aktarma başlamak üzere olduğunda kaynak aracı tarafından çağrılır.

Bu çıkış noktasının kullanımı, aşamalardaki dosyaları, aracının bir dış komut kullanarak okuma/yazma erişimine sahip olduğu bir dizine göndermek ya da hedef sistemdeki dosyaları yeniden adlandırmak için kullanılır.

Bu çıkışa aşağıdaki bağımsız değişkenleri iletin:

- Kaynak aracı adı
- Hedef aracı adı
- Ortam meta verileri
- Meta verileri aktar
- Dosya belirtileri (dosya meta verileri de içinde olmak üzere)

Bu çıkıştan döndürülen veriler aşağıdaki gibidir:

- Aktarım meta verileri güncellendi. Girdiler eklenebilir, değiştirilebilir ve silinebilir.
- Kaynak dosya adı ve hedef dosya adı çiftlerinden oluşan dosya belirtilerinin listesi güncellendi. Girdiler eklenebilir, değiştirilebilir ve silinebilir
- Aktarmaya devam edip etmeyeceğini belirten gösterge
- Aktarma Günlüğüne eklenecek dizgi.

Bu çıkış noktasındaki kullanıcı çıkış kodunu çağırarak için [SourceTransferStartExit.java](#) arabirimini uygulayın.

#### Tüm dosya aktarımı tamamlandıktan sonra

Bu çıkış, tüm dosya aktarımı tamamlandıktan sonra kaynak aracıya göre çağrılır.

Bu çıkış noktasının örnek kullanımı, aktarma işleminin tamamlandığını belirten bir e-posta ya da IBM MQ iletisi gönderme gibi bazı tamamlama görevlerini yerine getirmektedir.

Bu çıkışa aşağıdaki bağımsız değişkenleri iletin:

- Çıkış sonucunun aktarılması
- Kaynak aracı adı
- Hedef aracı adı
- Ortam meta verileri
- Meta verileri aktar
- Dosya sonuçları

Bu çıkıştan döndürülen veriler aşağıdaki gibidir:

- Aktarım Günlüğüne eklenecek güncellenmiş dizgi.

Bu çıkış noktasındaki kullanıcı çıkış kodunu çağırmak için [SourceTransferEndExit.java](#) arabirimini uygulayın.

## Hedef tarafı çıkış noktaları

### Tüm dosya aktarma işlemi başlamadan önce

Bu çıkış noktasının örnek kullanımı, hedefteki izinlerin geçerliliğini denetlemektedir.

Bu çıkışa aşağıdaki bağımsız değişkenleri iletin:

- Kaynak aracı adı
- Hedef aracı adı
- Ortam meta verileri
- Meta verileri aktar
- Dosya Belirtilimleri

Bu çıkıştan döndürülen veriler aşağıdaki gibidir:

- Hedef dosya adları kümesi güncellendi. Girişler değiştirilebilir, ancak eklenmez ya da silinmez.
- Aktarmaya devam edip etmeyeceğini belirten gösterge
- Aktarma Günlüğüne eklenecek dizgi.

Bu çıkış noktasındaki kullanıcı çıkış kodunu çağırmak için [DestinationTransferStartExit.java](#) arabirimini uygulayın.

### Tüm dosya aktarımı tamamlandıktan sonra

Bu kullanıcı çıkışından örnek olarak, aktarılan dosyaları kullanan bir toplu iş işlemi başlatmak ya da aktarma başarısız olursa bir e-posta göndermek olabilir.

Bu çıkışa aşağıdaki bağımsız değişkenleri iletin:

- Çıkış sonucunun aktarılması
- Kaynak aracı adı
- Hedef aracı adı
- Ortam meta verileri
- Meta verileri aktar
- Dosya sonuçları

Bu çıkıştan döndürülen veriler aşağıdaki gibidir:

- Aktarım Günlüğüne eklenecek güncellenmiş dizgi.

Bu çıkış noktasındaki kullanıcı çıkış kodunu çağırmak için [DestinationTransferEndExit.java](#) arabirimini uygulayın.

## İlgili kavramlar

[MFT kullanıcı çıkışları için Java arabirimleri](#)

### İlgili başvurular

“MFT kullanıcı çıkışları için uzaktan hata ayıklamayı etkinleştirme” sayfa 1262

Kullanıcı çıkışlarınızı geliştirirken, kodunuzdaki sorunları bulmanıza yardımcı olması için bir hata ayıklayıcı kullanmak isteyebilirsiniz.



“Örnek MFT kaynak aktarımı kullanıcı çıkışı” sayfa 1263

[MFT kaynak izleyicisi kullanıcı çıkışları](#)

## MFT aktarma G/Ç kullanıcı çıkışlarının kullanılması

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

Genellikle MFT aktarımları için, aracı, aktarıma ilişkin uygun dosya sistemleriyle etkileşim kurmak için yerleşik G/Ç sağlayıcılarından birinden seçer. Yerleşik G/Ç sağlayıcıları aşağıdaki dosya sistemi tiplerini destekler:

- Düzenli UNIX-type ve Windows-type dosya sistemleri
-  z/OS sıralı ve bölümlenmiş veri kümeleri (yalnızca z/OS üzerinde)
-  IBM i yerel saklama dosyaları (yalnızca IBM i üzerinde)
- IBM MQ Kuyruklar
- Uzak FTP ve SFTP iletişim kuralı sunucuları (yalnızca protokol köprüsü araçları için)
- Uzak Connect:Direct düğümleri (yalnızca Connect:Direct köprü araçları için)

Desteklenmeyen dosya sistemleri için ya da özel G/Ç davranışı gerektiren durumlarda, bir aktarma G/Ç kullanıcı çıkışı yazabilirsiniz.

Aktarma G/Ç kullanıcı çıkışları, kullanıcı çıkışları için var olan altyapıyı kullanır. Ancak, bu aktarma G/Ç kullanıcı çıkışları, her dosya için aktarma boyunca birden çok kez erişildiğinden, bu aktarma G/Ç kullanıcı çıkışları diğer kullanıcı çıkışlarından farklı olur.

Hangi G/Ç çıkış sınıflarının yükleneceğini belirtmek için IOExitClasses ( agent.properties dosyasında) adlı aracı özelliğini kullanın. Her çıkış sınıfını bir virgülle ayırın; örneğin:

```
IOExitClasses=testExits.TestExit1,testExits.testExit2
```

Aktarım G/Ç kullanıcı çıkışlarına ilişkin Java arabirimleri aşağıdaki gibidir:

### IOExit

G/Ç çıkışının kullanılıp kullanılmayacağı belirlendiğini belirlemek için kullanılan ana giriş noktası. Bu yönetim ortamı, IOExitPath yönetim ortamlarının yapılmasından sorumludur.

IOExitClasses aracı özelliği için yalnızca IOExit G/Ç çıkış arabirimini belirtmeniz gerekir.

### IOExitPath

Soyut bir arabirimi temsil eder; örneğin, bir veri kapsayıcıları kümesini temsil eden bir veri kapsayıcısı ya da joker karakter. Bu arabirimi gerçekleştiren bir sınıf eşgörünümü yaratamazsınız. Arabirim, yolun incelenmesini ve türetilen yolları listelemenizi sağlar. IOExitResourceYolu ve IOExitWildcardYol arabirimleri, IOExitPath' i uzatır.

### IOExitChannel

Enables data to be read from or written to an IOExitPath resource.

### IOExitRecordKanalı

Kayıt odaklı IOExitPath kaynakları için IOExitChannel arabirimini genişletir; bu kaynaklar, kayıtların çok sayıda kayıt içinde bir IOExitPath kaynağına okunmasını ya da bu kaynağa yazılmasını sağlar.

### IOExitLock

Paylaşılan ya da dışlayıcı erişim için bir IOExitPath kaynağındaki bir kilidi temsil eder.

### IOExitRecordResourcePath

Extends the IOExitResourcePath interface to represent a data container for a record-oriented file; for example, a z/OS data set. Verileri bulmak ve okuma ya da yazma işlemleri için IOExitRecordKanalı eşgörünümlerini yaratmak için arabirimi kullanabilirsiniz.

## IOExitResourceYolu

Bir veri kapsayıcısını temsil etmek için IOExitPath arabirimini genişletir; örneğin, bir dosya ya da dizin. Verileri bulmak için arabirimi kullanabilirsiniz. Arabirim bir dizini gösteriyorsa, yol listesini döndürmek için listPaths yöntemini kullanabilirsiniz.

## IOExitWildcardYolu

IOExitPath arabirimini, joker karakter anlamına gelen bir yolu temsil edecek şekilde genişletir. Bu arabirimi birden çok IOExitResourceYollarıyla eşleştirmek için kullanabilirsiniz.

## IOExitProperties

Specifies properties that determine how Managed File Transfer handles IOExitPath for certain aspects of I/O. Örneğin, ara kütüklerin kullanılması ya da aktarma yeniden başlatılıp başlatılmayacağı gibi, bir kaynağın yeniden başlanıp yeniden başlanmayacağı gibi.

## IBM i **IBM i üzerinde örnek MFT kullanıcı çıkışları**

Managed File Transfer , kuruluşunuzla birlikte IBM i ' e özgü örnek kullanıcı çıkışları sağlar. Örnekler, *MQMFT\_install\_dir/samples/ioexit-IBMi* ve *MQMFT\_install\_dir/samples/userexit-IBMidizinlerinde* yer alıyor.

### **com.ibm.wmqfte.exit.io.ibm.i.qdls.FTEQDLSExit**

com.ibm.wmqfte.exit.io.ibm.i.qdls.FTEQDLSExit örnek kullanıcı çıkışı, IBM i üzerindeki QDLS dosya sisteminde bulunan dosyaları aktarır. Çıkış kurulduktan sonra, /QDLS ile başlayan dosyalara otomatik olarak çıkışta yapılan aktarımlar otomatik olarak kullanılır.

Bu çıkışı kurmak için aşağıdaki adımları tamamlayın:

1. com.ibm.wmqfte.samples.ibm.i.ioexits.jar dosyasını *WMQFTE\_install\_dir/samples/ioexit-IBMi* dizininden temsilcinin exits dizinine kopyalayın.
2. Add com.ibm.wmqfte.exit.io.ibm.i.qdls.FTEQDLSExit to the IOExitClasses property.
3. Aracıyı yeniden başlatın.

### **com.ibm.wmqfte.exit.user.ibm.i.FileMemberMonitorExit**

com.ibm.wmqfte.exit.user.ibm.i.FileMemberMonitorExit örnek kullanıcı çıkışı, MFT dosya izleme programı 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 kullanın). Bu parametre, bir dosya üyeye IFS stili bir yol alır ve dosya ve üye genel arama karakterlerini 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" vardır. By default, this field is set to "unix," which causes the destination file to be called F00.MBR. Ayrıca, IBM i FTP FILE.MEMBER şemasını kullanmak için "ibmi" değerini de belirtebilirsiniz; örneğin, /QSYS.LIB/FOO.LIB/BAR.FILE/BAZ.MBR , BAR.BAZ olarak aktarılır.

Bu çıkışı kurmak için aşağıdaki adımları tamamlayın:

1. com.ibm.wmqfte.samples.ibm.i.userexits.jar dosyasını *WMQFTE\_install\_dir/samples/userexit-IBMi* dizininden temsilcinin exits dizinine kopyalayın.
2. agent.properties dosyasındaki sourceTransferStartExitSınıfları özelliğine com.ibm.wmqfte.exit.user.ibm.i.FileMemberMonitorExit ekleyin.
3. Aracıyı yeniden başlatın.

### **com.ibm.wmqfte.exit.user.ibm.i.EmptyFileDeleteExit**

Kaynak dosya üyesi, aktarımın bir parçası olarak silindiğinde, com.ibm.wmqfte.exit.user.ibm.i.EmptyFileDeleteExit örnek kullanıcı çıkışı boş bir dosya nesnesini siler. IBM i dosya nesnelere birçok üyeyi potansiyel olarak tutabileceği için, dosya nesnelere MFT tarafından

dizinler gibi işlem görür. Bu nedenle, MFT kullanarak bir dosya nesnesi üzerinde taşıma işlemi gerçekleştiremezsiniz; taşıma işlemleri yalnızca üye düzeyinde desteklenir. Sonuç olarak, bir üye üzerinde taşıma işlemi gerçekleştirdiğinizde, artık boş olan dosya geride bırakılır. Bu boş dosyaları, aktarma isteğinin bir parçası olarak silmek istiyorsanız bu örnek çıkışı kullanın.

"empty.file.delete" meta verileri için "true" değerini belirtiyorsanız ve bir FTEFileMemberaktarırsanız, dosya boşsa, örnek çıkışı üst dosyayı siler.

Bu çıkışı kurmak için aşağıdaki adımları tamamlayın:

1. Copy the com.ibm.wmqfte.samples.ibm.userexits.jar file from *WMQFTE\_install\_dir/samples/userexit-IBMi* to the agent's exits directory.
2. agent.properties dosyasındaki sourceTransferStartExitSınıfları özelliğine com.ibm.wmqfte.exit.user.ibm.EmptyFileDeleteExit ekleyin.
3. Aracıyı yeniden başlatın.

### İlgili başvurular

[“MFT aktarma G/Ç kullanıcı çıkışlarının kullanılması” sayfa 1260](#)

You can use Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for Managed File Transfer transfers.

[Kullanıcı çıkışlarına ilişkin MFT Agent özellikleri](#)

## MFT kullanıcı çıkışları için uzaktan hata ayıklamayı etkinleştirme

Kullanıcı çıkışlarınızı geliştirirken, kodunuzdaki sorunları bulmanıza yardımcı olması için bir hata ayıklayıcı kullanmak isteyebilirsiniz.

Aracıyı çalıştıran Java sanal makinesinde çalışan çıkışlar nedeniyle, genellikle tümleşik geliştirme ortamında bulunan doğrudan hata ayıklama desteğini kullanamazsınız. Ancak, JVM 'nin uzak hata ayıklamasını etkinleştirebilir ve uygun bir uzak hata ayıklayıcıyı bağlayabilirsiniz.

To enable remote debugging, use the standard JVM parameters **-Xdebug** and **-Xrunjdwp**. Bu özellikler, aracıyı **BFG\_JVM\_PROPERTIES** ortam değişkeniyle çalıştıran JVM 'ye geçirilir. Örneğin, UNIX 'da aşağıdaki komutlar aracıyı başlatır ve JVM 'nin TCP kapısı 8765 üzerindeki hata ayıklayıcı bağlantılarını dinlemesine neden olur.

```
export BFG_JVM_PROPERTIES="-Xdebug -Xrunjdwp:transport=dt_socket,server=y,address=8765"  
fteStartAgent -F TEST_AGENT
```

Aracı hata ayıklayıcı bağlanıncaya kadar başlamaz. **export** komutu yerine Windows üzerinde **set** komutunu kullanın.

Hata ayıklayıcı ile JVM arasında başka iletişim yöntemlerini de kullanabilirsiniz. Örneğin, JVM bu bağlantıyı tam tersi yerine hata ayıklayıcıya açabilir ya da TCP yerine paylaşılan bellek kullanabilirsiniz. Daha ayrıntılı bilgi için [Java Platform Debugger Architecture](#) belgelerine bakın.

Aracıyı uzaktan hata ayıklama kipinde başlatırken **-F** (ön plan) parametresini kullanmalısınız.

### Eclipse hata ayıklayıcısını kullanma

Aşağıdaki adımlar, Eclipse geliştirme ortamında uzaktan hata ayıklama yeteneği için geçerlidir. Ayrıca, JPDA-uyumlu diğer uzak hata ayıklayıcıları da kullanabilirsiniz.

1. Click **Çalıştır > Hata Ayıklama İletişim Kutu** (or **Çalıştır > Hata Ayıklama Yapılandırılmaları** or **Çalıştır > Hata Ayıklama** depending on your version of Eclipse).
2. Hata ayıklama yapılanışı yaratmak için yapılanış tipleri listesinde **Uzak Java Uygulaması** ögesini çift tıklatın.
3. Yapılandırma alanlarını doldurun ve hata ayıklama yapılandırmasını kaydedin. Aracı JVM 'yi hata ayıklama kipinde başlattığınız zaman, JVM 'ye şimdi bağlanabilirsiniz.

## Örnek MFT kaynak aktarımı 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 protokol köprüsü kimlik bilgileri kullanıcı çıkışı

Bu örnek kullanıcı çıkışının nasıl kullanılacağı hakkında bilgi için bakınız: [Mapping credential for a file server by using exit classicas.](#)

```
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 çıkışısının nasıl kullanılacağı hakkında bilgi için bkz. [ProtocolBridgePropertiesExit2: İletişim kuralı dosyası sunucusu özellikleri araniyor](#)

### 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 yerleştirerek MFT ' i denetleme

Aracı komut kuyruklarına ileti koyarak, Managed File Transfer ' i denetleyen bir uygulama yazabilirsiniz.

Aracının aşağıdaki eylemlerden birini gerçekleştirmesini istemek için bir aracının komut kuyruğuna bir ileti yerleştirebilirsiniz:

- Dosya Aktarma Yaratılması
- Zamanlanmış bir dosya aktarma işlemi yaratmanızı sağlar
- Bir dosya aktarımı iptal eder
- Zamanlanmış bir dosya aktarımı iptal eder
- Bir komut çağır
- İzleme Programı Yaratılması
- İzleme Programının Silinmesi
- Aracının etkin olduğunu belirtmek için ping işlemi gerçekleştirin

Aracının bu işlemlerden birini gerçekleştirmesini istemek için, iletinin aşağıdaki şemanın biriyle uyumlu bir XML biçiminde olması gerekir:

### **FileTransfer.xsd**

Bu biçimde iletiler, bir ktk aktarma ya da zamanlanan ktk aktarma, komut g " ndermek ya da bir ktk aktarma ya da zamanlanan ktk aktarmalarının iptal edilmesi için kullanılır. Ek bilgi için [Dosya aktarma isteği ileti biçimibaşlıklı konuya](#) bakın.

### **Monitor.xsd**

Bu biçimdeki iletiler bir kaynak izleme programı yaratmak ya da varolan bir kaynak izleme programını silmek için kullanılabilir. Ek bilgi için [MFT izleme isteği ileti biçimibaşlıklı konuya](#) bakın.

### **PingAgent.xsd**

Bu biçimdeki iletiler, etkin olduğunu denetlemek üzere bir aracıya ping komutu göndermek için kullanılabilir. Daha fazla bilgi için bkz. [Ping MFT aracı istek iletisi biçimi](#).

Aracı, istek iletilerine bir yanıt döndürür. Yanıt iletisi, istek iletisinde tanımlı bir yanıt kuyruğuna yerleştirilir. Yanıt iletisi, aşağıdaki şema tarafından tanımlanan bir XML biçiminde olur:

### **Reply.xsd**

Ek bilgi için [MFT aracı yanıt iletisi biçimibaşlıklı konuya](#) bakın.

## MQ Telemetry için uygulama geliştirilmesi

Telemetri uygulamaları, algılama ve kontrol cihazlarını internet üzerinde ve işletmelerde bulunan diğer bilgi kaynaklarıyla bütünleştirir.

Tasarım örüntülerini, çalışma örneklerini, 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 sorun giderme](#)

### **İlgili başvurular**

[MQ Telemetry Başvuru](#)

## IBM MQ Telemetry Transport Örnek programlar

Örnek komut dosyaları, örnek bir IBM MQ Telemetry Transport v3 istemci uygulaması (mqttv3app.jar) ile birlikte çalışır. IBM MQ 8.0.0 ve daha sonraki bir sürümü için, örnek istemci uygulaması artık MQ Telemetry içinde yer almıyor. Bu, (artık kullanılabilir değil) bir parçasıydı ( IBM Messaging Telemetry Clients SupportPac). Benzer örnek uygulamalar, Eclipse Paho ve MQTT.org' da serbestçe kullanılabilir olmaya 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 yayınlamak ve abone olmak ve güvenlik özellikleri eklemek üzere örnek programlar geliştirmenize yardımcı olması için bu siteleri kullanın.
- IBM Messaging Telemetry Clients SupportPac artık karşıdan yüklenebilir durumda değil. Daha önce 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 kitaplığını (mqttv3.jar) içeriyordu. Bunlar aşağıdaki dizinlerde sağlanmıştır:
    - ma9c/SDK/clients/java/org.eclipse.paho.sample.mqttv3app.jar
    - ma9c/SDK/clients/java/org.eclipse.paho.client.mqttv3.jar
  - Bu SupportPac' ın MA9C sürümünde /SDK/ dizini ve içeriği kaldırılmıştır:
    - Yalnızca örnek uygulamanın kaynağı (mqttv3app.jar) sağlandı. Bu dizindeydi:

```
ma9c/clients/java/samples/org/eclipse/paho/sample/mqttv3app/*.java
```

- Derlenmiş istemci kitaplığı sağlandı. Bu dizindeydi:

```
ma9c/clients/java/org.eclipse.paho.client.mqttv3-1.0.2.jar
```

If you still have a copy of the (no longer available) IBM Messaging Telemetry Clients SupportPac, information about installing and running the sample application is provided in [Komut satırını kullanarak MQ Telemetry kurulumu doğrulanıyor](#).

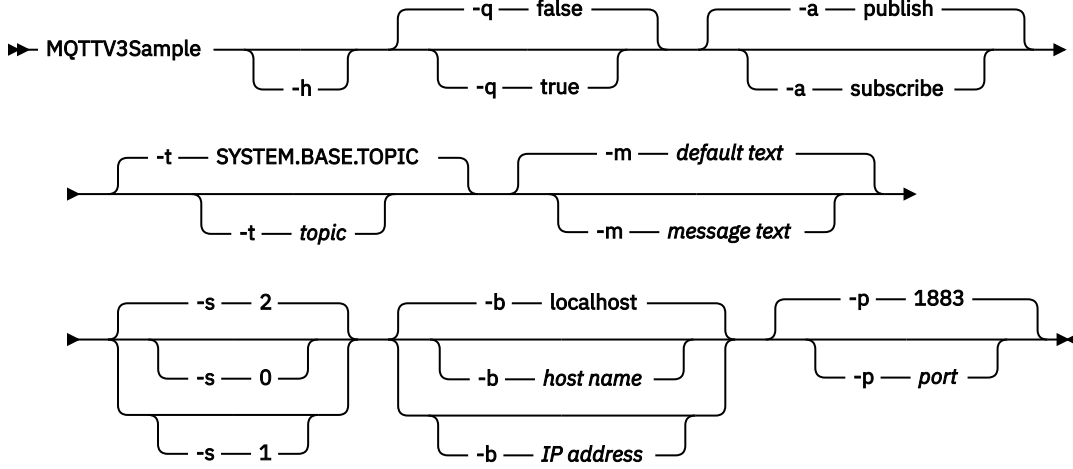
### MQTTV3Sample programı

MQTTV3Sample programına ilişkin örnek sözdizimi ve parametrelere ilişkin başvuru bilgileri.

#### Amaç

MQTTV3Sample programı, bir iletiyi yayınlamak ve bir konuya abone olmak için kullanılabilir. Bu örnek programın nasıl alayabileceği hakkında bilgi için bkz. ["IBM MQ Telemetry Transport Örnek programlar" sayfa 1270](#).

## MQTTV3Sample syntax



## Parametreler

### -h

Bu yardım metnini yazdır ve çık

### -q

False (yanlış) kipinin varsayılan kipini kullanmak yerine sessiz kipi ayarlayın.

### -a

Yayınlamanın varsayılan eylemini varsaymak yerine yayınlama ya da abone olma işlemini ayarlayın.

### -t

Varsayılan konuya abone olmak ya da varsayılan konuya abone olmak yerine, konuyu yayınlayın ya da konuya abone olun

### -m

İleti metnini, " MQTT v3 uygulamasından merhaba" varsayılan yayın metnini göndermek yerine 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

1883 varsayılan değerini kullanmak yerine bu kapıyı kullanın.

## MQTTV3Sample programını çalıştırın.

Windows üzerindeki bir konuya abone olmak için şu komutu kullanın:

```
run MQTTV3Sample -a subscribe
```

Windows' ta bir iletiyi yayınlamak için şu komutu kullanın:

```
run MQTTV3Sample
```

## MQTT istemci programlama kavramları

Bu bölümde anlatılan kavramlar, MQTT protokolünün 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 yükleyiyor. IBM MQ Telemetry Transport yayınlamak ve abone olmak ve güvenlik özellikleri eklemek üzere örnek programlar geliştirmenize yardımcı olması için bu siteleri kullanın.
- IBM Messaging Telemetry Clients SupportPac artık karşıdan yüklenebilir durumda değil. Daha önce karşıdan yüklenen bir kopyanızı varsa, bu kopyanın içeriği şöyledir:
  - IBM Messaging Telemetry Clients SupportPac MA9B sürümü, derlenmiş bir örnek uygulama (`mqt3app.jar`) ve ilişkili istemci kitaplığını (`mqt3.jar`) içeriyordu. Bunlar aşağıdaki dizinlerde sağlanmıştır:
    - `ma9c/SDK/clients/java/org.eclipse.paho.sample.mqt3app.jar`
    - `ma9c/SDK/clients/java/org.eclipse.paho.client.mqt3.jar`
  - Bu SupportPac' in MA9C sürümünde /SDK/ dizini ve içeriği kaldırılmıştır:
    - Yalnızca örnek uygulamanın kaynağı (`mqt3app.jar`) sağlandı. Bu dizindeydi:

```
ma9c/clients/java/samples/org/eclipse/paho/sample/mqt3app/*.java
```

- Derlenmiş istemci kitaplığı sağlandı. Bu dizindeydi:

```
ma9c/clients/java/org.eclipse.paho.client.mqt3-1.0.2.jar
```

Bir MQTT istemcisini geliştirmek ve çalıştırmak için bu kaynakları istemci aygıtına kopyanız ya da istemci aygıtında kurmanız gerekir. Ayrı bir istemci yürütme ortamı kurmanıza gerek yoktur.

İstemciler için lisans koşulları, istemcilere bağladığınız sunucuyla ilişkilendirilir.

MQTT istemci kitaplıkları, MQTT protocol' in başvuru somutlamalarıdır. Farklı aygıt platformlarına uygun farklı dillerde kendi müşterilerinizi uygulayabilirsiniz. Bkz. [IBM MQ Telemetry Transport biçimi ve iletişim kuralı](#).

API belgeleri, istemcinin bağlandığı MQTT sunucusu hakkında herhangi bir varsayımda yer vermez. İstemcinin davranışı, farklı sunuculara bağlanıldığında biraz farklı olabilir. The descriptions that follow describe the behavior of the client when connected to the IBM MQ telemetry service.

## Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback` u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

## Geri Çağırılmalar

**Not:** `MqttCallback` üzerindeki en son değişiklikler için [Eclipse Paho](#) web sitesine bakın. Örneğin, `MqttCallback` , istemcinin Paho sürümünde bir Arabirim olarak tanımlanır ve Asynchronous yöntemleri `PahoMqttAsyncClient` sınıfı tarafından sağlanır.

`MqttCallback` arabiriminde üç geri arama yöntemi vardır:

### **connectionLost(java.lang.Throwable cause)**

`connectionLost` , bir iletişim hatası bağlantının kesimine yol açınca çağrılır. Sunucu bağlantıyı, bağlantı kurulduktan sonra sunucudaki bir hatanın sonucu olarak düşerse de çağrılır. Sunucu hataları kuyruk yöneticisi hata günlüğüne kaydedilir. Sunucu, istemciye bağlantıyı atar ve istemci `MqttCallback.connectionLost` ' i çağırır.

İstemci uygulamasıyla aynı iş parçacığıdaki kural dışı durumlar olarak verilen uzak hatalar, `MqttClient.connect` ' ten kural dışı durumlardır. Errors detected by the server after the connection is established are reported back to the `MqttCallback.connectionLost` callback method as `throwables`.



connectionLost ile sonuçlanan tipik sunucu hataları, yetki hatalarıdır. Örneğin, telemetri sunucusu, konu üzerinde yayınlama yetkisi olmayan bir istemci adına bir konu üzerinde yayınlama girişiminde bulunuyor. Bir MQCC\_FAIL durum kodunun telemetri sunucusuna döndürülmesi sonucunda ortaya çıkan herhangi bir şey bağlantının atılma durumuyla sonuçlanabilir.

### **deliveryComplete(IMqttDeliveryToken token)**

deliveryComplete istemcisi, bir teslim simgesini istemci uygulamasına geri geçirmek için MQTT istemcisi tarafından çağrılır; bkz. [“Teslim simgeleri” sayfa 1278](#). Geri çağırma, teslim simgesini kullanarak yayınlanan iletiye token .getMessage yöntemiyle erişebilir.

Uygulama geri çağırısı, deliveryComplete yöntemi tarafından çağrıldıktan sonra MQTT istemcisine denetimi geri döndürdüğünde, teslim işlemi tamamlanır. Until delivery is completed, messages with QoS 1 or 2 are retained by the persistence class.

deliveryComplete çağırısı, uygulama ile kalıcılık sınıfı arasındaki eşitleme noktasıdır. deliveryComplete yöntemi, aynı ileti için hiçbir zaman iki kez çağrılmamaktadır.

Uygulama geri çağırısı deliveryComplete 'dan MQTT istemcisine geri döndüğünde, istemci QoS 1 ya da 2 olan iletiler için MqttClientPersistence .remove ' i çağırır. MqttClientPersistence .remove , yayınlanan iletinin yerel olarak saklanan kopyasını siler.

Bir hareket işleme perspektifinden deliveryComplete çağırısı, teslimi kesinleten tek aşamalı bir işlemdir. If processing fails during the callback, on restart of the client MqttClientPersistence .remove is called again to delete the local copy of the published message. Geri arama yeniden çağrılmaz. Geri bildirme olanağına, bir teslim edilen ileti günlüğünü saklamak için kullanıyorsanız, günlüğü MQTT istemcisiyle uyumlulaştıramazsınız. If you want to store a log reliably, then update the log in the MqttClientPersistence class.

Teslim simgesi ve iletilerine, ana uygulama iş parçacığı ve MQTT istemcisi tarafından başvuruluyor. MQTT istemcisi, teslim tamamlandığında MqttMessage nesnesini ve istemci bağlantısını kestiğinde teslim belirteci nesnesini kayıttan kaldırır. The MqttMessage object can be garbage collected after delivery is completed if the client application dereferences it. Teslim simgesi, oturumun bağlantısı kesildikten sonra çöp toplanabilir.

Bir ileti yayımlandıktan sonra IMqttDeliveryToken ve MqttMessage özniteliklerini elde edebilirsiniz. İleti yayımlandıktan sonra herhangi bir MqttMessage özneteliğini ayarlama girişiminde bulunursanız, sonuç tanımsız olur.

İstemci, aynı ClientIdentifier ile önceki oturuma yeniden bağlanıyorsa, MQTT istemcisi teslim onaylarını işlemeye devam eder. bkz. [“Temizleme oturumları” sayfa 1276](#). The MQTT client application must set MqttClient .CleanSession to false for the previous session, and set it to false in the new session. MQTT istemcisi, bekleyen teslimatlar için yeni oturum için yeni teslim simgeleri ve ileti nesneleri yaratır. MqttClientPersistence sınıfını kullanan nesneleri kurtarır. Uygulama istemcisinin eski teslim simgeleri ve iletilerine yönelik başvuruları varsa, bu istemcilerin başvuruları kaldırılır. Uygulama geri çağırısı, önceki oturumda başlatılan ve bu oturumda tamamlanan tüm teslimatlar için yeni oturumda çağrılır.

Uygulama geri çağırısı, uygulama istemcisi bağlandıktan sonra, bekleyen bir teslim tamamlandığında çağrılır. Before the application client connects, it can retrieve pending deliveries using the MqttClient .getPendingDeliveryTokens method.

İstemci uygulamasının özgün olarak yayınlanan ileti nesnesini ve bilgi yükü bayt dizisini oluşturduğunu fark edin. MQTT istemcisi bu nesnelere gönderme yapar. The message object returned by the delivery token in the method token .getMessage is not necessarily the same message object created by the client. Yeni bir MQTT istemcisi örneği teslim simgesini yeniden oluşturduysa, MqttClientPersistence sınıfı MqttMessage nesnesini yeniden oluşturur. For consistency token .getMessage returns null if token .isCompleted is true, regardless of whether the message object was created by the application client or the MqttClientPersistence class.

### **messageArrived(String topic, MqttMessage message)**

messageArrived , bir abonelik konularıyla eşleşen istemci için bir yayın geldiğinde çağrılır. konu , abonelik süzgecinin değil, yayın konudur. Süzgeç joker karakterler içeriyorsa, bu ikisi farklı olabilir.

Konu, istemci tarafından yaratılan birden çok abonelikte eşleşiyorsa, istemci yayının birden çok kopyasını alır. Bir istemci, aynı zamanda abone olduğu bir konuya yayınlarsa, kendi yayınının bir kopyasını alır.

If a message is sent with a QoS of 1 or 2, the message is stored by the `MqttClientPersistence` class before the MQTT client calls `messageArrived`. `messageArrived` behaves like `deliveryComplete`: it is only called once for a publication, and the local copy of the publication is removed by `MqttClientPersistence.remove` when `messageArrived` returns to the MQTT client. The MQTT client drops its references to the topic and message when `messageArrived` returns to the MQTT client. Uygulama istemcisi nesnelere ilişkin bir başvuruya tutmadıysa, konu ve ileti nesnelere çöp toplanır.

## Geri çağrılar, threading ve istemci uygulaması eşitlemesi

MQTT istemcisi, ana uygulama iş parçacığındaki ayrı bir iş parçacığında bir geri çağırma yöntemi çağırır. The client application does not create a thread for the callback, it is created by the MQTT client.

MQTT istemcisi geri çağırma yöntemlerini uyumlulaştırır. Geri bildirme yönteminin yalnızca bir kerede tek bir eşgörünümü çalışır. Eşitleme, yayınların teslim edildiği bir nesnenin güncellenmesini kolaylaştırır. `MqttCallback.deliveryComplete` 'un bir eşgörünümü bir kerede çalışır ve bu nedenle, daha fazla eşitleme olmadan tally' yi güncelleştirmek güvenlidir. Aynı zamanda bir kerede tek bir yayının gelmesi de geçerli olur. `messageArrived` yöntemindeki kodunuz, bir nesneyi uyumlulaştırmadan güncelleyebilir. Tally ya da güncellenmekte olan nesnede başka bir iş parçacığıysa, tally ya da object nesnesini eşitleyin.

Teslim belirteci, ana uygulama iş parçacığı ile bir yayının teslimi arasında bir eşitleme mekanizması sağlar. `token.waitForCompletion` yöntemi, belirli bir yayının teslimi tamamlanıncaya kadar bekler ya da isteğe bağlı bir zamanaşımı süresi doluncaya kadar bekler. Bir yayını aynı anda işlemek için `token.waitForCompletion` ' i aşağıdaki şekilde kullanabilirsiniz.

`MqttCallback.deliveryComplete` yöntemiyle eşitlemek için. Yalnızca `MqttCallback.deliveryComplete` , MQTT Client 'a geri dönünce `token.waitForCompletion` devam eder. Using this mechanism you can synchronize running code in `MqttCallback.deliveryComplete` before code runs in the main application thread.

Ya her yayının teslim edilmesini beklemeden yayınlamak istesiniz, ancak tüm yayınlar teslim edildiğinde onay almak istesiniz? Tek bir iş parçacığında yayınlarsanız, gönderilecek son yayın da son teslim edilecek son yayın olur.

## Sunucuya gönderilen isteklerin eşitlenmesi

Çizelge 195 sayfa 1274 , sunucuya bir istek gönderen MQTT Java istemcideki yöntemleri açıklar. Uygulama istemcisi belirsiz bir zamanaşımı ayarlamazsa, istemci sunucu için süresiz olarak beklemez. İstemci askıda kalırsa, bu bir uygulama programlama sorunu ya da MQTT istemcisinden bir hata.

Çizelge 195. Sunucuya gelen isteklerle sonuçlanan yöntemlerin eşzamanlama davranışı		
Yöntem	Eşitleme	Zamanaşımı aralığı
<code>MqttClient.Connect</code>	Sunucuya bağlantı kurulması için bekler.	Varsayılan olarak 30 saniye ya da bir parametre tarafından ayarlandığı gibi bir kural dışı durum yayınlıyor.
<code>MqttClient.Disconnect</code>	MQTT istemcisinin yapması gereken işi bitirmesini ve TCP/IP oturumunun bağlantısını kesmesini bekler.	
<code>MqttClient.Subscribe</code>	Subscribe ya da UnSubscribe yönteminin tamamlanmasını bekler.	
<code>MqttClient.UnSubscribe</code>		

Çizelge 195. Sunucuya gelen isteklerle sonuçlanan yöntemlerin eşzamanlama davranışı (devamı var)		
Yöntem	Eşitleme	Zamanaşımı aralığı
MqttClient.Publish	İsteği MQTT istemcisine ilettikten sonra uygulama iş parçacığına hemen döner.	Yok.
IMqttDeliveryToken.waitForCompletion	Teslim simgesinin geri döndürülmesini bekler.	Belirsiz ya da parametre olarak ayarlandığı gibi.

## İlgili kavramlar

### Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce MqttConnectOptions.cleanSession ayarını yaparak temizleme oturumu kipini değiştirin.

### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

### Teslim simgeleri

#### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

#### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez" ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

### Yayımlar

Yayımlar, bir konu dizgisiyle ilişkili MqttMessage örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

#### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayımları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

#### Alıkonan yayımlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayını hemen size iletilir.

### Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayımlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

#### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayımlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ içindeki konu dizgileriyle aynıdır.

## Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirin.

`MqttClient.connect` yöntemini kullanarak bir MQTT istemcisi uygulaması bağladığınızda, istemci bağlantıyı istemci tanıtıcısını ve sunucunun adresini kullanarak tanımlar. Sunucu, oturum bilgilerinin sunucuya önceki bir bağlantıdan saklanıp saklanmayacağını denetler. Önceki bir oturum hala varsa ve `cleanSession=true`, istemcideki ve sunucudaki önceki oturum bilgileri temizlenir. `cleanSession=false` önceki oturuma devam ederse. Önceki oturum yoksa, yeni bir oturum başlatılır.

**Not:** IBM MQ Administrator, açık bir oturumu zorla kapatabilir ve tüm oturum bilgilerini silebilirler. İstemci oturumu `cleanSession=false` ile yeniden açarsa, yeni bir oturum başlatılır.

## Yayınlar

Varsayılan `MqttConnectOptions` değerini kullanırsanız ya da istemciyi bağlamadan önce `MqttConnectOptions.cleanSession` değerini `true` olarak ayarlıyorsanız, istemci bağlandığında istemciye ilişkin bekleyen tüm yayın teslimleri kaldırılır.

Temizleme oturumu ayarının `QoS=0` ile gönderilen yayınlarda etkisi yoktur. `QoS=1` ve `QoS=2` için, `cleanSession=true` kullanılması bir yayını kaybetmeye neden olabilir.

## 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` özneliğinin abonelikleri etkileme şeklini anlamamanın başka bir yolu da bunu kalıcı bir öznelik 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 yayınları 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

### Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback`'u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

## Teslim simgeleri

### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez"ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

### Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili MqttMessage örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

### Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayın hemen size iletilir.

### Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ içindeki konu dizgileriyle aynıdır.

## **İstemci tanıtıcısı**

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

İstemci tanıtıcısı, bir MQTT sisteminin yönetiminde kullanılır. Potansiyel olarak yüz binlerce müşteriyle birlikte, belirli bir müşteriye hızla tanımlamayı başarmanız gerekir. Örneğin, bir aygıt arızalansa ve size bir yardım masası çalan bir müşteri tarafından bildirildiğini varsayalım. Müşterinin aygıtı tanınması gerekir ve bu tanıma, genellikle istemciye bağlı olan sunucu ile ilintilendirmeniz gerekir.

MQTT istemci bağlantılarına göz attığınızda, her bağlantı istemci tanıtıcısı ile etiketlenir. Bu tanıtıcıyı aygıt ve sunucuya eşlemek için en iyi şekilde karar vermeye yardımcı olmak için kendinize aşağıdaki soruları sorun:

- Her bir aygıtı bir istemci tanıtıcıyla ve bir sunucuya eşleyen bir veritabanını korumak ve kullanmak uygun olur mu?
- Aygıtın adı, bağlı olduğu sunucuyu tanımlıyor olabilir mi?
- Bir istemci tanıtıcısını fiziksel bir aygıtle eşleyen bir arama çizelgesine mi gereksiniminiz var?
- İstemci tanıtıcısı, belirli bir aygıtı, kullanıcıyı mı, yoksa istemcide çalışan bir uygulamayı mı tanımlıyor?
- Bir müşteri arızalı bir aygıtı yeni bir aygıtle değiştirirse, yeni aygıt eski aygıtle aynı tanıtıcıya sahip olur mu, yoksa yeni bir tanıtıcı mı ayırıyorsunuz? (Bir fiziksel aygıtı değiştirdiğinizde ve aynı tanıtıcıyı alıyorsanız, olağanüstü yayınlar ve etkin abonelikler otomatik olarak yeni aygıta aktarılır.)

İstemci tanıtıcılarının benzersiz olduğundan emin olmak için bir sisteme de gereksinim duyarsınız ve istemcide tanımlayıcıyı ayarlamak için güvenilir bir süreçiniz olmalıdır. İstemci aygıtı "kara kutu" ise, kullanıcı arabirimi yoksa, aygıtı bir istemci tanıtıcısına sahip olarak üretebilir ya da aygıtı etkinleştirmeden önce yapılandırabilecek bir yazılım kuruluşu ve yapılandırma işlemi olabilir.

Tanıttıcıyı kısa ve benzersiz tutmak için 48 bit aygıt MAC adresinden bir istemci tanıtıcısı yaratabilirsiniz. İletim büyüklüğü kritik bir sorun değilse, adresi denetlemek için kalan 17 baytı daha kolay kullanabilirsiniz.

## İlgili kavramlar

### Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback` 'u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

### Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirin.

### Teslim simgeleri

#### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez" ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

### Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili `MqttMessage` örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

### Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayını hemen size iletilir.

### Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ içindeki konu dizgileriyle aynıdır.

## Teslim simgeleri

Bir istemci bir konuda yayınlandığında yeni bir teslim belirteci oluşturulur. Bir yayının teslimini izlemek için teslim simgesini ya da teslim edilinceye kadar istemci uygulamasını engellemek için kullanın.

Simge, bir `MqttDeliveryToken` nesnesidir. It is created by calling the `MqttTopic.publish()` method and is retained by the MQTT client until the client session is disconnected and the delivery is completed.

Simgenin normal kullanımı, teslimin tamamlanıp tamamlanmadığını kontrol etmek için kullanılır. Teslim edilinceye kadar istemci uygulamasını engelle, döndürülen simgenin kullanılması için `token.waitForCompletion` komutunu çağırın. Diğer bir seçenek olarak, bir `MqttCallback` işleyicisi de sağlayın. When the MQTT client has received all the acknowledgments it expects as part of delivering the publication, it calls `MqttCallback.deliveryComplete` passing the delivery token as a parameter.

Until delivery is complete, you can inspect the publication using the returned delivery token by calling `token.getMessage`.

## Tamamlanan teslimatlar

Tesllerin tamamlanması zamanuyumsuz olur ve yayınlı ilişkili hizmet kalitesine bağlıdır.

### En çok bir kez

`QoS=0`

Delivery is complete immediately on return from `MqttTopic.publish`.  
`MqttCallback.deliveryComplete` hemen çağrılır.

### En az bir kez

`QoS=1`

Yayın, kuyruk yöneticisinden yayınlı ilgili bir alındı bildirimini alındığında tamamlanır. `MqttCallback.deliveryComplete` is called when the acknowledgment is received. İletişim yavaşsa ya da güvenilmezse, ileti `MqttCallback.deliveryComplete` çağrılmadan önce bir kereden fazla sağlanabilir.

### Tam bir kez

`QoS=2`

Müşteri, yayının abonelere yayımlandığını belirten bir tamamlanma iletisi aldığı anda teslim edilir. `MqttCallback.deliveryComplete` is called as soon as the publication message is received. Bu, tamamlanma iletisinin beklemesini beklemez.

In rare circumstances, your client application might not return to the MQTT client from `MqttCallback.deliveryComplete` normally. You know that delivery has completed, because the `MqttCallback.deliveryComplete` was called. İstemci aynı oturumu yeniden başlattıysa, `MqttCallback.deliveryComplete` yeniden çağrılmaz.

## Eksik teslim sayısı

İstemci oturumunun bağlantısı kesildikten sonra teslim tamamlanmazsa, istemciyi yeniden bağlayabilir ve teslimatı tamamlayabilirsiniz. İleti, `MqttConnectionOptions` özneliği `false` değerine ayarlanmış bir oturumda yayımlandıysa, iletinin teslim edilmesini yalnızca tamamlayabilirsiniz.

Aynı istemci tanıtıcısını ve sunucu adresini kullanarak istemciyi yaratın ve daha sonra, `cleanSession` `MqttConnectionOptions` özneliğini `false` ' e yeniden ayarlayarak bağlanın. `cleanSession` seçeneğini `true` olarak ayarlıyorsanız, bekleyen teslim simgeleri atılır.

Bekleyen teslimatın olup olmadığını denetlemek için `MqttClient.getPendingDeliveryTokens` ' i arayarak denetleyebilirsiniz. İstemciyi bağlamadan önce `MqttClient.getPendingDeliveryTokens` ' u arayabilirsiniz.

## İlgili kavramlar

[Callbacks and synchronization in MQTT client applications](#)

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback` ' u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

[Temizleme oturumları](#)

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirin.

#### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

#### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

#### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez" ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

#### Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili `MqttMessage` örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

#### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

#### Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayını hemen size iletilir.

#### Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

#### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ içindeki konu dizgileriyle aynıdır.

## **Son irade ve vasiyet yayını**

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

Son irade ve vasiyet için bir konu oluşturun. `MQTTManagement/Connections/server URI/client identifier/Last` gibi bir konu oluşturabilirsiniz.

`MqttConnectionOptions.setWill(MqttTopic lastWillTopic, byte [] lastWillPayload, int lastWillQos, boolean lastWillRetained)` yöntemini kullanarak bir "son irade ve ahit" olarak ayarlayın.

`lastWillPayload` iletisinde bir zaman damgası yaratmayı düşünün. İstemcinin belirlenmesine ve bağlantının koşullarına yardımcı olacak diğer istemci bilgilerini de ekleyin. `MqttConnectionOptions` nesnesini `MqttClient` oluşturucusuna geçirin.



Set `lastWillQos` to 1 or 2, to make the message persistent in IBM MQ, and to guarantee delivery. Son kayıp bağlantı bilgilerini korumak için `lastWillRetained`, `true` olarak ayarlayın.

Bağlantının beklenmedik bir şekilde sona ermesi durumunda abonelere "son irade ve ahit" yayını gönderilir. It is sent if the connection ends without the client calling the `MqttClient.disconnect` method.

Bağlantıları izlemek için, bağlantıları kaydetmek ve programlanmış bağlantıları kaydetmek için diğer yayınlarla "last will and ahit" yayını tamamlar.

## **İlgili kavramlar**

### Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback`' u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

### Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirir.

### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

### Teslim simgeleri

#### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez" ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

### Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili `MqttMessage` örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

### Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayın hemen size iletilir.

### Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ içindeki konu dizgileriyle aynıdır.

## Message persistence in MQTT clients

Yayın iletileri, "en az bir kez"ya da "aynen bir kez"hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

In MQTT, message persistence has two aspects; how the message is transferred, and whether it is queued in IBM MessageSight and IBM MQ as a persistent message.

1. MQTT istemcisi çiftleri, hizmet kalitesiyle ileti kalıcılığı sağlar. İleti için seçtiğiniz hizmet kalitesine bağlı olarak, ileti kalıcı olarak yapılır. İleti sürekliliği, gereken hizmet kalitesini uygulamak için gereklidir.

"En çok bir kez"seçeneğini belirlerseniz, QoS=0, iletiyi yayınlanır yayınlanmaz ileti atar. İletinin yukarı işlenmesinde herhangi bir hata varsa, ileti yeniden gönderilmez. İstemci etkin olmaya devam ederse bile, ileti yeniden gönderilmez. QoS=0 iletilerinin davranışı, IBM MQ hızlı olmayan kalıcı olmayan iletiler ile aynıdır.

Bir ileti, 1 ya da 2 değeri QoS olan bir istemci tarafından yayınlanırsa, kalıcı olarak yapılır. İleti yerel olarak saklanır ve istemciden "en az bir kez", QoS=1ya da "tam olarak bir kez", QoS=2, teslim etme konusunda garanti vermek zorunda kalmadığında çıkarılır.

2. Bir ileti QoS 1 ya da 2 olarak işaretlenirse, bu ileti IBM MessageSight ve IBM MQ ' ta kalıcı bir ileti olarak kuyruğa alınır. QoS=0olarak işaretlenmişse, IBM MessageSight ve IBM MQ içinde kalıcı olmayan bir ileti olarak kuyruğa alınır. In IBM MQ nonpersistent messages are transferred between queue managers "aynen bir kez", unless the message channel has the NPMSPEED attribute set to FAST.

Kalıcı bir yayın, istemci uygulaması tarafından alınıncaya kadar istemcide depolanır. QoS=2için, uygulama geri çağırısı denetimi geri döndürdüğünde, yayın istemciden atılır. QoS=1 için, bir hata oluşursa, uygulama yayını yeniden alabilir. QoS=0için geri bildirme, yayını bir kereden fazla alır. Bir hata varsa ya da yayınlama sırasında istemcinin bağlantısı kesildiyse, bu yayın olmayabilir.

Bir konuya abone olduğunuzda, abonenin kalıcılık yetenekleriyle eşleşmesi için iletilerin aldığı QoS değerini azaltabilirsiniz. Daha yüksek bir QoS düzeyinde oluşturulan yayınlar, abonenin talep ettiği en yüksek QoS ile gönderilir.

## İletilerin saklanması

küçük cihazlarda veri depolamanın uygulanması büyük bir anlaşmaya göre değişiklik gösteriyor. MQTT istemcisi tarafından yönetilen depolama alanındaki kalıcı iletilerin geçici olarak kaydedilmesi modeli çok yavaş olabilir ya da çok fazla depolama talep edebilir. Mobil aygıtlarda, mobil işletim sistemi, MQTT iletileri için ideal olan bir depolama hizmeti sağlayabilir.

Küçük aygıtların kısıtlamalarını yerine getirmede esneklik sağlamak için, MQTT istemcisinin iki kalıcılık arabirimi vardır. Arabirimler, kalıcı iletilerin depolanması içinde yer alan işlemleri tanımlar. Arabirimler, MQTT client for Javaiçin API belgelerinde açıklanmıştır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. [MQTT istemci programlama başvurusu](#). Bir aygıtta uyacak şekilde arabirimleri uygulayabilirsiniz. Java SE ' de çalışan MQTT istemcisinde, dosya sisteminde kalıcı iletileri saklayan arabirimlerin varsayılan bir somutlaması vardır. `java.io` paketini kullanır.

## Kalıcılık sınıfları

### **MqttClientPersistence**

MqttClientPersistence uygulamanızın bir eşgörünümünü, MqttClient oluşturucusunun bir parametresi olarak MQTT istemcisine geçirin. MqttClientPersistence parametresini MqttClient oluşturucudan çıkarırsanız, MQTT istemcisi kalıcı iletileri MqttDefaultFilePersistencesınıfını kullanarak saklar.

### **MqttPersistable**

MqttClientPersistence , MqttPersistable nesnelere bir depolama anahtarı kullanarak alır ve yerleştirir. You must provide an implementation of MqttPersistable as well as the implementation of MqttClientPersistence if you are not using the MqttDefaultFilePersistence.

## MqttDefaultFilePersistence

MQTT istemcisi MqttDefaultFilePersistence sınıfını sağlar. If you instantiate MqttDefaultFilePersistence in your client application, you can provide the directory to store persistent messages as a parameter of the MqttDefaultFilePersistence constructor.

Alternatively, the MQTT client can instantiate MqttDefaultFilePersistence and place files in the following default directory:

```
client identifier -tcp hostname portnumber
```

Dizin adı dizgisinden aşağıdaki karakterler kaldırılır:

```
"\", "\\\", \"/\", \":\" ve " "
```

The path to the directory is the value of the system property `tcp.data`; If `tcp.data` is not set, the path is the value of the system property `usr.data`, where

- `tcp.data` , bir OSGi ya da Eclipse Rich Client Platform (RCP) kuruluşuyla ilişkilendirilmiş bir özeldir.
- `usr.data` , uygulamanın başlatıldığı Java komutunun başlatıldığı dizindir.

## İlgili kavramlar

### Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, MqttCallback' u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

### Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce MqttConnectOptions.cleanSession ayarını yaparak temizleme oturumu kipini değiştirir.

### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

### Teslim simgeleri

#### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

### Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili MqttMessage örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

### Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayın hemen size iletilir.

### Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir.

Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

#### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ'deki konu dizgileriyle aynıdır.

## Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili `MqttMessage` örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

`MqttMessage`, bilgi yükü olarak bayt dizisine sahiptir. İletilerin mümkün olduğunca küçük olmasını hedefle. MQTT protocol tarafından izin verilen ileti uzunluğu üst sınırı 250 MB'dir.

Tipik olarak, bir MQTT istemci programı ileti içeriğini işlemek için `java.lang.String` ya da `java.lang.StringBuffer` kullanır. Kolaylık sağlamak amacıyla, `MqttMessage` sınıfında bilgi yükünü dizgiye dönüştürecek bir `toString` yöntemi vardır. To create the byte array payload from a `java.lang.String` or `java.lang.StringBuffer`, use the `getBytes` method.

`getBytes` yöntemi, bir dizeyi, platform için varsayılan karakter kümesine dönüştürür. Varsayılan karakter kümesi genellikle UTF-8 karakteridir. Yalnızca metin içeren MQTT yayınları genellikle UTF-8 ile kodlanır. Varsayılan karakter kümesini geçersiz kılmak için `getBytes("UTF8")` yöntemini kullanın.

IBM MQ' ta, bir MQTT yayını `jms-bytes` iletisi olarak alınır. İleti, `<mqtt>` ve bir `<mqps>` klasörünü içeren bir `MQRFH2` klasörü içerir. `<mqtt>` klasörü, `clientId`, `msgId` ve `qos` sözcüklerini içerir, ancak bu içerik ileride değişebilir.

Bir `MqttMessage` ' de üç ek öznitelik vardır: hizmet kalitesi, alıkonulup tutulmadığına ve yinelenip yinelenmeyeceği. Yinelenen işaret, hizmet kalitesi "en az bir kez" ya da "tam olarak bir kez" olduğunda ayarlanır. İleti daha önce gönderildiyse ve MQTT istemcisi tarafından yeterince hızlı bir şekilde onaylanmadıysa, yinelenen öznitelik `true` olarak ayarlanarak ileti yeniden gönderilir.

## Yayınlama

Bir MQTT istemcisi uygulamasında bir yayın oluşturmak için bir `MqttMessage` oluşturun. Bilgi yükünü, hizmet kalitesini ve alıkonulup tutulmadığını belirleyin ve `MqttTopic.publish(MqttMessage message)` yöntemini çağırın; `MqttDeliveryToken` iade edilir ve yayının tamamlanması da zamanuysuz olur.

Diğer bir seçenek olarak, MQTT istemcisi bir yayın yarattığında `MqttTopic.publish(byte [] payload, int qos, boolean retained)` yöntemindeki değiştirgelerden sizin için geçici bir ileti nesnesi yaratabilir.

Yayının "en az bir kez" ya da "tam olarak bir kez" hizmet kalitesi, `QoS=1` ya da `QoS=2` varsa, MQTT istemcisi `MqttClientPersistence` arabirimini çağırır. Uygulamaya bir teslim simgesi döndürülmeden önce iletiyi depolamak için `MqttClientPersistence` ' i çağırır.

Uygulama, `MqttDeliveryToken.waitForCompletion` yöntemini kullanarak, ileti sunucuya teslim edilinceye kadar bloke etmeyi seçebilir. Diğer bir seçenek olarak, uygulama engellemeden de devam edebilir. Yayınların engellemeden teslim olup olmadığını denetlemek istiyorsanız, `MqttCallback` istemcisini MQTT istemcisiyle gerçekleştiren bir geri çağrı sınıfı örneğini kaydedin. The MQTT client calls the `MqttCallback.deliveryComplete` method as soon as the publication has been delivered. Hizmet kalitesine bağlı olarak, teslim alma işlemi `QoS=0` için hemen hemen hemen hemen hemen hemen hemen hemen olabilir ya da `QoS=2` için biraz zaman alabilir.

Teslimlerin tamamlandıysa yoklama yapmak için `MqttDeliveryToken.isComplete` yöntemini kullanın. `MqttDeliveryToken.isComplete` değeri `false` ise, ileti içeriğini almak için `MqttDeliveryToken.getMessage` adını arayabilirsiniz. `MqttDeliveryToken.isComplete` çağrısının sonucu `true` ise, ileti atılır ve `MqttDeliveryToken.getMessage` çağrıldığında boş değerli gösterge kural dışı durumu yayınlanır. `MqttDeliveryToken.getMessage` ile `MqttDeliveryToken.isComplete` arasında yerleşik eşitleme yok.

İstemci, bekleyen teslim simgeleri almadan önce bağlantı kesilirse, istemcinin yeni bir eşgörünümü bağlanmadan önce teslim alma simgelerini sorgulayabilir. İstemci bağlanıncaya kadar, yeni teslimatlar tamamlanmaz ve `MqttDeliveryToken.getMessage()` i aramanız güvenlidir. Hangi yayınların teslim edilmediğini öğrenmek için `MqttDeliveryToken.getMessage()` yöntemini kullanın. Pending delivery tokens are discarded if you connect with `MqttConnectOptions.cleanSession` set to its default value, `true`.

## abone olunması

Bir MQTT aboneline göndermek üzere yayın yaratmaktan bir kuyruk yöneticisi ya da IBM MessageSight sorumludur. Kuyruk yöneticisi, bir MQTT istemcisi tarafından yaratılan bir abonelikte konu süzgecinin yayındaki konu dizisiyle eşleşip eşleşmediğini denetler. Eşleşme tam olarak eşleşebilir ya da eşleşme joker karakterler içerebilir. Yayın, kuyruk yöneticisi tarafından aboneye iletilmeden önce, kuyruk yöneticisi yayınlı ilişkili konu özniteliklerini denetler. Bir denetim konusu nesnesinin abone olma yetkisi olup olmadığını saptamak için, Genel arama karakterleri içeren bir konu dizisini kullanarak abone olunması başlıklı konuda açıklanan arama yordamlarından sonra gelir.

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", `QoS=2` ise, MQTT istemcisi, iletiyi alındığında iletiyi depolamak için `MqttClientPersistence` arabirimini çağırır. Daha sonra `MqttCallback.messageArrived()` u çağırır.

## İlgili kavramlar

Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback` 'u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirin.

İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

Teslim simgeleri

Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry 'u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

Message persistence in MQTT clients

Yayın iletileri, "en az bir kez" ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ 'e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayın hemen size iletilir.

## Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

## Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ içindeki konu dizgileriyle aynıdır.

## **MQTT istemcisi tarafından sağlanan hizmet kalitesi**

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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` ' nin bir öznesidir. Bu, `MqttMessage.setQoS` yöntemi tarafından ayarlanır.

`MqttClient.subscribe` yöntemi, bir konuyla ilgili olarak bir istemciye gönderilen yayınlara uygulanan hizmet kalitesini düşürebilir. Bir aboneye iletilen bir yayının hizmet kalitesi, yayının hizmet kalitesinden farklı olabilir. İki değerin alt değeri bir yayını iletmek için kullanılır.

### **En çok bir kez**

`QoS=0`

İleti en çok bir kez teslim edilir ya da hiç teslim edilmez. Ağ üzerindeki teslimi onaylanmadı.

İleti saklanmaz. İstemcinin bağlantısı kesildiyse ya da sunucu başarısız olursa ileti kaybedilebilir.

`QoS=0` , aktarma için en hızlı kiptir. bazen "ateş ve unutun" denilir.

The MQTT protocol does not require servers to forward publications at `QoS=0` to a client.

Sunucunun yayını aldığı süre içinde istemcinin bağlantısı kesilirse, sunucuya bağlı olarak, yayın atılabilir. Telemetry (MQXR) hizmeti, `QoS=0` ile gönderilen iletileri atmıyor. Bunlar, kalıcı olmayan iletiler olarak depolanır ve kuyruk yöneticisi durursa yalnızca atılır.

### **En az bir kez**

`QoS=1`

`QoS=1` , varsayılan aktarma kipidir.

İleti her zaman en az bir kez teslim edilir. Gönderen bir alındı bildirimini almazsa, bir alındı bildirimini alınincaya kadar ileti, yeniden DUP işaretiyle gönderilir. Sonuç olarak, alıcı birden çok kez aynı iletiyi gönderilebilir ve bu işlemi birden çok kez işleyebilirler.

İleti, işleninceye kadar gönderene ve alıcıya 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 bildirimini gönderir.

İleti, alıcıdan bir alındı bildirimini aldıktan sonra göndericiden silinir.

### **Tam bir kez**

`QoS=2`

İleti her zaman tam olarak bir kez teslim edilir.

İleti, işleninceye kadar gönderene ve alıcıya yerel olarak saklanmalıdır.

`QoS=2` , aktarım için en güvenli, ancak en yavaş kiptir. İleti göndericiden silinmeden önce, gönderen ile alıcı arasında en az iki çift iletim alır. İleti ilk iletilmeden sonra alıcıda işlenebilir.

İlk iletimler çiftinde, gönderen iletiyi iletir ve iletiyi sakladığı alıcısından alındı bildirimini alır.

Gönderen bir alındı bildirimini almazsa, bir alındı bildirimini alınincaya kadar ileti, yeniden DUP işaretiyle gönderilir.

İkinci iletimler çiftinde, gönderen, iletiyi işlemeyi tamamlayabileceğini belirtir, "PUBREL ".

Gönderen, "PUBREL " iletisine ilişkin bir alındı bildirimini almazsa, bir alındı bildirimini alınincaya kadar

"PUBREL" iletisi yeniden gönderilir. The sender deletes the message it saved when it receives the acknowledgment to the "PUBREL" message

İleti, iletiiyi yeniden işlememesi koşuluyla, iletiiyi birinci ya da ikinci aşamalarda işleyebilir. Alıcı bir aracıysa, iletiiyi abonelere yayınlar. Alıcı bir istemciyse, iletiiyi abone uygulamasına teslim eder. Alıcı, iletiiyi işlemeyi bitirdiği bir tamamlanma iletisini gönderene geri gönderir.

## **İlgili kavramlar**

### Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar dekoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback`'u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

### Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirin.

### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

### Teslim simgeleri

#### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry 'u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez"ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

### Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili `MqttMessage` örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

### Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayını hemen size iletilir.

### Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ'indeki konu dizgileriyle aynıdır.

## **Alıkonan yayınlar ve MQTT istemcileri**

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayını hemen size iletilir.

Bir konudaki yayınının alıkonulup tutulmadığını belirtmek için `MqttMessage.setRetained` yöntemini kullanın.

Alıkonan bir yayını yarattığınızda ya da güncellediğinizde, yayını 1 ya da 2 numaralı QoS ile gönderin. Bunu bir QoS /0 ile gönderdiğinizde, IBM MQ , kalıcı olmayan bir alıkonacı yayını yaratır. Kuyruk yöneticisi durursa, yayını korumaz.

Alıkonmayan bir yayını alıkonan bir yayına ilgili bir konuya yayınlıyorsanız, alıkonan yayını bundan etkilenmez. Yürürlükteki aboneler yeni yayını alır. Yeni aboneler önce alıkonan yayını alır, sonra da yeni yayınlar alır.

Bir ölçümün en son değerini kaydetmek için alıkonan bir yayını kullanabilirsiniz. Yeni aboneler bir konuya hemen en son ölçümün değerini alır. Yayın konusuna son abone olan abonedan bu yana yeni ölçümler alınmazsa ve abone yeniden abone olursa, abone yine konuyla ilgili en son tutulan yayını alır.

Alıkonan bir yayını silmek için iki seçeneğiniz vardır:

- **CLEAR TOPICSTR** MQSC komutunu çalıştırın.
- Sıfır uzunluklu bir alıkonan yayını yaratır. MQTT 3.1.1 belirtiminde belirtildiği gibi, bir konuya sıfır uzunluklu alıkonan bir ileti yayınlanırsa, o konu için alıkonan tüm iletiler temizlenir.

### **İlgili kavramlar**

#### Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback` u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

#### Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirir.

#### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

#### Teslim simgeleri

##### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

#### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez" ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

#### Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili `MqttMessage` örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

#### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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 abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

#### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ'indeki konu dizgileriyle aynıdır.

## Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci bağlantısı kesilirken abonelikler etkin kalabilir. Yayınlar yeniden bağlandığında istemciye gönderilir.

`MqttClient.subscribe` yöntemlerini kullanarak abonelikler oluşturun, bir ya da daha fazla konu süzgeci ve hizmet kalitesi parametreleri ile. Hizmet kalitesi parametresi, bir iletiyi almak için abonenin kullanmak üzere hazırlanmış olduğu maksimum hizmet kalitesini ayarlar. Bu istemciye gönderilen iletiler, daha yüksek hizmet kalitesiyle teslim edilemez. İleti yayımlandığında ve abonelik için belirtilen düzey olduğunda, hizmet kalitesi özgün değerinin alt değerine ayarlanır. İleti almak için varsayılan hizmet kalitesi: `QoS=1`, en az bir kez.

Abonelik isteği, `QoS=1` ile birlikte gönderilir.

Publications are received by a subscriber when the MQTT client calls the `MqttCallback.messageArrived` method. `messageArrived` yöntemi, iletinin aboneye yayımlandığı konu dizisini de geçirir.

`MqttClient.unsubscribe` yöntemlerini kullanarak bir aboneliği ya da bir kümeyi ya da abonelikleri kaldırabilirsiniz.

Bir IBM MQ komutu, bir aboneliği kaldırabilir. Abonelikleri IBM MQ Explorer komutunu 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` özneliğinin abonelikleri etkileme şeklini anlamamanın başka bir yolu da bunu kalıcı bir öznelik 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 yayınları 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.

Etkin aboneliklerle eşleşen yayınlar, yayımlandığı anda istemciye gönderilir. İstemcinin bağlantısı kesildiyse, aynı istemci tanıtıcısı ve `MqttConnectOptions.cleanSession` ile `false` değerine ayarlanmış aynı sunucuya yeniden bağlanıyorsa, istemci istemciye gönderilir.

Belirli bir istemciye ilişkin abonelikler istemci tanıtıcısı tarafından tanımlanır. İstemciyi farklı bir istemci aygıtından aynı sunucuya yeniden bağlayabilir ve aynı aboneliklere devam edebilir ve teslim edilmemiş yayınlarını alabilirsiniz.

## **İlgili kavramlar**

### Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback` 'u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

### Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirin.

### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

### Teslim simgeleri

#### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez" ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmasını uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

### Yayınlar

Yayınlar, bir konu dizgisiyle ilişkili `MqttMessage` örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

### Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayın hemen size iletilir.

### Topic strings and topic filters in MQTT clients

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ içindeki konu dizgileriyle aynıdır.

## **Topic strings and topic filters in MQTT clients**

Konu dizgileri ve konu süzgeçleri yayınlamak ve abone olmak için kullanılır. MQTT istemcilerinde konu dizgilerinin ve süzgeçlerin sözdizimi büyük ölçüde IBM MQ içindeki konu dizgileriyle aynıdır.

Konu dizgileri, abonelere yayın göndermek için kullanılır. `MqttClient.getTopic(java.lang.String topicString)` yöntemini kullanarak bir konu dizgisi yaratın.

Konu süzgeçleri konu başlıklarına abone olmak ve yayınları almak için kullanılır. Konu süzgeçleri joker karakterler içerebilir. Genel arama karakterleriyle birden çok konuya abone olabilirsiniz. Bir abonelik

yöntemi kullanarak bir konu süzgeci yaratın; örneğin, `MqttClient.subscribe(java.lang.String topicFilter)`.

## Konu dizgileri

Bir IBM MQ konu dizgisinin sözdizimi, [Konu Dizgileri](#)' nde açıklanmaktadır. MQTT konu dizgilerinin sözdizimi, MQTT client for Java için API belgelerindeki `MqttClient` sınıfında açıklanmıştır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. [MQTT istemci programlama başvurusu](#).

Her konu dizgisinin sözdizimi hemen hemen aynı. Dört küçük fark vardır:

1. Topic strings sent to IBM MQ by MQTT clients must follow the convention for queue manager names.
2. Uzunluk üst sınırı farklıdır. IBM MQ konu dizgileri 10.240 karakterle sınırlıdır. Bir MQTT istemcisi, 65535 byte 'a kadar konu dizgileri yaratabilir.
3. MQTT istemcisi tarafından yaratılan bir konu dizgisi boş değerli karakter içeremez.
4. IBM Integration Bus içinde, boş bir konu düzeyi ' . . . / / . . . ' geçersizdir. Boş konu düzeyleri IBM MQ tarafından desteklenmektedir.

IBM MQ yayınlama/abone olma gibi, `mqttv3` iletişim kuralının bir denetim konusu nesnesi kavramı yoktur. Bir konu nesnesinden ve konu dizgisinden bir konu dizgisi oluşturamazsınız. Ancak, bir konu dizgisi IBM MQ içindeki bir yönetim konularıyla eşlenir. Yönetimle ilgili konu ile ilişkilendirilen erişim denetimi, bir yayının konuya yayınlanıp yayınlanmadığını ya da atılıp atılmadığını belirler. Abonelere iletildiğinde bir yayına uygulanan öznitelikler, denetim konularının özniteliklerinden etkilenir.

## Konu süzgeçleri

Bir IBM MQ konu süzgecinin sözdizimi, [Konu tabanlı genel arama şeması](#)' nde açıklanmıştır. Bir MQTT istemcisiyle oluşturabileceğiniz konu süzgeçlerinin sözdizimi, MQTT client for Java için API belgelerindeki `MqttClient` sınıfında açıklanmıştır. MQTT istemci kitaplıklarına ilişkin istemci API belgelerine ilişkin bağlantılar için bkz. [MQTT istemci programlama başvurusu](#).

### İlgili kavramlar

#### Callbacks and synchronization in MQTT client applications

MQTT istemci programlama modeli iş parçacıklarını kapsamlı olarak kullanır. İş parçacıkları, bir MQTT istemci uygulamasını, iletilerin iletilmesinde ve sunucudan iletilmesinde gecikmelerden, mümkün olduğu kadar, mümkün kılacak kadar decoz. Yayınlar, teslim simgeleri ve bağlantı kaybedilen olaylar, `MqttCallback`' u gerçekleştiren bir geri çağrı sınıfındaki yöntemlere teslim edilir.

#### Temizleme oturumları

MQTT istemcisi ve telemetri (MQXR) hizmeti, oturum durumu bilgilerini korur. The state information is used to ensure "en az bir kez" and "aynen bir kez" delivery, and "aynen bir kez" receipt of publications. Oturum durumu, bir MQTT istemcisi tarafından yaratılan abonelikleri de içerir. Oturumlar arasında durum bilgilerini korumadan ya da bakım yapmadan bir MQTT istemcisi çalıştırabilir. Bağlanmadan önce `MqttConnectOptions.cleanSession` ayarını yaparak temizleme oturumu kipini değiştirin.

#### İstemci tanıtıcısı

İstemci tanıtıcısı, MQTT istemcisini tanımlayan 23 baytlık bir dizgidir. Her tanıtıcı, bir kerede yalnızca tek bir bağlı istemciye benzersiz olmalıdır. Tanıtıcı yalnızca, kuyruk yöneticisi adında geçerli olan karakterleri içermelidir. Bu kısıtlar içinde herhangi bir tanımlama dizilimini kullanabilirsiniz. İstemci tanıtıcılarının ayrılmasına ilişkin bir yordam ve seçilen tanıtıcısı ile bir istemci yapılandırılması için bir yöntemdir.

#### Teslim simgeleri

##### Son irade ve vasiyet yayını

Bir MQTT istemcisi bağlantısı beklenmeyen bir şekilde sona ererse, MQ Telemetry ' u "son bir irade ve ahit" yayını gönderecek şekilde yapılandırabilirsiniz. Yayının içeriğini ve gönderinin gönderileceği konuyu önceden tanımlayın. "Son irade ve vasiyet" bir bağlantı özelliğidir. İstemciyi bağlamadan önce bunu yaratın.

#### Message persistence in MQTT clients

Yayın iletileri, "en az bir kez" ya da "aynen bir kez" hizmet kalitesiyle gönderilirse, kalıcı olarak yapılır. İstemcide kendi kalıcılık mekanizmanızı uygulayabilir ya da istemciyle birlikte sağlanan varsayılan kalıcılık

mekanizmasını kullanabilirsiniz. Kalıcılık, istemciye ya da istemciden gönderilen yayınlar için her iki yönde de çalışır.

#### Yayınlar

Yayınlar, bir konu dizisiyle ilişkili MqttMessage örnekleridir. MQTT clients can create publications to send to IBM MQ, and subscribe to topics on IBM MQ to receive publications.

#### MQTT istemcisi tarafından sağlanan hizmet kalitesi

Bir MQTT istemcisi, yayınları IBM MQ ' e ve MQTT istemcisine teslim etmek için üç özellik sağlar: "en az 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.

#### Alıkonan yayınlar ve MQTT istemcileri

Bir konunun tek bir adı olabilir ve yalnızca bir kişi yayımlanabilir. Alıkonan bir yayını olan bir konuya ilişkin abonelik oluşturursanız, bu yayın hemen size iletilir.

#### Abonelikler

Bir konu süzgecini kullanarak, yayın konularına ilgi kaydetmek için abonelikler oluşturun. Bir istemci birden çok abonelik ya da genel arama karakterleri kullanan bir konu süzgeci içeren bir abonelik yaratabilir ve birden çok konuya ilgi gösterebilirler. Süzgeçlerle eşleşen konularla ilgili yayınlar istemciye gönderilir. Bir istemci 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 iletileri alır.

#### **İlgili kavramlar**

[“.NET ile WCF için IBM MQ özel kanalına giriş” sayfa 1292](#)

The custom channel for IBM MQ is a transport channel using the Microsoft Windows Communication Foundation (WCF) unified programming model.

[“WCF için IBM MQ özel kanallarını kullanma” sayfa 1297](#)

Overview of the information available for programmers using IBM MQ custom channels for Windows Communication Foundation (WCF).

[“WCF örneklerinin kullanılması” sayfa 1316](#)

Windows Communication Foundation (WCF) örnekleri, IBM MQ özel kanalının nasıl kullanılabileceğiyle ilgili bazı basit örnekler sağlar.

[FFST: WCF XMS İlk Hata Desteği Teknolojisi](#)

#### **İlgili görevler**

[Tracing the WCF custom channel for IBM MQ](#)

[IBM MQ sorunları için WCF özel kanalıyla ilgili sorun giderme](#)

## **.NET ile WCF için IBM MQ özel kanalına giriş**

The custom channel for IBM MQ is a transport channel using the Microsoft Windows Communication Foundation (WCF) unified programming model.

Microsoft.NET 3 'te kullanıma sunulan Microsoft Windows Communication Foundation çerçevesi, .NET uygulamalarının ve hizmetlerinin, bunları bağlamak için kullanılan iletim ve iletişim kurallarından bağımsız olarak geliştirilmesini, hizmetin ya da uygulamanın devreye alındığı ortama göre kullanılacak alternatif aktarımların veya yapılandırmaların kullanılmasını sağlar.

Bağlantılar, gereken bileşimi içeren bir kanal yığını oluşturarak WCF tarafından çalıştırma zamanında yönetilir:

- İletişim kuralı öğeleri: WS-\* standartları gibi destek protokollerine hiçbirinin, bir ya da daha fazlasının eklenebileceği, isteğe bağlı bir öğe kümesi.
- İletim kodlayıcısı: İletinin aktarım kanalı biçimine diziselleştirmesini denetleyen yığındaki zorunlu bir öğe.

- İletim kanalı: Serileştirilmiş iletiyi uç noktasına taşımaktan sorumlu yığın içinde zorunlu bir öge.

IBM MQ için özel kanal bir iletim kanalıdır ve bir WCF özel bağlaması kullanılarak uygulamanın gerektirdiği bir ileti kodlayıcı ve isteğe bağlı protokolle eşlenmiş olmalıdır. In this way, applications which have been developed to use WCF can use the custom channel for IBM MQ to send and receive data in the same way as they use the built-in transports provided by Microsoft, enabling simple integration with the asynchronous, scalable, and reliable messaging functions of IBM MQ. Desteklenen işlevlerin tam listesi için bkz. [“WCF Özel kanal özellikleri ve yetenekleri” sayfa 1297.](#)

## When and why do I use the IBM MQ custom channel for WCF?

You can use the IBM MQ custom channel to send and receive messages between WCF clients and services in the same way as the built-in transports provided by Microsoft, enabling applications to access the features of IBM MQ within the WCF unified programming model.

A typical usage pattern scenarios for the IBM MQ custom channel for WCF is as a non-SOAP interface for transmission of native IBM MQ messages.

## SOAP/Non-JMS dışı ileti (Pure MQMessage) biçimi kullanılarak gerçekleştirilen iletiler

When you use the IBM MQ custom channel for WCF as a non-SOAP interface for the transmission of native IBM MQ messages, the messages are carried by using the Non-SOAP/Non-JMS message (Pure MQMessage) format of IBM MQ.

WCF kullanıcıları hizmeti başlatabilir ya da diğer bir deyişle, hizmet kullanıcıları MQMessages kullanarak bir IBM MQ kuyruğuna ileti gönderebilirler. Uygulamalar, MQMD alanlarını ve bilgi yükünü alabilir ve ayarlayabilirler. When the message is available in IBM MQ MQ queues, this message can be processed by any WCF service or non-WCF applications such as C or Java applications that are running on Windows, UNIX or z/OS.

## WCF için IBM MQ özel kanalına ilişkin yazılım gereksinimleri

Bu konuda, WCF için IBM MQ özel kanalına ilişkin yazılım gereksinimleri özetlenmiştir. The IBM MQ custom channel for WCF can only connect to IBM WebSphere MQ 7.0 or higher queue managers.

## Yürütme ortamı gereksinimleri

- Microsoft.NET Framework v4.5.1 or higher must be installed on the host machine.
- *Java ve .NET Messaging ve Web Hizmetleri* , varsayılan olarak IBM MQ kuruluş programının bir parçası olarak kurulur. Bu bileşen, özel kanal için gereken .NET düzeneklerini Global Assembly Cache (Genel Derim Ön Belleği) içine kurar.

**Not:** IBM MQürünü kurmadan önce Microsoft .NET Framework V4.5.1 ya da üstü kurulu değilse, IBM MQ ürünü hatasız olarak devam eder, ancak IBM MQ classes for .NET kullanılabilir durumda değildir. If the .NET Framework is installed after you install IBM MQ, then the IBM MQ.NET assemblies must be registered by running the *WMQInstallDir\bin\amqiRegisterdotNet.cmd* script, where *WMQInstallDir* is the directory where IBM MQ is installed. Bu komut dosyası, gerekli düzenekleri Global Assembly Cache (GAC) içine kurar. Alınan eylemleri kaydetmek için %TEMP% dizininde oluşturulan *amqi\*.log* dosyaları kümesi. .NET , daha önceki bir sürümden v4.5.1 ya da daha yüksek bir sürümüne yükseltmişse, *amqiRegisterdotNet.cmd* komut kütüğünü yeniden çalıştırmak gerekli değildir; örneğin, .NET V3.5.

## Geliştirme ortamı gereksinimleri

- Microsoft Visual Studio 2008 ya da Windows Software Development Kit for .NET 4.5.1 ya da üstü.
- Örnek çözüm dosyalarını oluşturmak için, anasistem makinesinde Microsoft.NET Framework V4.5.1 ya da sonraki sürümü kurulu olmalıdır.

## IBM MQ Custom Channel for WCF: What's installer?

The custom channel for IBM MQ is a transport channel using the Microsoft Windows Communication Foundation (WCF) unified programming model. Özel kanal, kuruluşun bir parçası olarak varsayılan olarak kurulur.

### WCF için IBM MQ özel kanalı

Özel kanal ve bağımlılıkları, varsayılan olarak kurulu olan Java and .NET Messaging and Web Services bileşeni içinde yer alır. When upgrading IBM MQ from an earlier version than IBM MQ 8.0, the update installs the IBM MQ custom channel for WCF by default if the Java and .NET Messaging and Web Services component was previously installed in an earlier installation.

.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 Genel 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 196. Özel kanalı kullanmak için gerekli anahtar sınıfları		
	SOAP/JMS arabirimi (Var olan)	SOAP/Non-JMS arabirimi ( IBM MQ 8.0 ' tan)
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 İçerik Aktarıcısı	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementImporter	IBM.WMQ.WCF.WmqIbmTransportBindingElementImporter
İletim Bağ Tanımı Yapılandırması	IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig	IBM.WMQ.WCF.WmqIbmTransportBindingElementConfig
Örnekler (Oneway)	SimpleOneWayy_Client, SimpleOneWayy_Service	MQMessaging_Oneway_Client, MQMessaging_Oneway_service
Örnekler (RequestReply)	SimpleRequestreply_Client, SimpleRequestreply_Service	MQMessaging_RequestReply_Client, MQMessaging_RequestReply_Service

IBM.WMQ.WCF.dll, hem SOAP/JMS hem de SOAP/Non-JMS dışı arabirimleri destekler. Geliştirilen yeni uygulamaların IBM.WMQ.WCF düzeneği.

### MQSTR biçimli iletilerin gönderilmesi

▶ V 9.1.0

İstek iletişi MQSTR tipinse, yanıt iletişini MQSTR biçiminde göndermeyi seçebilirsiniz.

Yanıt iletişinin biçimini değiştirmek için ek bir URI parametresi **replyMessageFormat** kullanmanız gerekir. Desteklenen değerler şunlardır:

""

"" varsayılan değerdir.

Yanıt iletişi byte (MQMFT\_NONE) biçiminde olur. Örneğin:

```
"jms:/queue?  
destination=SampleQ@QM1&connectionFactory=binding(server)connectQueueManager(QM1)  
&initialContextFactory=com.ibm.mq.jms.NoJndi&replyDestination=SampleReplyQ&replyMessageFormat="
```

## MQSTR

Yanıt iletisi MQSTR (MQMFT\_STRING) biçiminde olur. 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** için değer büyük ve küçük harfe duyarlı değildir.
2. "" ya da MQSTR dışında herhangi bir değer kullanılması, geçersiz bir parametre değeri kural dışı durumuna neden olur.

## IBM MQ özel kanal örnekleri

Örnekler, IBM MQ özel kanalının WCF için nasıl kullanılabileceği ile ilgili bazı basit örnekler sağlar. Örnekler ve bunların ilişkili dosyaları, *MQ\_INSTALLATION\_PATH* \tools\dotnet\samples\cs\wcf dizininde bulunur; burada *MQ\_INSTALLATION\_PATH*, IBM MQ için kuruluş dizinidir. IBM MQ özel kanal örnekleriyle ilgili daha fazla bilgi için bkz. ["WCF örneklerinin kullanılması" sayfa 1316.](#)

### svcutil.exe.config

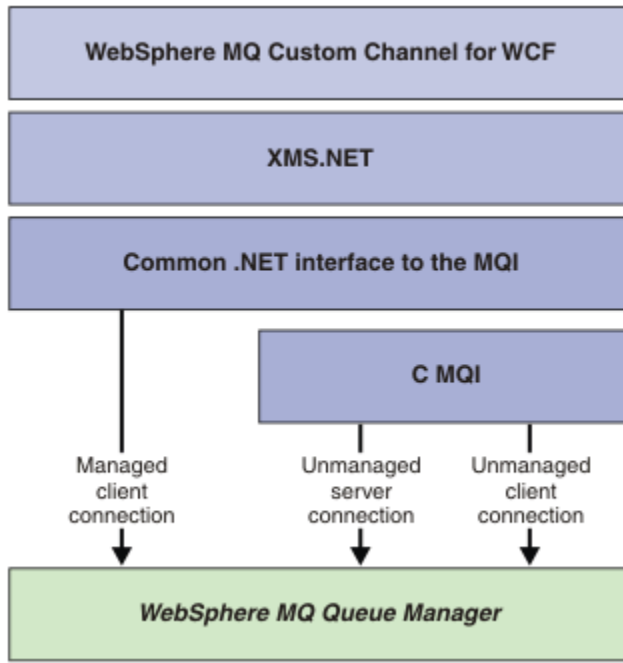
The *svcutil.exe.config* is an example of the configuration settings required to enable the Microsoft WCF *svcutil* client proxy generation tool to recognize the custom channel. *svcutil.exe.config* dosyası, *MQ\_INSTALLATION\_PATH* \tools\wcf\docs\examples\ dizininde bulunur; burada *MQ\_INSTALLATION\_PATH*, IBM MQ için kuruluş dizinidir. *svcutil.exe.config* kullanımıyla ilgili daha fazla bilgi için bkz. ["Çalışan bir hizmetteki meta verileri içeren svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturulması" sayfa 1313.](#)

## WCF mimarisi

WCF için IBM MQ özel kanalı, IBM Message Service Client for .NET (XMS .NET) API'nin üst kısmında tümleştirilmiştir.

## SOAP/JMS arabirimi

WCF mimarisi aşağıdaki şemada gösterilmektedir:



Şekil 154. SOAP/JMS arabirimi için WCF mimarisi

Gerekli tüm bileşenler varsayılan olarak ürün kuruluşuyla kurulur.

Üç bağlantı şunlardır:

- Yönetilen istemci bağlantıları
- Yönetilmeyen sunucu bağlantıları
- Yönetilmeyen istemci bağlantıları

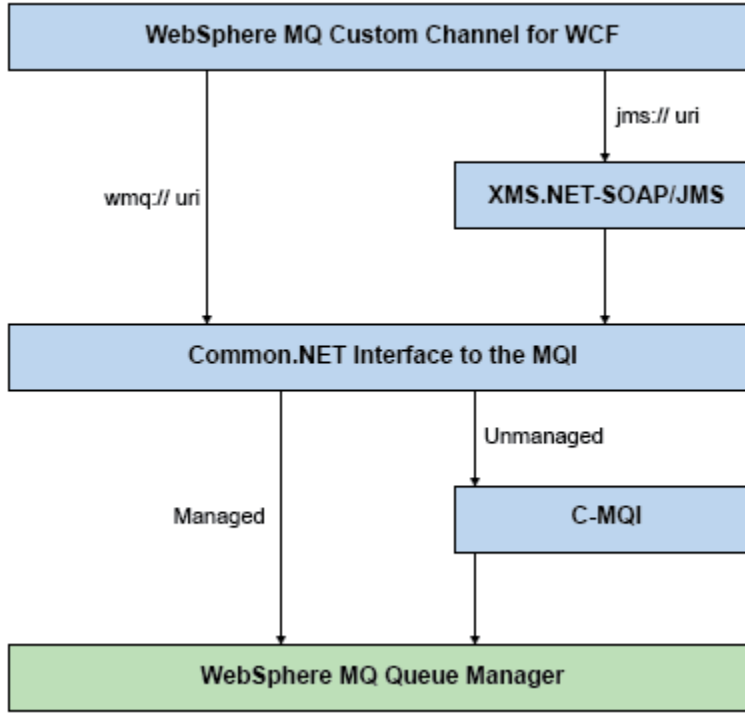
Bu bağlantılarla ilgili daha fazla bilgi için bkz. [“WCF bağlantısı seçenekleri” sayfa 1303.](#)

### SOAP/Non-JMS arabirimi

WCF için IBM MQ özel kanalı hem SOAP/JMS arabirimini ( IBM WebSphere MQ 7.0.1 ' dan edinilebilir) destekler. ve SOAP/Non-JMS arabirimi.

WCF mimarisi aşağıdaki şemada gösterilmektedir:





Şekil 155. SOAP/Non-JMS arabirimi için WCF mimarisi

## WCF için IBM MQ özel kanallarını kullanma

Overview of the information available for programmers using IBM MQ custom channels for Windows Communication Foundation (WCF).

The Microsoft Windows Communication Foundation underpins the web services and messaging support in the Microsoft.NET Framework 3. IBM WebSphere MQ 7.0 or later can be used as a custom channel within WCF in the .NET Framework 3 in the same manner as the built-in channels offered by Microsoft.

Özel kanalda taşınan iletiler, IBM WebSphere MQ 7.0 ya da sonraki yayın düzeylerinin JMS uygulaması üzerinden SOAP 'a göre biçimlendirilir. Applications can then communicate with services hosted by WCF or by the WebSphere SOAP over JMS service infrastructure.

## WCF Özel kanal özellikleri ve yetenekleri

WCF özel kanal özellikleri ve yetenekleriyle ilgili bilgi için aşağıdaki konuları kullanın.

### WCF özel kanal şekilleri

IBM MQ 'un Microsoft Windows Communication Foundation (WCF) özel kanalları içinde kullanılabileceği özel kanal şekillerine genel bakış.

The IBM MQ custom channel for WCF supports two channel shapes:

- Tek Yönlü
- İstek-yanıt

WCF, barındırılmakta olan hizmet sözleşmesine göre otomatik olarak kanal şeklini seçer.

Yalnızca **IsOneWay** parametresini kullanan yöntemleri içeren sözleşmeler, tek yönlü kanal şekli tarafından bakıma alınmakta, örneğin:

```
[OperationContract(IsOneWay = true)]
void printString(String text);
```

Tek yönlü ve istek-yanıt yöntemlerinin bir karışımının ya da tüm istek-yanıt yöntemlerinin bir karışımı içeren sözleşmeler, istek yanıtı kanalı şekli tarafından bakıma alınmadır. Örneğin:

```
[OperationContract]
int subtract(int a, int b);

[OperationContract(IsOneWay = true)]
void printString(string text);
```

**Not:** Tek yönlü ve istek yanıtı yöntemlerini aynı sözleşmede birlikte kullanırken, özellikle tek yönlü yöntemler, hizmetten boş bir yanıt alınıncaya kadar beklediğinden, bu davranışın, özellikle de karma bir ortam içinde çalışırken kullanılması amaçlanan gibi olduğundan emin olmalısınız.

## Tek yönlü kanal

WCF için IBM MQ tek yönlü özel kanal (örneğin, tek yönlü kanal şeklini kullanarak bir WCF istemcisinden ileti göndermek için kullanılır). Kanal, iletileri yalnızca tek bir yönde gönderebilir; örneğin, bir istemci kuyruk yöneticisinden bir WCF hizmetindeki bir kuyruğa gönderme yapabilir.

## İstek-yanıt kanalı

WCF için IBM MQ istek-yanıt özel kanalı kullanılır; örneğin, iletileri zamanuyumsuz olarak iki yönde göndermek için kullanılır; zamanuyumsuz ileti sistemi için aynı istemci yönetim ortamının kullanılması gerekir. Kanal, iletileri tek bir yöne (örneğin, bir istemci kuyruk yöneticisinden bir WCF hizmetindeki bir kuyruğa gönderebilir) gönderebilir ve daha sonra, WCF ' den istemci kuyruk yöneticisinde bir kuyruğa bir yanıt iletilisi gönderebilir.

## WCF URI değiştirge adları ve değerleri

SOAP/JMS arabirimi ve SOAP/Non JMS arabirimi için URI parametre adları ve değerleri.

## SOAP/JMS arabirimi

### connectionFactory

connectionFactory parametresi gerekli.

### initialContextFactory

initialContextFactory parametresi gerekli ve WebSphere Application Server ve diğer ürünlerle uyumluluk için "com.ibm.mq.jms.Nojndi" olarak ayarlanmalıdır.

## SOAP/Non JMS arabirimi

URI biçimi, MA93 belirtilmelerine uygun olarak gösterilmektedir. IBM MQ IRI belirtilmelerine ilişkin ek bilgi için SupportPac - MA93 başlıklı konuya bakın.

## 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 isteğinde bulunanın, 1414 kapısında example.com adlı bir makineye IBM MQ TCP istemci bağlama bağlantısını kullanabileceğini ve kalıcı istek iletilerini QM1kuyruk

yöneticisinden SampleQ adlı bir kuyruğa koyduğunu bildirir. IRI, hizmet sağlayıcının yanıtları SampleReplyQ olarak adlandırılan bir kuyruğa 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 bağlantıları için

WCF İstemcisi/Hizmeti 'ni kullanarak Secured (TLS) bağlantıları yapmak için, URI ' de uygun değerleri içeren aşağıdaki özellikleri ayarlayın. "" önekli olan tüm özellikler, güvenli bir bağlantı yapmak için zorunludur.

- **sslKeyRepository:** \*SYSTEM ya da \*USER
- \* **sslCipherSpec:** geçerli bir CipherSpec, örneğin, TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256.
- **sslCertRevocationCheck:** true ya da false.
- **sslKeyResetCount:** 32kbdeğerinden 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=ibmwebspheremqmm&sslkeyresetcount=45000"
```

### WCF için özel kanal garantili teslim

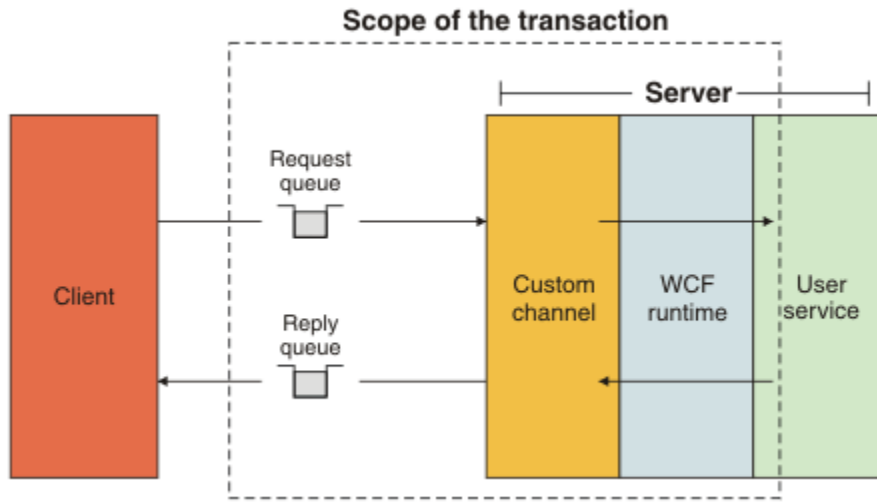
Bir hizmet isteğinin ya da yanıtının geçersiz olduğunu ve kaybedilmediğini garanti altına almak için Güvenli Teslim 'e garanti edilir.

Bir istek iletisi alındı ve herhangi bir yanıt iletisi, yürütme zamanı hatası durumunda geri döndürülebilecek bir yerel hareket eşitleme noktası altında gönderildi. Bu başarısızlıklara ilişkin örnekler: Hizmet tarafından yayınlanan işlenemeyen bir kural dışı durum, iletiyi hizmete gönderememek ya da yanıt iletisinin teslim edilmemesi.

AssuredDelivery , bir hizmet sözleşmesinde alınan istek iletilerinin ve bir hizmetten gönderilen herhangi bir yanıt iletisinin, çalıştırma zamanı hatası durumunda kaybedilmediğini garanti etmek için bir hizmet sözleşmesinde belirtilebilecek olan güvenli teslim özneliğinden söz eder.

Sistem arızası ya da güç kesintisi durumunda iletilerin aynı zamanda korunmasını sağlamak için iletiler kalıcı olarak gönderilmelidir. Kalıcı iletileri kullanmak için, istemci uygulamasının uç noktası URI 'sında belirtilen bu seçeneğe sahip olması gerekir.

Dağıtımli hareketler desteklenmez ve hareketin kapsamı, IBM MQtarafından gerçekleştirilen istek ve yanıt iletisi işlemenin ötesine geçmiyor. Hizmet içinde gerçekleştirilen tüm işler, iletinin yeniden alınmasına neden olan bir hatanın sonucu olarak yeniden çalıştırılabilir. Aşağıdaki çizge işlemin kapsamını gösterir:



Assured delivery is enabled by applying the `AssuredDelivery` attribute to the service class as shown in the following example:

```
[AssuredDelivery]
class TestCalculatorService : IWMQSampleCalculatorContract
{
    public int add(int a, int b)
    {
        int ans = a + b;
        return ans;
    }
}
```

`AssuredDelivery` özneteliğini kullanırken, aşağıdaki noktalardan haberdar olmalısınız:

- Bir kanal, bir iletinin geriye işlenip geri gönderilip alınmadığını saptadığında, ileti bir zehir iletisi olarak kabul edilir ve yeniden işlenmek üzere istek kuyruğuna döndürülmez. Örneğin, alınan ileti doğru biçimlendirilmediyse ya da bir hizmete dağıtılamazsa. Bir hizmet işleminden atılan işlenmeyen kural dışı durumlar, ileti, istek kuyruğunun geriletme eşiği özelliği tarafından belirtilen süre üst sınırını yeniden teslim edilinceye kadar her zaman yeniden içerilir. Daha fazla bilgi için bkz. [“WCF özel kanal zehirli iletileri” sayfa 1301](#)
- Kanal, hareket bütünlüğünü zorlamak için tek bir yürütme parçası kullanarak, her istek iletisini bir atomik işlem olarak okuma, işleme ve yanıtlama işlemini gerçekleştirir. Hizmet işlemlerinin koştuzamanlı olarak çalışmasını sağlamak için kanal, WCF 'nin kanala ilişkin birden çok yönetim ortamı yaratmasını sağlar. İsteklerin işlenmesi için kullanılabilir olan kanal eşgörünümlerinin sayısı, `MaxConcurrentCalls` bağ tanımlama özelliği tarafından denetlenir. Daha fazla bilgi için bkz. [“WCF bağ tanımlama yapıları seçenekleri” sayfa 1309](#)
- Güvenli teslim işlevi, hem `IOperationInvoker` hem de `IErrorHandler` WCF genişletilebilirlik noktalarını kullanır. Bu genişletilebilirlik noktaları bir uygulama tarafından dışarıdan kullanılırsa, uygulamanın önceden kayıtlı genişletilebilirlik noktalarının çağrıldığından emin olması gerekir. `IErrorHandler` için bu işlemi yapmamanız, bildirilmemiş hatalarla sonuçlanabilir. `Failure to do so for IOperationInvoker can cause WCF to stop responding.`

### **WCF özel kanal güvenliği**

The IBM MQ custom channel for WCF supports the use of TLS only for unmanaged client connections to the queue manager.

TLS 'yi istemci kanalı tanımlama çizelgesinde (CCDT) bir giriş kullanarak belirtin. CCDTs ile ilgili ek bilgi için [Client channel definition](#) tablebaşlıklı konuya bakın.

### **WCF istemci kanal tanımlama çizelgeleri (CCDT)**

WCF için IBM MQ özel kanalı, istemci bağlantılarına ilişkin bağlantı bilgilerini yapılandırmak için istemci kanal tanımlama çizelgelerinin (CCDT) kullanımını destekler.

CCDT ' ler bu iki ortam deęişkeni aracılıęıyla denetlenir:

- MQCHLLIB , çizelgenin bulunduęu dizini belirtir.
- MQCHLTAB , çizelgenin dosya adını belirtir.

Bu ortam deęişkenleri tanımlandıysa, URI ' de belirtilen istemci bağlantısı ayrıntılarının üzerinde öncelięe sahip olur.

İstemci kanal tanımlama çizelgelerine ilişkin ek bilgi için [Client channel definition](#) tablebaşlıklı konuya bakın.

### **İlgili kavramlar**

[İstemci kanal tanımlama çizelgesi](#)

### **WCF özel kanal zehirli iletileri**

Bir hizmet bir istek iletisini işlerken başarısız olduęunda ya da yanıt kuyruęuna yanıt iletisi gönderemediğinde, ileti bir zehir iletisi olarak işlenir.

### **Zehir isteęi iletileri**

Bir istek iletisi işlenemezse, bu ileti bir zehir iletisi olarak işlem görür. Bu işlem, hizmetin yeniden işlenemeyen aynı iletiyi almasını engeller. İşlenemeyen bir istek iletisinin bir zehir iletisi olarak işlenmesini sağlamak için aşağıdaki durumlardan biri doęru olmalıdır:

- Geri çıkış sayısı, istek kuyruęunda belirtilen geriletme eşięini aştı; bu, yalnızca hizmet için güvenli teslim belirtildiğinde ortaya çıkar. Güvenli teslimata ilişkin daha fazla bilgi için bkz. "[WCF için özel kanal garantili teslim](#)" sayfa 1299
- İleti düzgün şekilde biçimlendirilmedi ve JMS iletisi üzerinden bir SOAP olarak yorumlanamadı.

### **Zehirli yanıt iletileri**

Bir hizmet yanıt kuyruęuna bir yanıt iletisi gönderemezse, yanıt iletisi bir zehir iletisi olarak işlem görür. Yanıt iletileri için bu işlem, yanıt iletilerinin daha sonra yardım sorununun belirlenmesi için alınmasını sağlar.

### **Zehirli ileti işleme**

Bir zehir iletisi için alınan işlem, kuyruk yöneticisi yapılanışına ve iletinin rapor seçeneklerinde belirlenen deęerlere baęlıdır. JMSüzerinde SOAP için, istek iletilerine varsayılan olarak aşağıdaki rapor seçenekleri ayarlanır ve yapılandırılmaz:

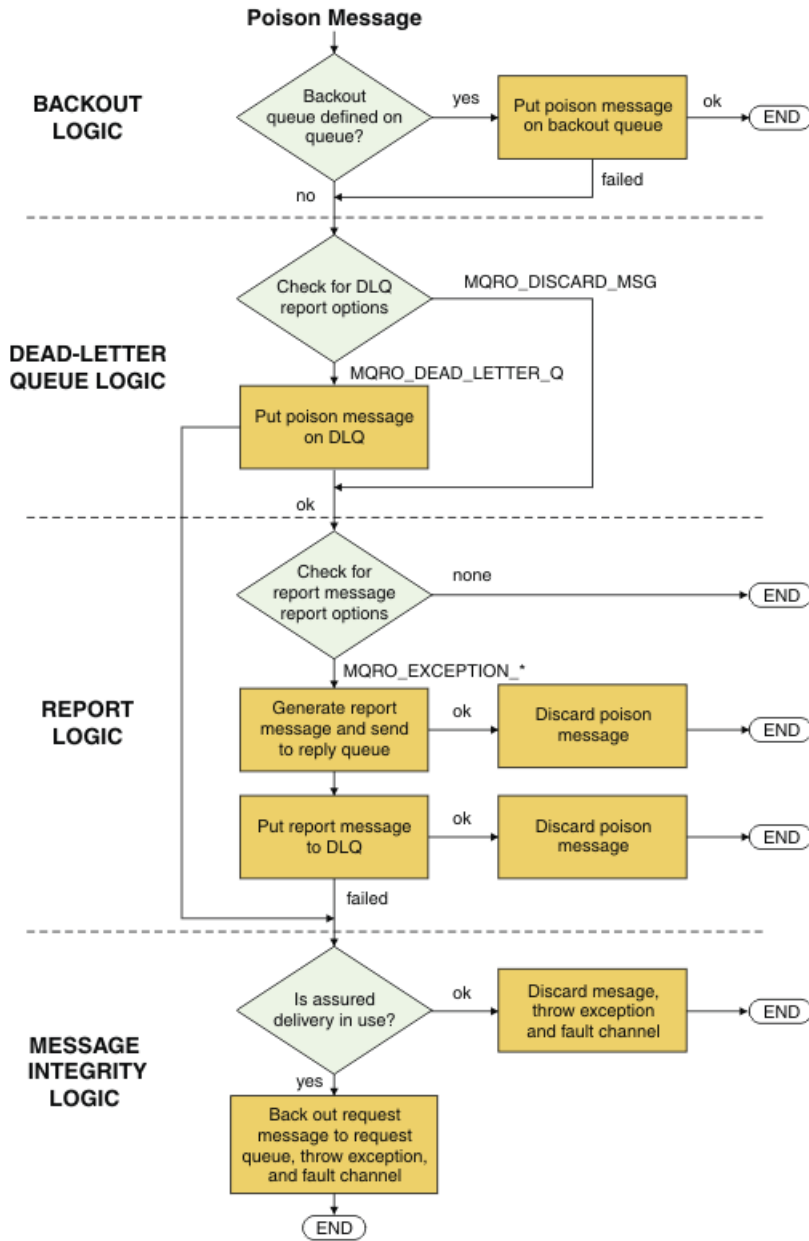
- MQRO\_EXCEPTION\_WITH\_FULL\_DATA
- MQRO\_EXPIRATION\_WITH\_FULL\_DATA
- MQRO\_DISCARD\_MSG

JMSüzerinde SOAP için, varsayılan olarak yanıt iletileri üzerinde aşağıdaki rapor seçeneęi ayarlanır ve yapılandırılmaz:

- MQRO\_DEAD\_LETTER\_Q

İleti, WCF olmayan bir kaynaktan geldiyse, o kaynaęa ilişkin belgelere bakın.

Aşağıdaki şekilde, olası işlemler ve zehirli ileti işleme başarısız olduęunda atılan adımlar gösterilmektedir:



### WCF uygulamaları için IBM MQ ileti yetenekleri

SOAP/Non-JMS dışı (yani, IBM MQ ) WCF uygulamaları için ileti yetenekleri sağlar.

SOAP/Non-JMS arabirimi için, WCF uygulamalarına ilişkin IBM MQ ileti yetenekleri aşağıdaki gibidir:

- WCF uygulamaları, herhangi bir IBM MQ uygulaması tarafından işlenebilen temel IBM MQ iletileri gönderip alabilir.
- WCF uygulamalarının, MQMD ' yi ve bilgi yükünü güncellemek için tam denetimi vardır.
- The WCF client can send IBM MQ messages that can be consumed by any IBM MQ clients, for example C, Java, JMS, and .NET clients.

SOAP/Non-JMS arabirimi için WCF, ileti bilgi yükünü ve ileti için MQMD ' yi ayarlamak için aşağıdaki sınıfları kullanmalıdır:

- WmqStringDize tipi bilgi yükü için ileti
- WmqBytesBytes tipinde bilgi yükü için ileti
- WmqXmlXML tipli bilgi yükü için ileti

İletinin bilgi yükünü ayarlamak için, bilgi yükü tipine bağlı olarak, WmqStringMessage, WmqBytesMessage ya da WmqXmlMessage class için **Data** özelliğini kullanın. Örneğin, String 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

There are three modes of connecting an IBM MQ custom channel for WCF to a queue manager. Gereklerinize en uygun bağlantı tipini göz önünde bulundurun.

Bağlantı seçeneklerine ilişkin daha fazla bilgi için bkz. [“Bağlantı farkları” sayfa 548](#)

WCF mimarisi hakkında daha fazla bilgi için bkz. [“WCF mimarisi” sayfa 1295](#)

## Yönetilmeyen istemci bağlantısı

Bu kipte yapılan bir bağlantı, yerel makineden ya da uzak bir makinede çalışan bir IBM MQ sunucusuna IBM MQ istemcisi olarak bağlanır.

To use the IBM MQ custom channel for WCF as an IBM MQ client, you can install it, with the IBM MQ MQI client, either on the IBM MQ server, or on a separate machine.

## Yönetilmeyen sunucu bağlantısı

Sunucu bağ tanımları kipinde kullanıldığında, WCF için IBM MQ özel kanalı, ağ üzerinden iletişim kurmak yerine kuyruk yöneticisi API 'sini kullanır. Bağ tanımları bağlantılarının kullanılması, IBM MQ uygulamaları için ağ bağlantılarını kullanmaktan daha iyi başarımlar sağlar.

Bağ tanımları bağlantısını kullanmak için, IBM MQ sunucusunda WCF için IBM MQ özel kanalını kurmalısınız.

## Yönetilen istemci bağlantısı

Bu kipte yapılan bir bağlantı, yerel makineden ya da uzak bir makinede çalışan bir IBM MQ sunucusuna IBM MQ istemcisi olarak bağlanır.

Bu kipe bağlanan .NET 3 için IBM MQ özel kanal sınıfları .NET tarafından yönetilen kodda kalır ve yerel hizmetlere çağrılar yapmaz. Yönetilen kodla ilgili daha fazla bilgi için Microsoft belgelerine bakın.

Yönetilen istemciyi kullanmak için bir dizi sınırlama vardır. Bu sınırlamalarla ilgili daha fazla bilgi için bkz. [“Yönetilen istemci bağlantıları” sayfa 548](#).

## WCF için IBM MQ özel kanalının yaratılması ve yapılandırılması

WCF 'nin IBM MQ özel kanalları, Microsoft tarafından sunulan iletim WCF kanallarıyla aynı şekilde çalışır. WCF için IBM MQ özel kanalı, iki yöntemden birinde oluşturulabilir.

### Bu görev hakkında

IBM MQ özel kanalı WCF ile bir WCF iletim kanalı olarak tümleşir ve bir ileti kodlayıcı ve isteğe bağlı iletişim kuralı kanallarıyla eşleşmiş olmalıdır; böylece, uygulama tarafından kullanılacak tam bir kanal yığını yaratabilir. Tam kanal yığınının başarıyla yaratılmasına ilişkin iki öge gereklidir:

1. Bağlama 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 tanımlamasının bir WCF özel bağlaması biçiminde yaratılması gerekir.

2. Uç nokta tanımlaması: Hizmet sözleşmesini bağ tanımlamasıyla bağlar ve uygulamanın bağlanabileceği yeri tanımlayan gerçek bağlantı URI 'sini de sağlar. For the custom channel, the URI is in the form of a SOAP over JMS URI.

Bu tanımlamalar farklı iki yoldan biriyle yaratılabilir:

- Yönetimsel olarak; tanımlar, bir uygulama yapılandırma dosyasında (örneğin: `app.config`) ayrıntılar sağlanarak oluşturulur.
- Programlı olarak; tanımlamalar doğrudan uygulama kodundan yaratılır.

Tanımları yaratmak için kullanılacak yöntemin, uygulamanın gereklerine göre aşağıdaki gibi olması gerekir:

- Yapılandırma için Yönetimle görevli yöntem, uygulamayı yeniden oluşturmadan hizmetin ayrıntılarını ve istemci konuşlandırmasını değiştirme esnekliğini sağlar.
- Yapılandırmaya ilişkin Programmatik yöntem, yapılandırma hatalarından daha fazla koruma sağlar ve yürütme sırasında dinamik olarak bir yapılandırma oluşturma yeteneği sağlar.

### **Bir uygulama yapılandırma dosyasında bağ tanımlaması ve uç nokta bilgileri sağlayarak bir WCF özel kanal yönetimi yaratılması**

WCF için IBM MQ özel kanalı, bir iletim düzeyi WCF kanalına sahip. Bir uç nokta ve bağ tanımlaması, özel kanalı kullanmak için tanımlanmalıdır ve bu tanımlar, bir uygulama yapılandırma dosyasında bağ tanımlaması ve uç nokta bilgileri sağlanarak yapılabilir.

Bir iletim düzeyi WCF kanalı olan WCF için IBM MQ özel kanalını yapılandırmak ve kullanmak için, bir bağ tanımlaması ve bir uç nokta tanımlaması tanımlanmalıdır. Bağlama, kanala ilişkin yapılandırma bilgilerini bulundurmaya ve uç nokta tanımlaması bağlantı ayrıntılarını içerir. Bu tanımlamalar iki şekilde yaratılabilir:

- Burada açıklandığı gibi, doğrudan uygulama kodundan programlı olarak: [“Bağ tanımlaması ve uç nokta bilgilerini programlı olarak gizleyerek bir WCF özel kanalı yaratılması” sayfa 1306](#)
- Aşağıdaki yordamda açıklandığı gibi, ayrıntıları bir uygulama yapılandırma dosyasında sağlayarak yönetimsel olarak.

İstemci ya da hizmet uygulaması yapılanış kütüğü yaygın olarak `yourappname.exe.config` adını taşır; burada *yakınadı* uygulamanızın adıdır. Uygulama yapılandırma dosyası, Microsoft hizmet yapılandırması düzenleyicisi aracı `SvcConfigEditor.exe` adlı aracı aşağıdaki şekilde kullanarak en kolay şekilde değiştirilir:

- `SvcConfigEditor.exe` yapılandırma düzenleyicisi aracını başlatın. Aracın varsayılan kuruluş yeri şudur: `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 bağlama ögesi uzantısı ekleyin**

1. Menüü açmak için **Gelişmiş > Uzantı > bağlayıcı ögesi** seçeneğini sağ tıklayın ve **Yeni** seçeneğini belirleyin.
2. Bu çizelgede gösterildiği gibi, alanları doldurun:

Alan	Değer
NAME	IBM.XMS.WCF.SoapJmsIbmTransportChannel
Tür	Genel Assembly Cache (GAC) içinde IBM.XMS.WCF.dll seçeneğine gidin ve IBM.XMS.WCF.SoapJmsIbmTransportBindingElementConfig seçeneğini belirleyin.



## Adım 2: Bir WCF ileti kodlayıcısıyla özel kanalı çiftleyen özel bir bağ tanımlaması yaratın.

1. Menüü açmak için **Bağlamalar** 'ı sağ tıklayın ve **Yeni Bağ Tanımı Yapılandırması**' yi seçin.
2. Bu çizelgede gösterildiği gibi, alanları doldurun:

Çizelge 198. Yeni bağ tanımlama yapılanış alanları	
Alan	Değer
NAME	CustomBinding_WMQ
BindingElement 1	textMessageEncoding (MessageVersion: Soap11)
BindingElement 2	IBM.XMS.WCF.SoapJmsIbmTransportChannel

## Adım 3: Bağ tanımlama özelliklerini belirtin

1. *IBM.XMS.WCF.SoapJmsIbmTransportChannel* : “Adım 2: Bir WCF ileti kodlayıcısıyla özel kanalı çiftleyen özel bir bağ tanımlaması yaratın.” sayfa 1305 içinde oluşturduğunuz bağlayıcıdan bağlama bağlanması.
2. Aşağıdaki yerde açıklandığı gibi, özelliklerin varsayılan değerlerinde gerekli değişiklikleri yapın: “WCF bağ tanımlama yapılanış seçenekleri” sayfa 1309

## Adım 4: Bir uç nokta tanımlaması yaratın

Create an endpoint definition which references the custom binding you created in: “Adım 2: Bir WCF ileti kodlayıcısıyla özel kanalı çiftleyen özel bir bağ tanımlaması yaratın.” sayfa 1305 and provides the connection details of the service. Bu bilgilerin belirtilme şekli, tanımın bir istemci uygulaması ya da hizmet uygulaması olup olmadığına bağlıdır.

Bir istemci uygulaması için, istemci kısmına aşağıdaki gibi bir uç nokta tanımlaması ekleyin:

1. Menüü açmak için **İstemci** > **Uç Noktalar** seçeneğini sağ tıklayın ve **Yeni İstemci Uç Noktası**' yi seçin.
2. Bu çizelgede gösterildiği gibi, alanları doldurun:

Çizelge 199. Yeni istemci uç noktası alanları	
Alan	Değer
NAME	Endpoint_WMQ
Address	<i>Hizmet için erişmek için gerekli WMQ bağlantı ayrıntılarını açıklayan SOAP/JMS URI. Daha fazla ayrıntı için bkz. “WCF uç noktası URI adresi biçimi için IBM MQ özel kanalı” sayfa 1308</i>
Bağ Tanımı	customBinding
BindingConfiguration	CustomBinding_WMQ
Sözleşme	<i>Hizmet sözleşmesi arabiriminizin adı</i>

Bir hizmet uygulaması için, hizmetler bölümüne aşağıdaki gibi bir hizmet tanımlaması ekleyin:

1. Menüü açmak için **Hizmetler** seçeneğini sağ tıklayın ve **Yeni Hizmet** seçeneğini belirleyin ve daha sonra, barındırılacak hizmet sınıfını seçin.
2. Yeni hizmetinize ilişkin **Uç Noktalar** bölümüne bir uç nokta tanımlaması ekleyin ve bu çizelgede gösterildiği gibi, alanları doldurun:

Çizelge 200. Yeni hizmet uç noktası alanları	
Alan	Değer
NAME	Endpoint_WMQ
Address	Hizmet için erişmek için gerekli WMQ bağlantı ayrıntılarını açıklayan SOAP/JMS URI. Daha fazla ayrıntı için bkz. <a href="#">“WCF uç noktası URI adresi biçimi için IBM MQ özel kanalı” sayfa 1308</a>
Bağ Tanımı	customBinding
BindingConfiguration	CustomBinding_WMQ
Sözleşme	Hizmet gerçekleştirme sınıfınızın adı

### **Bağ tanımı ve uç noktası bilgilerini programlı olarak gizleyerek bir WCF özel kanalı yaratılması**

WCF için IBM MQ özel kanalı, bir iletim düzeyi WCF kanalına sahip. Bir uç nokta ve bağ tanımı, özel kanalı kullanmak için tanımlanmalıdır ve bu tanımlar doğrudan uygulama kodundan programlanabilir.

Bir iletim düzeyi WCF kanalı olan WCF için IBM MQ özel kanalını yapılandırmak ve kullanmak için, bir bağ tanımı ve bir uç nokta tanımlaması tanımlanmalıdır. Bağlama, kanala ilişkin yapılandırma bilgilerini bulundurmaz ve uç nokta tanımlaması bağlantı ayrıntılarını içerir. Daha fazla bilgi için bkz. [“WCF örneklerinin kullanılması” sayfa 1316](#).

Bu tanımlamalar iki şekilde yaratılabilir:

- Administratively, by providing the details in an application configuration file, as described in [“Bir uygulama yapılandırma dosyasında bağ tanımı ve uç nokta bilgileri sağlayarak bir WCF özel kanal yönetimi yaratılması” sayfa 1304](#).
- Aşağıdaki alt konularda anlatıldığı gibi, doğrudan uygulama kodundan programsal olarak.

*Bağ tanımı ve uç noktası bilgilerinin programsal olarak tanımlanması: SOAP/JMS arabirimi*  
SOAP/JMS arabirimi için, doğrudan doğruya uygulama kodundan bir uç nokta ve bağ tanımı tanımlayabilirsiniz.

### **Bu görev hakkında**

Bağ tanımı ve uç noktası bilgilerini programlı olarak sağlamak için, aşağıdaki adımları tamamlayarak uygulamanızın gerekli kodunu ekleyin.

### **Yordam**

1. Uygulamanıza aşağıdaki kodu ekleyerek, kanala ilişkin iletim bağ tanımı öğesinin bir eşgörünümünü yaratın:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
```

2. Set any required binding properties, for example, by adding the following code to your application to set the ClientConnectionMode:

```
transportBindingElement.ClientConnectionMode = XmsWCFBindingProperty.AS_URI;
```

3. Uygulamanıza aşağıdaki kodu ekleyerek iletim kanalını bir ileti kodlayıcısıyla eşleştiren özel bir bağ tanımı yaratın:

```
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
```

#### 4. SOAP/JMS URI 'yı yaratın.

Hizmete erişmek için gereken IBM MQ bağlantı ayrıntılarını açıklayan SOAP/JMS URI 'si uç adres olarak sağlanmalıdır. Belirlediğiniz adres, kanalın bir hizmet uygulaması ya da istemci uygulaması için kullanılıp kullanılmadığına bağlıdır.

- İstemci uygulamaları için, SOAP/JMS URI 'si 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 'si aşağıdaki gibi bir URI olarak yaratılmalıdır:

```
Uri address = new Uri("jms:/queue?destination=SampleQ@QM1&connectionFactory=
connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

Uç noktalarla ilgili daha fazla bilgi için bkz. [“WCF uç noktası URI adresi biçimi için IBM MQ özel kanalı” sayfa 1308.](#)

*Bağ tanımlama ve uç noktası bilgilerinin programsal olarak tanımlanması: SOAP/Non-JMS arabirimi*  
SOAP/Non-JMS arabirimi için, doğrudan uygulama kodundan bir uç nokta ve bağ tanımlama programı tanımlayabilirsiniz.

## Bu görev hakkında

Bağ tanımlama ve uç noktası bilgilerini programlı olarak sağlamak için, aşağıdaki adımları tamamlayarak uygulamanızın gerekli kodunu ekleyin.

## Yordam

1. Uygulamanıza aşağıdaki kodu ekleyerek bir WmqBinding yaratın:

```
WmqBinding binding = new WmqBinding();
```

Bu kod, SOAP/Non-JMS dışı arabirim için gerekli olan WmqMsgEncodingElement ve WmqIbmTransportBinding öğesini çiftleyen bir bağ tanımlama yaratır.

2. Hizmete erişmek için gereken IBM MQ bağlantı ayrıntılarını açıklayan wmq:// URI adresini belirtin.  
wmq:// URI 'yı sağlamanın yolu, kanalın bir hizmet uygulaması ya da istemci uygulaması için kullanılıp kullanılmadığına bağlıdır.
- İstemci uygulamaları için, wmq:// URI değeri 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 değeri 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**

Konum ve bağlantı ayrıntıları sağlayan bir URI (Universal Resource Identifier; Evrensel Kaynak Tanıtıcısı) kullanılarak bir web hizmeti belirtildi. URI biçimi, SOAP/JMS arabirimini mi, yoksa SOAP/Non-JMS arabirimini mi kullandığınızı içerir.

### **SOAP/JMS arabirimi**

SOAP için IBM MQ iletiminde desteklenen URI biçimi, hedef hizmetlere erişirken SOAP/ IBM MQ ile özel parametreler ve seçenekler üzerinde kapsamlı bir denetim derecelerine izin verir. Bu biçim WebSphere Application Server ile ve CICS ile uyumlu ve IBM MQ ile bu ürünlerle bütünleşme kolaylaşmıştır.

URI sözdizimi aşağıdaki gibidir:

```
jms:/queue? name=value&name=value...
```

Burada ad bir değiştirge adı ve *değer* uygun bir değerdir ve ad = *değer* ögesi, başında ve işareti (&) olan ikinci ve sonraki yinelemeleriyle herhangi bir sayıda yinelenir.

Parametre adları büyük/küçük harf duyarlıdır, IBM MQ nesnelere adları gibi. Herhangi bir parametre bir kereden fazla belirtilirse, parametrenin son oluşumu, istemci uygulamalarının URI ' ye sonuna ekleyerek parametre değerlerini geçersiz kılacağı anlamına gelir. Herhangi bir ek tanınmayan parametre varsa, bunlar yok sayılır.

Bir URI ' yi bir XML dizisinde saklıyorsa, ve imi karakterini "&" olarak göstermeniz gerekir. Benzer şekilde, bir URI bir komut dosyasında kodlandıysa, tersi durumda kabuk tarafından yorumlanacak & gibi çıkış karakterlerine özen gösteriniz.

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 yer alıyor:

```
jms:/queue?destination=myQ&connectionFactory=()&targetService=MyService.asmx  
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Yalnızca gerekli parametreler sağlanır (yalnızca .NET hizmetleri için targetService gereklidir) ve connectionFactory seçeneği belirtilmez.

Bu Axis örneğinde, connectionFactory bir dizi seçenek içerir:

```
jms:/queue?destination=myQ@myRQM&connectionFactory=connectQueueManager(myconnQM)  
binding(client)clientChannel(myChannel)clientConnection(myConnection)  
&initialContextFactory=com.ibm.mq.jms.Nojndi
```

Bu Axis örneğinde, connectionFactory için sslPeerName seçeneği de belirtildi. sslPeerName 'in değeri, ad değer çiftlerini ve önemli 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
```

### **SOAP/Non-JMS arabirimi dışı**

SOAP/Non-JMS arabirimi için URI biçimi, hedef hizmetlere erişirken IBM MQ ' e özgü parametreler ve seçenekler üzerinde kapsamlı bir denetim derecesi sağlar.

URI sözdizimi aşağıdaki gibidir:

```
wmq://example.com:1415/msg/queue/INS.QUOTE.REQUEST@MOTOR.INS ?ReplyTo=msg/queue/INS.QUOTE.REPLY@BRANCH452&persistence=MQPER_NOT_PERSISTENT
```

Bu IRI, hizmet isteğinde bulunan bir istekte bulunan, 1415 kapısında example.com adlı bir makineye IBM MQ TCP istemci bağlama bağlantısı kullanabileceğini ve kalıcı istek iletilerini MOTOR.INSkuyruk yöneticisinden INS.QUOTE.REQUEST adlı bir kuyruğa koyduğunu bildirir. IRI, hizmet sağlayıcının yanıtları, INS.QUOTE.REPLY (kuyruk yöneticisi) BRANCH452. URI biçimi, SupportPac MA93 için belirtildiği gibidir. IBM MQ IRI belirteçlerine ilişkin ek bilgi için [SupportPac MA93: IBM MQ -Service Definition](#) başlıklı konuya bakın.

### **WCF bağ tanımlama yapılandırma seçenekleri**

Özel kanallar bağ tanımlama bilgilerine yapılandırma seçeneklerinin uygulanmasına ilişkin iki yöntem vardır. Özellikleri yönetimsel olarak ayarlayabilir ya da programsal olarak ayarlayabilirsiniz.

Bağ tanımlama yapılandırma seçenekleri iki farklı yoldan biriyle ayarlanabilir:

1. Yönetimsel olarak: Bağlama özelliği ayarlarının, uygulamalar yapılandırma dosyasındaki özel bağlama tanımlamasının iletim kısmında belirtilmesi gerekir; örneğin: `app.config`.
2. Programlı olarak: Özel bağ tanımlamanın kullanıma hazırlanması sırasında özelliği belirtmek için uygulama kodunun değiştirilmesi gerekir.

### **Bağ tanımlama özelliklerinin yönetimsel olarak ayarlanması**

Bağlama 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 programsal olarak ayarlanması**

İstemci bağlantı kipini belirtmek üzere bir WCF bağ tanımlama özelliği eklemek için, özel bağ tanımlamanın kullanıma hazırlanması sırasında özelliği belirtmek için hizmet kodunu 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 bađ tanımlama özellikleri

Çizelge 201. Yönetimsel olarak ya da programlı olarak ayarlarken, bađ tanımlama özelliklerinin deđerleri

Özellik adı	İstemci ya da Hizmet uygulaması	Yönetim deđeri	Programlı deđer	Tanım
maxBufferPoolSize	Her ikisi	0-64 bit işaretli tamsayı	0-64 bit işaretli tamsayı	Kanal örneđine ilişkin WCF ileti arabelleklerini saklamak için kullanılabilecek bellek büyüklüğü üst sınırını belirler.
maxMessageBoyutu	Her ikisi	1-32 bit işaretli tamsayı	1-32 bit işaretli tamsayı	Tek bir WCF iletisi için kullanılabilecek bellek üst sınırını belirler.
clientConnectionKipi	Her ikisi	0 (Varsayılan deđer) 1	AS_URI (Varsayılan deđer) CLIENT_UNMANAGED	İletim kanalının istemci bađlantı kipini belirtir.  0 , istemci bađlantı kipinin URI 'de belirtildiđi gibi olduđu anlamına gelir. Yalnızca istemci bađlantısı kullanılırsa kullanılır. İstemci bađlantı kipinin URI 'de belirtildiđi gibi olduđunu belirtir. İstemci bađlantı kipi belirlenmezse, varsayılan deđer 0 olur.  1 , istemci bađlantı kipinin yönetilmeyen bir istemci olduđu anlamına gelir. Yalnızca istemci bađlantısı kullanılırsa kullanılır.
MaxConcurrentÇađrılar	İstemci	menzil 0-2 147 483 647 Varsayılan deđer 16 'tır	menzil 0-2 147 483 647 Varsayılan deđer 16 'tır	Bu özellik, herhangi bir zamanda tek bir istemci yetkili sunucusunda yer alabilen kořutzamanlı işlem sayısı üst sınırını tanımlar. Daha fazla işlem bařlatılırsa, bunlar, devam eden bir işlem tamamlanıncaya kadar ya da zamanařımına kadar kuyruđa alınır. Bu ayar, tek bir yetkili sunucu tarafından tüketilebilecek iş parçacıklarını ve kaynakları denetlemek için kullanılabilir.  0 , bu sınırı kaldırır ve tüm işlemlerin eşzamanlı olarak denenmesini sađlar.

Çizelge 201. Yönetimsel olarak ya da programlı olarak ayarlarken, bağ tanımlama özelliklerinin değerleri (devamı var)

Özellik adı	İstemci ya da Hizmet uygulaması	Yönetim değeri	Programlı değer	Tanım
MaxConcurrentÇağrılar	Hizmet	menzil 1-2 147 483 647  Varsayılan değer 16 'tır	menzil 1-2 147 483 647  Varsayılan değer 16 'tır	<p>Bu özellik, yalnızca güvenli teslim özelliği etkinleştirilmişse kullanılır (güvenli teslimle ilgili daha fazla bilgi için bkz. “WCF için özel kanal garantili teslim” sayfa 1299 ). Verili uç nokta için aynı anda devam edebilen koşutzamanlı işlem sayısı üst sınırını belirtir.</p> <p>Bu ayarı değiştirirken dikkatli olun. Eşzamanlı işlemlerin her biri, özel kanalın yeni bir eşgörünümü ve iş parçacığı havuzundaki ilişkili iş parçacıklarının istekleri işleme almak için ek kaynakları gerektirir. Aşırı ayırma işlemi, üretkenliğinizi etkileyebilir ve performansı ağır bir şekilde etkileyebilir. Bu özelliği desteklemek için iş parçacığı havuzunun uygun yapılandırması yapılmalıdır.</p>

## WCF için hizmetler oluşturuluyor ve barındırılıyor

WCF hizmetlerini nasıl yaratacağınızı ve yapılandıracağını açıklayan Microsoft Windows Communication Foundation (WCF) hizmetlerine genel bakış.

WCF için IBM MQ özel kanalı ve bunu kullanan WCF hizmetleri aşağıdaki yöntemlerle barındırılabilir:

- Kendi kendine barındırma
- Windows Hizmeti

WCF için IBM MQ özel kanalı, Windows Process Activation Service 'te barındırılmaz.

Aşağıdaki konularda, ilgili adımları göstermek için bazı basit kendi kendine barındırma örnekleri sağlanmaktadır. Daha fazla bilgi ve en son ayrıntıları içeren Microsoft WCF çevrimiçi belgeleri, <https://msdn.microsoft.com> adresindeki Microsoft MSDN web sitesinde bulunabilir.

### **1. yöntemi kullanarak WCF hizmeti uygulamaları oluşturuluyor: Bir uygulama yapılandırma dosyası kullanılarak otomatik olarak barındırma**

Bir uygulama yapılandırma dosyası oluşturduktan sonra, hizmetin bir eşgörünümünü açın ve belirtilen kodu uygulamanızın üzerine ekleyin.

### **Başlamadan önce**

Create or edit an application configuration file for the service, as described in: “Bir uygulama yapılandırma dosyasında bağ tanımları ve uç nokta bilgileri sağlayarak bir WCF özel kanal yönetimi yaratılması” sayfa 1304

## Bu görev hakkında

1. Hizmet anasisteminde hizmetin bir eşgörünümünü somutlaştırır ve açar. Hizmet tipi, hizmet yapıları kütüğünde belirtilen hizmet tipiyle aynı olmalıdır.
2. Uygulamanınıza aşağıdaki kodu ekleyin:

```
ServiceHost service = new ServiceHost(typeof(MyService));
service.Open();
...
service.Close();
```

## 2. yöntemi kullanarak WCF hizmeti uygulamaları oluşturuluyor: Kendi kendine barındırma (self-hosting) programlı olarak doğrudan uygulamadan

Bağ tanımlama özelliklerini ekleyin, hizmet anasistemini gerekli hizmet sınıfının bir somut örneğiyle yaratın ve hizmeti açın.

## Başlamadan önce

1. Proje için özel kanal IBM.XMS.WCF.dll dosyasına bir başvuru ekleyin. IBM.XMS.WCF.dll, *WMQInstallDir\bin* dizininde, *WMQInstallDir*, IBM MQ 'in kurulu olduğu dizindir.
2. IBM.XMS.WCF ad alanına bir *using* deyimini ekleyin, örneğin: `using IBM.XMS.WCF`
3. Aşağıdaki yerde açıklandığı gibi, kanal bağ tanımı ögesinin ve uç noktasının bir eşgörünümünü oluşturun: "Bağ tanımı ve uç noktası bilgilerini programlı olarak gizleyerek bir WCF özel kanalı yaratılması" sayfa 1306

## Bu görev hakkında

Kanalın bağ tanımı özelliklerinde değişiklik yapılması gerekiyorsa, aşağıdaki adımları tamamlayın:

1. Bağlama özelliklerini aşağıdaki örnekte gösterildiği gibi `transportBindingElement` olarak ekleyin:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
Uri address = new Uri("jms:/queue?destination=SampleQQ@QM1&connectionFactory=
connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.NoJndi");
```

2. Hizmet anasistemini, gerekli hizmet sınıfının bir somut örneğiyle yaratın:

```
ServiceHost service = new ServiceHost(typeof(MyService));
```

3. Hizmeti açın:

```
service.AddServiceEndpoint(typeof(IMyServiceContract), binding, address);
service.Open();
...
service.Close();
```

## HTTP uç noktası kullanılarak meta verilerin gösterilmesi

WCF için IBM MQ özel kanalını kullanmak üzere yapılandırılmış bir hizmetin meta verilerinin gösterilmesine ilişkin yönergeler.

## Bu görev hakkında

Hizmetler meta verileri açıksa (örneğin, `svcutil` gibi araçların, örneğin, çevrimdışı WSDL dosyasından değil, doğrudan çalışan hizmetten erişebilmesi için), HTTP uç noktası ile hizmet meta verilerinin gösterilmesiyle gerçekleştirilmelidir. Bu ek uç noktayı eklemek için aşağıdaki adımlar kullanılabilir.



1. Meta verilerin ServiceHost' e gösterilmesi gereken temel adresi ekleyin, örneğin:

```
ServiceHost service = new ServiceHost(typeof(TestService),
    new Uri("http://localhost:8000/MyService"));
```

2. Hizmet açılmadan önce aşağıdaki kodu ServiceHost ' a ekleyin:

```
ServiceMetadataBehavior metadataBehavior = new ServiceMetadataBehavior();
metadataBehavior.HttpGetEnabled = true;
service.Description.Behaviors.Add(metadataBehavior);
service.AddServiceEndpoint(typeof(IMetadataExchange),
    MetadataExchangeBindings.CreateMexHttpBinding(), "mex");
```

## Sonuçlar

Meta veriler şu anda şu adreste bulunur: <http://localhost:8000/MyService>

## WCF için istemci uygulamaları oluşturuluyor

Microsoft Windows Communication Foundation (WCF) istemci uygulamaları oluşturulmasına ve oluşturulmasına genel bakış.

Bir WCF hizmeti için istemci uygulaması yaratılabilir; istemci uygulamaları genellikle uygulama tarafından kullanılabilir gereken yapıları ve yetkili sunucu dosyalarını yaratmak için Microsoft ServiceModel Metadata Utility Tool (Svcutil.exe) kullanılarak oluşturulabilmektedir.

### ***Çalışan bir hizmetteki meta verileri içeren svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturulması***

WCF için IBM MQ özel kanalını kullanmak üzere yapılandırılmış bir hizmet için istemci oluşturmak üzere Microsoft svcutil.exe aracını kullanma yönergeleri.

## Başlamadan önce

Doğrudan uygulama tarafından kullanılabilir zorunlu yapılandırma ve yetkili sunucu dosyalarına yaratmak için svcutil aracını kullanmak için üç önkoşul vardır:

- svcutil aracı başlatılmadan önce WCF hizmeti çalışıyor olmalıdır.
- WCF hizmeti, doğrudan çalışan bir hizmetten istemci oluşturmak için IBM MQ özel kanal uç noktası başvurularına ek olarak bir HTTP kapısı kullanarak meta verilerini açıklamalıdır.
- svcutil için yapılandırma verilerinde özel kanal kaydedilmelidir.

## Bu görev hakkında

Aşağıdaki adımlarda, IBM MQ özel kanalını kullanmak üzere yapılandırılmış bir hizmet için nasıl istemci oluşturulacağı açıklanır, ancak yürütme sırasında meta verileri ayrı bir HTTP kapısı aracılığıyla da açıklanır:

1. WCF hizmetini başlatın (hizmet, svcutil aracı başlatılmadan önce çalışır durumda olmalıdır).
2. Add the details from the svcutil.exe config file from the root of the installation, into the active svcutil configuration file, typically C:\Program Files\Microsoft SDKs\Windows\v6.0A\bin\svcutil.exe.config so svcutil recognizes the IBM MQ custom channel.
3. Bir komut isteminden svcutil komutunu 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 istemcisi projesine kopyalayın.

## Sonraki adım

Hizmet meta verileri doğrudan alınamazsa, istemci dosyalarını wsdl yerine wsdl 'den oluşturmak için svcutil kullanılabilir. Daha fazla bilgi için bkz. [“WSDL ile svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturuluyor” sayfa 1314](#)

## **WSDL ile svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturuluyor**

Hizmetin meta verileri kullanılmazsa, WSDL ' den WCF istemcileri oluşturmaya ilişkin yönergeler.

Bir çalışan hizmetten meta verilerden istemci oluşturmak için hizmetin meta verileri doğrudan alınamazsa, istemci dosyalarını WSDL ' den oluşturmak için svcutil kullanılabilir. IBM MQ özel kanalının kullanılacağını belirtmek için WSDL ' de aşağıdaki değişiklikler yapılması gerekir:

1. Aşağıdaki ad alanı tanımlarını 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:ExactlyOne>
            <wsp>All>
                <xms:xms xmlns:xms="http://sample.schemas.ibm.com/policy/xms" />
            </wsp>All>
        </wsp:ExactlyOne>
    </wsp:Policy>

    ...

</wsdl:definitions>
```

2. Bağ tanımları bölümünü değiştirerek, yeni ilke kısmına bakın ve temeldeki bağ tanımlama öğesinden transport tanımını kaldırın:

```
<wsdl:definitions ...>

    <wsdl:binding ...>
        <wsp:PolicyReference URI="#CustomerBinding_IWMQSampleContract_policy" />
        <[soap]:binding ... transport="" />
    </wsdl:binding>
</wsdl:definitions>
```

3. Bir komut isteminden svcutil komutunu 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 yapılandırma kütüğüyle istemci yetkili sunucusu kullanılarak WCF istemcisi uygulamaları oluşturulması**

### Başlamadan önce

Create or edit an application configuration file for the client, as described in: [“Bir uygulama yapılandırma dosyasında bağ tanımı ve uç nokta bilgileri sağlayarak bir WCF özel kanal yönetimi yaratılması” sayfa 1304](#)

## Bu görev hakkında

İstemci yetkili sunucusunun bir eşgörünümünü somutlaştırır ve açar. Oluşturulan yetkili sunucuya geçirilen deęiştirge, istemci yapılanış kütüğünde belirtilen uç nokta adıyla (örneğin Endpoint\_WMQ) aynı olmalıdır.

```
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();
    }
}
```

## Bir istemci yetkili sunucusunu programlı yapılandırmayla oluşturmak için WCF istemcisi uygulamaları oluşturulması

### Başlamadan önce

1. Proje için özel kanal IBM.XMS.WCF.dll dosyasına bir başvuru ekleyin. IBM.XMS.WCF.dll , *WMQInstallDir*\bin dizininde; burada *WMQInstallDir* , IBM MQ ' in kurulu olduğu dizindir.
2. IBM.XMS.WCF ad alanına bir *using* deyimi ekleyin, örneğin: `using IBM.XMS.WCF`
3. Şu yerde açıklandığı gibi, kanalın uç noktası ve uç noktasının bir eşgörünümünü oluşturun: “Bağ tanımı ve uç noktası bilgilerini programlı olarak gizleyerek bir WCF özel kanalı yaratılması” sayfa 1306

## Bu görev hakkında

Kanalın bağ tanımı özelliklerinde deęişiklik yapılması gerekiyorsa, aşağıdaki adımları tamamlayın.

1. Bağlama özelliklerini aşağıdaki şekilde gösterildiği gibi `transportBindingElement` içine ekleyin:

```
SoapJmsIbmTransportBindingElement transportBindingElement = new
SoapJmsIbmTransportBindingElement();
Binding binding = new CustomBinding(new TextMessageEncodingBindingElement(),
transportBindingElement);
EndpointAddress address =
    new EndpointAddress("jms:/queue?destination=SampleQQ@QM1&connectionFactory=
connectQueueManager(QM1)&initialContextFactory=com.ibm.mq.jms.Nojndi");
```

2. İstemci yetkili sunucusunu aşağıdaki şekilde gösterildiği gibi yaratın; burada *bağ tanımı* ve *uç noktası adresi* 1. adımda yapılandırılmış olan bağ tanımı ve uç noktası adresidir ve aşağıdaki adreslere geçilir:

```
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 kullanılabileceğiyle ilgili 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 şekli kullanarak bir WCF istemcisinden bir Windows Communication Foundation (WCF) hizmetini başlatmak için kullanılan IBM MQ özel kanalını gösterir.

#### Bu görev hakkında

Hizmet, konsola bir dizgi çıkırtıran tek bir yöntemi uygular. The client has been generated by using the `svcutil` tool to retrieve the service metadata from a separately exposed HTTP endpoint as described in [“Çalışan bir hizmetteki meta verileri içeren svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturulması” sayfa 1313](#)

Örnek, aşağıdaki yordamda açıklandığı gibi belirli kaynak adları ile yapılandırılmıştır.

If you must change the resource names, then you must also change the corresponding value on the client application in the `MQ_INSTALLATION_PATH` \tools\dotnet\samples\cs\wcf\samples\WCF\oneway\client\app.config file, and on the service application in the `MQ_INSTALLATION_PATH` \tools\dotnet\samples\cs\wcf\samples\WCF\oneway\service\TestServices.cs file, where `MQ_INSTALLATION_PATH` is the installation directory for IBM MQ. JMS uç noktası URI 'sını biçimlendirme hakkında daha fazla bilgi için IBM MQ ürün belgelerindeki *IBM MQ Transport for SOAP* başlıklı konuya bakın. Örnek çözümü ve kaynağı değiştirmeniz gerekirse, örneğin, Microsoft Visual Studio 8 ya da sonraki bir IDE ' ye (örneğin, Visual Studio 8 ya da üstü) gerek duyarsınız.

#### Yordam

1. *QM1* adlı bir kuyruk yöneticisi yaratın.
2. *SampleQ* adlı bir kuyruk hedefi yaratın.
3. İletişimci iletilerin beklediği şekilde 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 MQ için kuruluş 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 MQ için kuruluş dizinidir. İstemci uygulaması döngüleri beş kez beş ileti göndererek *SampleQ* ' a gönderilir.

#### Sonuçlar

The service application gets the messages from *SampleQ* and displays Hello World on the screen five times.

#### Sonraki adım

### Basit istek yanıtama istemcisi ve sunucu WCF örneği

This sample demonstrates the IBM MQ custom channel being used to start a Windows Communication foundation (WCF) service from a WCF client using a request-reply channel shape.

## Bu görev hakkında

Bu hizmet, iki sayı eklemek ve çıkarması için bazı basit hesap makinesi yöntemleri sağlar ve sonucu döndürür. The client has been generated by using the `svcutil` tool to retrieve the service metadata from a separately exposed HTTP endpoint as described in [“Çalışan bir hizmetteki meta verileri içeren svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturulması” sayfa 1313](#)

Örnek, açıklanan aşağıdaki yordamda olduğu gibi belirli kaynak adlarıyla yapılandırılmıştır. If you need to change the resource names, then you also need to change the corresponding value on the client application in the `MQ_INSTALLATION_PATH\Tools\wcf\samples\WCF\requestreply\client\app.config` file, and on the service application in the `MQ_INSTALLATION_PATH\Tools\wcf\samples\WCF\requestreply\service\RequestReplyService.cs` file, where `MQ_INSTALLATION_PATH` is the installation directory for IBM MQ. JMS uç noktası URI 'sını biçimlendirme hakkında daha fazla bilgi için IBM MQ ürün belgelerindeki *IBM MQ Transport for SOAP* başlıklı konuya bakın. Örnek çözümü ve kaynağı değiştirmeniz gerekirse, örneğin, Microsoft Visual Studio 8 ya da sonraki bir IDE ' ye (örneğin, Visual Studio 8 ya da üstü) gerek duyarsınız.

## Yordam

1. `QM1` adlı bir kuyruk yöneticisi yaratın.
2. `SampleQ` adlı bir kuyruk hedefi yaratın.
3. `SampleReplyQ` adlı bir kuyruk hedefi yaratın.
4. İletişimci iletilerin beklediği şekilde hizmeti başlatın: `MQ_INSTALLATION_PATH\Tools\wcf\samples\WCF\requestreply\service\bin\Release\SimpleRequestReply_Service.exe` dosyasını çalıştırın; burada `MQ_INSTALLATION_PATH` , IBM MQ için kuruluş 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 MQ için kuruluş dizinidir.

## Sonuçlar

İstemci çalıştırıldığı zaman, aşağıdaki işlem başlatılır ve dört kez yinelenir; bu nedenle, her bir yolla toplam beş ileti gönderilir:

1. İstemci, `SampleQ` ' a bir istek iletisi koyar ve yanıt bekler.
2. Hizmet, istek iletisini `SampleQ` ' den alır.
3. Hizmet, iletinin içeriğini kullanarak bazı değerleri ekler ve çıkarır.
4. Daha sonra hizmet, sonuçları `SampleReplyQ` ' da bir iletiye koyar ve istemcinin yeni bir ileti koymasını bekler.
5. İstemci `SampleReplyQ` ' dan iletiyi alır ve sonuçları ekranda görüntüler.

## Sonraki adım

### WCF client to a .NET service hosted by IBM MQ sample

Hem .NET hem de Java için örnek istemci uygulamaları ve örnek hizmet yetkili sunucusu uygulamaları sağlanır. Örnekler, hisse senedi fiyat teklifi isteği alan bir hisse senedi senedi hizmetine dayalıdır ve hisse senedi alıntısını sağlar.

## Başlamadan önce

The sample requires that the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM MQ and is accessible from a local queue manager.

When the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM MQ and is accessible from a local queue manager, additional configuration steps must be completed.

1. WMQSOAP\_HOME ortam deęişkenini IBM MQ kuruluř dizinine ayarlayın; örneęin: C:\Program Files\IBM\MQ
2. Java derleyicisi javac ' in kullanılabilir durumda olduęundan ve PATH deęişkeninde olduęundan emin olun.
3. axis.jar dosyasını, WebSphere kuruluř CD ' nin prereqs/axis dizininden IBM MQ üretim dizinine kopyalayın; örneęin: C:\Program Files\IBM\MQ\java\lib\soap
4. PATH deęişkenine ekleyin: MQ\_INSTALLATION\_PATH\Java\lib burada MQ\_INSTALLATION\_PATH , IBM MQ ' un kurulu olduęu dizini temsil eder; örneęin: C:\Program Files\IBM\MQ
5. Ensure that the location of .NET is specified correctly in MQ\_INSTALLATION\_PATH\bin\amqwcalls\cmd where MQ\_INSTALLATION\_PATH represents the directory where IBM MQ is installed, for example: C:\Program Files\IBM\MQ. .NET konumu örnek olarak belirtilebilir: set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin

Önceki adımlar tamamlandıca, hizmeti test edin ve çalıştırın:

1. Navigate to your SOAP over JMS working directory.
2. Doğrulama sınavmasını 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 olduęu dizini temsil eder.
  - AXIS için: MQ\_INSTALLATION\_PATH\Tools\soap\samples\runivt Dotnet2AxisClient hold ; burada MQ\_INSTALLATION\_PATH , IBM MQ ' in kurulu olduęu dizini gösterir.

hold bağımsız deęişkeni, sınavı tamamlandıktan sonra dinleyicilerin çalışır durumda kalmasını sağlar.

Bu yapılandırma sırasında hatalar bildirilirse, yordamın aşağıdaki şekilde yeniden başlatılabilmesi için tüm deęişiklikleri kaldırabilirsiniz:

1. Oluřturulan SOAP over JMS dizinini silin.
2. Kuyruk yöneticisini silin.

## Bu görev hakkında

This sample demonstrates a connection from a WCF client to the .NET SOAP over JMS sample service provided in IBM MQ using a one-way channel shape. Hizmet, konsola bir metin dizesi çıkaran basit bir StockQuote örneęi uygular.

The client has been generated by using WSDL to generate client files as described in [“WSDL ile svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturuluyor” sayfa 1314](#)

Örnek, aşağıdaki yordamda açıklandığı gibi belirli kaynak adları ile yapılandırılmıştır. If you need to change the resource names, then you must also change the corresponding value on the client application in the MQ\_INSTALLATION\_PATH\tools\wcf\samples\WMQNET\default\client\app.config file, and on the service application in the MQ\_INSTALLATION\_PATH\tools\wcf\samples\WMQNET\default\service\WmqDefaultSample\_StockQuoteDotNet.wsdl file, where MQ\_INSTALLATION\_PATH represents the installation directory for IBM MQ. JMS uç noktası URI 'sını biçimlendirme hakkında daha fazla bilgi için IBM MQ ürün belgelerindeki *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 MQ için kuruluř dizinini temsil eder.

İstemci uygulaması döngüleri beř kez örnek kuyruęa beř ileti gönderir.

## Sonuçlar

Hizmet uygulaması, örnek kuyruktan iletileri alır ve ekranda Hello World beş kez görüntüler.

## IBM MQ Sample tarafından barındırılan bir Axis Java hizmetine WCF istemcisi

Hem Java hem de .NET için örnek istemci uygulamaları ve örnek hizmet yetkili sunucusu uygulamaları sağlanır. Örnekler, hisse senedi fiyat teklifi isteği alan bir hisse senedi senedi hizmetine dayalıdır ve hisse senedi alıntısını sağlar.

### Başlamadan önce

This sample requires that the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM MQ and is accessible from a local queue manager.

When the .NET SOAP over JMS service hosting environment is correctly installed and configured in IBM MQ and is accessible from a local queue manager, additional configuration steps must be completed.

1. WMQSOAP\_HOME ortam değişkenini IBM MQ kuruluş dizinine ayarlayın; örneğin: C:\Program Files\IBM\MQ
2. Java derleyicisi javac ' in kullanılabilir durumda olduğundan ve PATH değişkeninde olduğundan emin olun.
3. axis.jar dosyasını WebSphere kuruluş CD ' nin 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. Ensure that the location of .NET is specified correctly in MQ\_INSTALLATION\_PATH\bin\amqwcallsdls.cmd where MQ\_INSTALLATION\_PATH represents the directory where IBM MQ is installed, for example: C:\Program Files\IBM\MQ. .NET konumu örnek olarak belirtilebilir: set msfwdir=%ProgramFiles%\Microsoft Visual Studio .NET 2003\SDK\v1.1\Bin

Önceki adımlar tamamlandıktan sonra, hizmeti test edin ve çalıştırın:

1. Navigate to your SOAP over JMS working directory.
2. Doğrulama sınavını 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 temsil eder.
  - 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, sınav tamamlandıktan sonra dinleyicilerin çalışır durumda kalmasını sağlar.

Bu yapılandırma sırasında hatalar bildirilirse, yordamın aşağıdaki şekilde yeniden başlatılmasını için tüm değişiklikleri kaldırabilirsiniz:

1. Oluşturulan SOAP over JMS dizinini silin.
2. Kuyruk yöneticisini silin.

### Bu görev hakkında

The sample demonstrates a connection from a WCF client to the Axis Java SOAP over JMS sample service provided in IBM MQ using a one-way channel shape. Hizmet, bir metin dizesini geçerli dizine kaydedilen bir dosyaya çıkararak basit bir StockQuote örneği uygular.

The client has been generated by using WSDL to generate client files as described in [“WSDL ile svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturuluyor” sayfa 1314](#)

Örnek, bu paragrafta açıklandığı gibi, belirli kaynak adlarıyla yapılandırılmış olmalıdır. If you need to change the resource names, then you must also change the corresponding value on the client application in the `MQ_INSTALLATION_PATH\tools\wcf\samples\WMQAxis\default\client\app.config` file, and on the service application in the `MQ_INSTALLATION_PATH\tools\wcf\samples\WMQAxis\default\service\WmqDefaultSample_StockQuoteDotNet.wsdl` file, where `MQ_INSTALLATION_PATH` represents the installation directory for IBM MQ.

## 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 MQ için kuruluş dizinini temsil eder.

İstemci uygulaması döngüleri beş kez örnek kuyruğa beş ileti gönderir.

## Sonuçlar

Hizmet uygulaması, örnek kuyruktan iletileri alır ve yürürlükteki dizinde bir dosyaya Hello World beş kez ekler.

## WCF client to Java service hosted by WebSphere Application Server sample

WebSphere Application Server için örnek istemci uygulamaları ve örnek hizmet yetkili sunucusu uygulamaları sağlanır. Bir istek-yanıt hizmeti de sağlanır.

## Başlamadan önce

Bu örnek, aşağıdaki IBM MQ yapılandırmasının kullanılmasını gerektirir:

<i>Çizelge 202. IBM MQ gerekli yapılandırma</i>	
Nesne	Gerekli ad
Kuyruk yöneticisi	QM1
Yerel kuyruk	HelloWorld
Yerel kuyruk	HelloWorldYanıtla

Bu örnek ayrıca, bir WebSphere Application Server 6 barındırma ortamının doğru olarak kurulup yapılandırılmasını gerektirir. WebSphere Application Server 6, varsayılan olarak IBM MQ 'a bağlanmak için bir bağ tanımlama kipi bağlantısı kullanır. Therefore WebSphere Application Server 6 must be installed on the same machine as the queue manager.

WAS ortamı yapılandırıldıktan sonra, aşağıdaki ek yapılandırma adımları tamamlanmalıdır:

1. WebSphere Application Server JNDI havuzunda aşağıdaki JNDI nesnelerini yaratın:
  - a. HelloWorldadlı bir JMS kuyruk hedefi
    - JNDI adını `jms/HelloWorld` olarak ayarlayın
    - Kuyruk adını `HelloWorld` olarak ayarlayın.
  - b. HelloWorldQCFadlı bir JMS kuyruk bağlantısı üreticisi
    - JNDI adını `jms/HelloWorldQCF` olarak ayarlayın
    - Kuyruk yöneticisi adını `QM1` olarak ayarlayın.
  - c. WebServicesReplyQCFadlı bir JMS kuyruk bağlantısı üreticisi
    - JNDI adını `jms/WebServicesReplyQCF` olarak ayarlayın
    - Kuyruk yöneticisi adını `QM1` olarak ayarlayın.
2. Aşağıdaki yapılandırlarla WebSphere Application Server içinde HelloWorldPort adlı bir Message Listener kapısı yaratın:



- Bağlantı üreticisi JNDI adını `jms/HelloWorldQCF` olarak ayarlayın.
  - Hedef JNDI adını `jms/HelloWorld` olarak ayarlayın
3. Web hizmeti `HelloWorldEJB`.ear uygulamasını WebSphere Application Server 'a aşağıdaki şekilde kurun:
- Uygulamalar > Yeni Uygulama > Yeni Kurumsal Uygulama** öğelerini tıklattın.
  - `MQ_INSTALLATION_PATH\tools\wcf\samples\WAS\HelloWorldsEJB`.ear 'a gidin; burada `MQ_INSTALLATION_PATH` , IBM MQ' in kuruluş dizinidir.
  - Sihirbazdaki varsayılan seçeneklerden herhangi birini değiştirmeyin ve uygulama kurulduktan sonra uygulama sunucusunu yeniden başlatın.

WAS yapılandırması tamamlandığında, hizmeti bir kez çalıştırarak sınavarak aşağıdakileri gerçekleştirin:

1. Navigate to your Soap over JMS working directory.
2. Örneği çalıştırmak için bu komutu girin: `MQ_INSTALLATION_PATH\tools\wcf\samples\WAS\TestClient.exe` burada `MQ_INSTALLATION_PATH` , IBM MQ' in kuruluş dizinidir.

## Bu görev hakkında

Bu örnek, bir WCF istemcisinden, request içinde yer alan WCF örneklerinde sağlanan WebSphere Application Server SOAP over JMS örnek hizmetine yönelik bir bağlantıyı göstermektedir; bu örnek bir istek yanıt kanalı şekli kullanılarak IBM MQ' e eklenmiştir. Messages flow between WCF and the WebSphere Application Server using IBM MQ queues. Hizmet, bir dizgiyi alan ve istemciye bir selamlama döndüren `HelloWorld(...)` yöntemini uygular.

“Çalışan bir hizmetteki meta verileri içeren svcutil aracını kullanarak bir WCF istemcisi yetkili sunucusu ve uygulama yapılandırma dosyaları oluşturulması” sayfa 1313 içinde açıklandığı gibi, ayrı olarak gösterilen bir HTTP uç noktasından hizmet meta verilerini almak için svcutil aracı kullanılarak istemci üretildi.

Örnek, aşağıdaki yordamda açıklandığı gibi belirli kaynak adları ile yapılandırılmıştır.

If you need to change the resource names, then you must also change the corresponding value on the client application in the `MQ_INSTALLATION_PATH\tools\wcf\samples\WAS\default\client\app.config` file, and on the service application in the `MQ_INSTALLATION_PATH\tools\wcf\samples\WAS\HelloWorldsEJB`.ear where `MQ_INSTALLATION_PATH` is the installation directory of IBM MQ.

The service and client are based upon the service and client outlined in the IBM Developer article *Building a JMS web service using SOAP over JMS and WebSphere Studio*. For more information about developing SOAP over JMS web services that are compatible with the IBM MQ WCF custom channel, see [https://www.ibm.com/developerworks/websphere/library/techarticles/0402\\_du/0402\\_du.html](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 MQ için kuruluş dizinidir.

İstemci uygulaması hizmet yöntemlerinin her ikisini de aynı anda başlatır, örnek kuyruğa iki ileti gönderir.

## Sonuçlar

Hizmet uygulaması, iletileri örnek kuyruktan alır ve istemci uygulaması çıkışlarının konsola verdiği `HelloWorld(...)` yöntemi çağırısına yanıt sağlar.



## 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. Bulduğunuz yerde kullanıma sunulan ürün ve hizmetleri yerel IBM müşteri temsilcisinden ya da çözüm ortağınızdan öğrenebilirsiniz. Bir IBM ürün, program ya da hizmetine gönderme yapılması, açık ya da örtük olarak yalnızca o IBM ürünü, programı ya da hizmetinin kullanılabilirliğini 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

**Aşağıdaki paragraf, İngiltere ya da bu tür hükümlerin yerel yasalarla uyuşmadığı diğer ülkelerde geçerli değildir:** INTERNATIONAL BUSINESS MACHINES CORPORATION BU YAYINI, HAK İHLALİ YAPILMAYACAĞINA DAİR GARANTİLERLE TİCARİLİK VEYA BELİRLİ BİR AMACA UYGUNLUK İÇİN ZİMNİ GARANTİLER DE DAHİL OLMAK VE FAKS BUNLARLA SINIRLI OLMAMAK ÜZERE AÇIK YA DA ZİMNİ HİÇBİR GARANTİ VERMEKSİZİN "OLDUĞU GİBİ" ESASIYLA SAĞLAMAKTADIR. 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 Highway 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 kullanılacak tüm lisanslı malzeme, IBM tarafından, IBM Müşteri Sözleşmesi, IBM Uluslararası Program Lisansı Sözleşmesi ya da eşdeğer herhangi bir sözleşmenin kayıt ve koşulları altında 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 doğrulamalıdır.

IBM dışı ürünlerle ilgili bilgiler, bu ürünleri sağlayan firmalardan, bu firmaların yayın ve belgelerinden ve genel kullanıma açık diğer kaynaklardan alınmıştır. IBM bu ürünleri sınınamamıştır 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 sınınamamıştır. 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

Programlama arabirimi bilgileri (sağlandıysa), bu programla birlikte kullanılmak üzere uygulama yazılımları yaratmanıza yardımcı olmak üzere hazırlanmıştır.

Bu kitap, müşterinin WebSphere MQ hizmetlerini edinmek üzere program yazmasına olanak tanıyan, amaçlanan programlama arabirimlerine ilişkin bilgiler içerir.

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, değişiklik söz konusu olduğunda kullanılır.

## 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 listesini Web üzerinde "Telif hakkı ve ticari marka bilgileri" [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml) adresinde bulabilirsiniz. Diğer ürün ve hizmet adları IBM'in veya diğer şirketlerin ticari markaları olabilir.

Microsoft ve Windows, Microsoft Corporation'ın ABD ve/veya diğer ülkelerdeki ticari markalarıdır.

UNIX, The Open Group şirketinin ABD ve diğer ülkelerdeki tescilli ticari markasıdır.

Linux, Linus Torvalds'ın ABD ve/ya da diđer ÷lkelerdeki tescilli ticari markasıdır.

Bu ÷r÷n, Eclipse Project (<http://www.eclipse.org/>) tarafından geliřtirilen yazılımları ierir.

Java ve Java tabanlı t÷m markalar ve logolar, Oracle firmasının ve/ya da iřtiraklerinin markaları ya da tescilli markalarıdır.







Parça numarası:

(1P) P/N: